WP-MIRROR 0.6 Reference Manual

Dr. Kent L. Miller

December 16, 2013

To Tylery

WP-MIRROR 0.6 Reference Manual

Legal Notices

The WP-MIRROR logotype (see margin) includes a sunflower that is derived from 119px-Mediawiki_logo_sunflower_Tournesol_5x_rev2.png, which is in the public domain. See http://en.wikipedia.org/wiki/File:Mediawiki_logo_sunflower_Tournesol_5x.png.

Abstract

The Wikimedia Foundation offers wikipedias in nearly 300 languages. In addition, the WMF has several other projects (e.g. wikibooks, wiktionary, etc.) for a total of around 1000 wikis.

WP-MIRROR is a free utility for mirroring any desired set of these wikis. That is, it builds a wiki farm that the user can browse locally. Many users need such off-line access, often for reasons of mobility, availability, and privacy. They currently use kiwix which provides selected articles and thumbnail images. WP-MIRROR builds a complete mirror with original size images. WP-MIRROR is robust and uses check-pointing to resume after interruption.

By default, WP-MIRROR mirrors the simple wikipedia and the simple wiktionary (Simple English means shorter sentences). The default should work 'out-of-the-box' with no user configuration. It should build in 200ks (two days), occupy 90G of disk space, be served locally by virtual hosts http://simple.wikipedia.site/ and http://simple.wiktionary.site/, and update automatically every week. The default should be suitable for anyone who learned English as a second language (ESL).

The top ten wikis are the en wikipedia, followed by the de, nl, fr, it, ru, es, sv, pl, and ja wikipedias. Because WP-MIRROR uses original size image files, these wikis are too large to fit on a laptop with a single 500G disk. When higher capacity hard disk drives reach the market, this list may be shortened. But for now, a desktop PC with ample disk space and main memory is required for any of the top ten, unless the user does not need the images (and this is configurable).

The en wikipedia is the most demanding case. It should build in 1Ms (twelve days), occupy 3T of disk space, be served locally by a virtual host `http://en.wikipedia.site/`, and update automatically every month.

This project's current Web home page is `http://www.nongnu.org/wp-mirror/`.

This document was generated on December 16, 2013.

Features

WP-MIRROR has been designed for robustness. WP-MIRROR asserts hardware and software prerequisites, skips over unparsable articles and bad file names, waits for Internet access when needed, retries interrupted downloads, uses check-pointing to resume after interruption, and offers concurrency to accelerate mirroring of the largest wikis.

WP-MIRROR issues a warning if `MySQL` is insecure (e.g. has a root account with no password), or if a caching web proxy is detected. WP-MIRROR warns if disk space may be inadequate for the default (a mirror of the simple wikipedia and simple wiktionary), and gracefully exits if disk space runs too low.

WP-MIRROR normally runs in background as a weekly `cron` job, updating the mirror whenever the Wikimedia Foundation posts new dump files.

Most features are configurable, either through command-line options, or via the configuration file `/etc/wp-mirror/local.conf`. Wikis can be added to the mirror, or dropped from it. If an user makes an hash of things, and wants to start over, just execute

```
root-shell# wp-mirror --restore-default
```

which drops all WP-MIRROR related databases and files (except image files). This option is used regularly during software development and debugging. An user who wishes also to delete the image files, may execute:

```
root-shell# cd /var/lib/mediawiki/images/
root-shell# rm --recursive --force *
```

That said, a good network citizen would try to avoid burdening the Internet by downloading the same files, over and over.

Troubleshooting

If WP-MIRROR fails or hangs, please take a look at the log files `/var/log/wp-mirror.log`. Alternatively, try running

```
root-shell# wp-mirror --debug
root-shell# tail -n 1000 /var/log/wp-mirror.log
```

The last few lines in the log file should provide a clue.

Developers should select some of the smaller wikis (e.g. the cho, xh, and zu wikipedias) to speed up the test cycle.

WP-MIRROR was written and is maintained by Dr. Kent L. Miller. Please see §1.8, Credits for a list of contributors.

Please report bugs in WP-MIRROR to <wp-mirror-list@nongnu.org>.

Contents

List of Figures

List of Tables

Chapter 1

General Information

The Wikimedia Foundation offers wikipedias in nearly 300 languages. In addition, the WMF has several other projects (e.g. wikibooks, wiktionary, etc.) for a total of around 1000 wikis.

WP-MIRROR is a free utility for mirroring any desired set of these wikis. That is, it builds a wiki farm that the user can browse locally. WP-MIRROR builds a complete mirror with original size image files.

The following list describes some sections of particular interest in this manual:

- **Installation.** If you wish to build a wiki farm, then you will need to install WP-MIRROR and a number of other utilities as well. Presently, the easiest way to manage dependencies is to install from a DEB package. See §2.2, Installing WP-MIRROR on Linux.
- **Default.** In its default configuration, WP-MIRROR mirrors the simple wikipedia and simple wiktionary (Simple English means shorter sentences). This should:
  - work 'out-of-the-box' with no user configuration,
  - build a complete mirror in 200ks (two days),
  - occupy 90G on an hard disk drive,
  - be served locally by virtual hosts `http://simple.wikipedia.site/` and `http://simple.wiktionary.site/` for web browsing, and
  - update automatically every week.

  If the default configuration is what you want, then you need read no further.
- **Configure a wiki farm.** Mirroring a set of wikis requires some configuration. This could be easy or hard depending on how large the selected wikis are. Therefor, before attempting to build your own wiki farm, please read §2.4.3, Configuration of a Wiki Farm and Chapter 3, System Planning and Configuration, as it may save you weeks of effort. Really.
- **Mirror a top ten wiki.** The ten largest wikis are judged to be too large to fit on a laptop PC equipped with a single 500G hard disk drive. The top ten wikis are: the en wikipedia (the largest at about 3T), followed by the de, nl, fr, it, ru, es, sv, pl, and ja wikipedias. When higher capacity drives reach the market, this list may be shortened. But for now, a desktop PC with ample disk space and main memory is required for any of the top ten, unless you can make do without the images (and this is configurable). See §3.3, Desktop Planning and Configuration.

1.1 About This Manual

This is the Reference Manual for WP-MIRROR, version 0.2, through version 0.6.

This manual is not intended for use with older versions of WP-MIRROR due to the many functional and other differences between WP-MIRROR 0.2 and previous versions. In particular, WP-MIRROR 0.1 was written as a proof of concept.

The Reference Manual source files are written in LaTeX format. The other formats are produced automatically from this source. For more information about LaTeX, see http://www.latex-project.org/.

This manual was orginally written by Dr. Kent L. Miller in 2012. It is maintained by Dr. Kent L. Miller.

1.2 Typographical and Syntax Conventions

This manual uses certain typographical conventions:

- `Text in this style` is used for directory names, file names, file types, database names, and table names.
- `Text in this style` is used for software utility names.
- `Text in this style` is used for options (command line and configuration file options).
- `Text in this style` indicates command line input.
- *`Text in this style`* is used for variable input for which you should substitute your own value.
- *Text in this style* is used for emphasis.
- URLs are formatted like http://www.nongnu.org/wp-mirror/.
- E-mail addresses are formatted like <wpmirrordev@gmail.com>.

Command-line interface (CLI) sessions are indicated as follows:

```
shell$ indicates a shell command
root-shell# indicates a shell command entered as root
mysql> indicates a mysql statement
```

Command-line interface sessions involving SSH name the remote host as follows:

```
shell$ ssh remote-host
remote-host$ indicates a shell command entered on the remote host
remote-host$ exit
shell$ indicates a shell command entered on the local host
```

Configuration file contents are indicated as follows:

```
config line 1
config line 2
config line 3
```

1.3 Overview of WP-MIRROR

1.3.1 What is WP-MIRROR?

- **Mirror building software**. The Wikimedia Foundation offers wikipedias in nearly 300 languages. In addition, the WMF has several other projects (e.g. wikibooks, wiktionary, etc.) for a total of around 1000 wikis. WP-MIRROR is a free utility for mirroring any desired set of these wikis. That is, it builds a wiki farm that the user can browse locally.

  Many users need such off-line access, often for reasons of mobility, availability, and privacy. They currently use kiwix which provides selected articles and thumbnail images. WP-MIRROR builds a complete mirror with original size images.
- **Robust and efficient**. WP-MIRROR is stable even in the face of: corrupt dump files, corrupt image files, incomplete downloads, Internet access interruptions, and low disk space. WP-MIRROR uses check-pointing to resume after process interruptions.

  WP-MIRROR automatically configures software dependencies, such as `apache2`, `cURL`, `MediaWiki`, and `MySQL`, to improve performance.

  If disk space runs low (below 5G), WP-MIRROR gracefully exits.

- **Easy to install, configure, and use**. WP-MIRROR offers a default installation that 'just works'. It mirrors the simple wikipedia and simple wiktionary (Simple English means shorter sentences). It should build in 200ks (two days), occupy 90G of disk space (which should fit on most laptops), be served locally by virtual hosts http://simple.wikipedia.site/ and http://simple.wiktionary.site/, and update automatically every week. The default should be suitable for anyone who learned English as a second language (ESL).

  The user may select any desired set of wikis with just one line in a configuration file.

  WP-MIRROR runs as a weekly cron job updating the mirror. Once WP-MIRROR is installed and configured, there is nothing more that the user needs to do.

  WP-MIRROR sets up virtual hosts such as http://simple.wikipedia.site/ and http://simple.wiktionary.site/, one for each wiki in the set, which the user may access with a web browser.
- **Free software**. The Free Software Foundation uses the following definition, which reads (in part):

  > A program is free software if the program's users have the four essential freedoms:
  > – The freedom to run the program, for any purpose (freedom 0).
  > – The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
  > – The freedom to redistribute copies so you can help your neighbor (freedom 2).
  > – The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.
  >
  > http://www.gnu.org/philosophy/free-sw.html, accessed 2012-12-19
- **Free documentation**. The WP-MIRROR Reference Manual (this document) has been released under the GNU Free Documentation License, version 1.3. A copy of the license is included in the section entitled GNU Free Documentation License.

1.3.2 History of WP-MIRROR

Summer of 2010 the author decided to build a mirror of http://en.wikipedia.org/. Having mirrored other sites, the author did not think it would be too challenging. The pet project, however, encountered a morass of corrupt dump files, unsupported utilities, dismal database performance, thousands of corrupt image files, stalled downloads, proxy malfunctions, system crashes, and hardware failures.

By Summer of 2011, most problems were in hand.

Up to this point, the project did not have a name, and it only built a mirror of the en wikipedia. It would later be named WP-MIRROR 0.1.

On 2011-Nov-22, the author ran into Richard Stallman of the Free Software Foundation, who encouraged the author to release the software under the GNU General Public License (GPLv3). The name WP-MIRROR was selected for the project. The code was refactored in order to mirror a set of wikipedias (i.e. build a wikipedia farm).

On 2011-Dec-25, WP-MIRROR 0.2 was released under GPLv3, and hosted by the Free Software Foundation at http://www.nongnu.org/wp-mirror/.

On 2012-Feb-12, the author submitted an intent to package (ITP) to the Debian bug tracking system BTS (bug #659640).

On 2012-Mar-04, WP-MIRROR 0.3 was released as DEB and RPM packages. During Summer of 2012, user experience revealed that: configuration of dependencies was a barrier to adoption. Therefore, automation of the dependency configuration became the main design objective for the next release.

On 2012-Nov-12, WP-MIRROR 0.4 was released as a DEB package. User feedback indicated that most were using Ubuntu. Therefore, increasing the number of distributions and platforms,

for which WP-MIRROR worked 'out-of-the-box' became the main design objective for the next release.

On 2012-Dec-14, WP-MIRROR 0.5 was released as a DEB package. It works 'out-of-the-box' on Debian 7.0 (wheezy) and Ubuntu 12.10 (quantal). WP-MIRROR 0.5 is 'feature complete' and deemed 'beta-ware'. User and WMF feedback suggested numerous improvements. Therefore, performance and reliability became the main design objectives for the next release.

On 2013-Jun-02, WP-MIRROR 0.6 was released as a DEB package. It builds the en wikipedia in 80% less time, using 75% less DRAM. WP-MIRROR 0.6 also builds wikis from other WMF projects (e.g. wikibooks, wiktionary).

1.3.3 The Main Features of WP-MIRROR

Functionality:

- **Access.** WP-MIRROR creates virtual hosts, so that the user may access mirrored wikis locally using a web browser.
- **Mirror.** WP-MIRROR can run in two modes. In mirror mode, it builds a mirror of one or more wikis, the choice of which may be set in a configuration file, or as a command-line option. Wikis can be added and dropped from the mirror dynamically.
- **Monitor.** WP-MIRROR when running in monitor mode, reports on the state of the mirror and the building process. Information may be displayed in three ways:
  - **GUI.** State information is displayed as a set of colored progress bars in a separate window, if the host has a suitable windowing system such as X.
  - **Screen.** State information is displayed as a set of ASCII art progress bars in a terminal (or console).
  - **Text.** State information is displayed in a terminal (or console) and scrolls upwards. This last display method is most often used for debugging purposes.

Efficiency:

- **Concurrency.** Multiple instances of WP-MIRROR can run concurrently. This is done to accelerate the mirroring of the largest wikis (such as the en wikipedia).
- **Dependency configuration.** An order of magnitude performance increase is possible by careful configuration of MySQL. WP-MIRROR provides sample wp-mirror.cnf files for the laptop and desktop cases. The laptop case is installed by default.
- **HTTP/1.1 persistent connections.** Large numbers of image files can be downloaded using a single HTTP connection.
- **Profiling.** WP-MIRROR keeps track of the time spent by various functions.

Robustness:

- **Assert hardware.** WP-MIRROR, when launched, first asserts adequate DRAM, adequate disk space, and Internet connectivity. This is because: a large DRAM improves MySQL performance, and ample disk space must be available for downloaded image files (e.g. simple wikipedia requires 90G, en wikipedia requires 3T, as of year 2013).
- **Assert minimum disk space.** WP-MIRROR periodically asserts minimum disk space, and gracefully exits if disk space runs low.
- **Assert software.** WP-MIRROR, when launched in mirror mode, first asserts the configuration of dependencies such as apache2, MediaWiki, MySQL, and others.
- **Automated dependency configuration.** WP-MIRROR automatically configures dependencies such as apache2, MediaWiki, MySQL, and others. For the default case (a clean install of the simple wikipedia and simple wiktionary on a laptop) everything should 'just work'.
- **Check-pointing.** WP-MIRROR may be interrupted (user closes laptop, power fails, cat walks across keyboard) and resumed later. This is important because building a mirror of one of the top ten wikis could take weeks to complete, a process that one should be loathe to repeat.

- **Retry failed tasks.** WP-MIRROR keeps track of the failed tasks. After all other tasks have run to completion, WP-MIRROR retries the failed tasks.
- **Retry interrupted downloads.** WP-MIRROR uses `wget` for file downloads. `wget` retries interrupted downloads and, if possible, continues downloading the file from the offset where the interruption occurred. This eliminates the problem of partial downloads.
- **Skip over bad file names.** Many image files have names containing control characters. Such file names pose a security risk to shell scripts (because they may insert malicious shell commands) and to `SQL` commands (`SQL` injection). Such files are not downloaded.
- **Skip over unparsable articles.** Dump files of each of wiki, are posted online by the Wikimedia Foundation. These dump files are in `XML` format, and are updated monthly, more or less. Unfortunately, about one per million articles are corrupt, and cause the `XML` parser to fail. WP-MIRROR copes by splitting each dump file into small chunks of a thousand articles each. These chunks are called `xchunk`s. The processing of each `xchunk` is forked as a separate process. So, when an unparsable article is encountered, the failure is limited to that `xchunk`.
- **Wait for Internet access.** WP-MIRROR performs a number of tasks that require Internet access (and many that do not). When WP-MIRROR encounters a task that requires Internet access, it first asserts connectivity. If the assert fails, WP-MIRROR sleeps for 10 seconds, and tries again. When Internet connectivity is restored, WP-MIRROR resumes.
- **Warnings.** WP-MIRROR issues warnings if `MySQL` is insecure (e.g. has a root account with no password), if disk space may be inadequate, or if a caching web proxy is detected.

1.4 WP-MIRROR Development History

Features, ordered by the version in which they were implemented, are summarized in the Table 1.1, WP-MIRROR Development History.

Table 1.1: History of WP-MIRROR Features

Version	Feature
0.6	Command-line options `--add`, `--drop`, `--delete`, `--dump`, and `--update` enhanced for multiple wikis Command-line option `--profile` displays performance data HDD write-caching for disks underlying `InnoDB` now configurable HTTP/1.1 persistent connections used for downloading image files Images now stored in a directory tree like that used by the WMF Image dump tarballs may be downloaded Interwiki links all moved to Navigation Bar Performance improvements (less wall clock time) in all areas Protocols for downloading dump files expanded to: `http`, `ftp`, and `rsync` Site that serves WMF dump files now configurable Reliability of dump file download from `http` sites enhanced by using `wget` Wikis from other projects (e.g. wikibooks, wiktionary, etc.) can now be mirrored Virtual hosts created for all WMF projects (e.g. wikibooks, wiktionary, etc.) `wikix` removed
0.5	Command-line option `--add` lets user add wikipedia to mirror concurrently with building Command-line options `--dump` and `--update` ease database maintenance Concurrent adding, building, and dropping of wikipedias is permitted GUI mode opens and closes windows dynamically as wikipedias are added and dropped Interwiki (interlanguage) links now appear in browser navigation sidebar Virtual host renamed to `http://simple.wpmirror.site/`

continued on next page

continued from previous page

Version	Feature
	WP-MIRROR logotype and favicon now appear in browser
0.4	Assert minimum disk space for images and InnoDB Automated default configuration for dependencies Command-line option --restore-default lets user start over Shell commands and scripts now POSIX-compliant Warn if MySQL is insecure or disk space low WP-MIRROR Reference Manual provided in LaTeX and PDF format WP-MIRROR Reference Manual released under GFDL 1.3
0.3	Build DEB from source using pbuilder clean-room Released as DEB and RPM packages Scheduling algorithm rewritten
0.2	Mirror a set of wikipedias Images processed with built-in alternative to wikix
0.1	Assert hardware and software prerequisites (at start) Check-pointing to resume after interruption Concurrency to accelerate mirroring of largest wikipedias Disable write-caching for disks underlying InnoDB Mirror a single wikipedia Monitor mode to display state of each mirror Skip over bad image file names Skip over unparsable articles Validate image files and sequester those that are corrupt Wait for Internet access when needed Warn if proxy detected

1.5 What Is New in WP-MIRROR

This section complements §1.4, WP-MIRROR Development History. For each WP-MIRROR version, this section lists the: development phase, packaging options, licensing, features added, features deprecated, and features removed.

1.5.1 What Is New in WP-MIRROR 0.6

Phase	beta
Packaging	DEB and tarball
License	GPLv3

Features added:

- **Command-line options**: The --add, --drop, --delete, --dump, and --update options take a wider range of arguments to facilitate handling of multiple wikis.
- **Command-line options**: The --profile option displays how much time each of the Finite State Machine functions are consuming. It may be run concurrently with adding, building, and dropping wikipedias from the mirror. This is used for performance analysis.
- **HDD write caching**: The degree of Durability (the 'D' in ACID) of COMMITted transactions is now configurable. To achieve the greatest possible Durability, WP-MIRROR's database transactions must be written to disk immediately after they are COMMITted. However, modest performance gains (22% less time) are seen by flushing transactions to disk once per second (default).
- **HTTP/1.1 persistent connections**: The downloading of image files now takes advantage of HTTP/1.1 persistent connections [RFC2616]. Significant performance gains (64% less time) are obtained.

- **Images**: Now stored in a directory like that of WMF. Now /var/lib/mediawiki/images/<project>/<language- was /var/lib/mediawiki/images/[0-9a-f]/.
- **Image dumps**: Image dump tarballs can now be downloaded and extracted.
- **Interwiki**: Previously at the bottom of each article there was a mass of 'red links' (inter-language links to articles in wikis not a part of the mirror). These links have now been moved to the Navigation Bar.
- **Monitor**: In --gui mode, WP-MIRROR now displays aggregate information in a single window.
- mwxml2sql: A new utility from WMF is used to convert XML dump files into SQL files which are easier to import into the database. Some performance advantage.
- **Performance**: Significant gains in all areas. Building the en wikipedia now takes 80% less time.
- **Reliability**: Downloading dump files and image files from http sites is now far more reliable. This was accomplished by using wget, which has an automatic retry feature, instead of cURL, which often leaves partial files. This obviates the need for image validation which was a CPU intensive process.
- **Sites**: Dump files can now be downloaded from alternative sites, and using a number of protocols (http, ftp, and rsync).
- **Virtual host**: Virtual hosts are now created for all WMF projects (e.g. wikibooks, wiktionary, etc.)
- **WMF projects**: The WMF has several projects besides the wikipedia (e.g. wikibooks, wiktionary, etc.) etc.). WP-MIRROR can now mirror their wikis.

Features deprecated:
Features removed:

- wikix: The C language program wikix generates shell scripts that download image files. These scripts contain "bashisms" that do not work with the Debian Almquist SHell dash. dash is a POSIX-compliant implementation of /bin/sh, which has become the default /bin/sh for Debian.
- **FSM**: Many functions in the Finite State Machine have been removed:
  - fsm-file-import replaced with fsm-file-mwxml2sql,
  - fsm-file-scrape replaced with fsm-file-list-missing,
  - fsm-file-shell replaced with fsm-file-wget,
  - fsm-image-count, fsm-image-chown, and fsm-image-rebuild removed
  - fsm-image-validate and its database table wpmirror.image removed.

1.5.2 What Is New in WP-MIRROR 0.5

Phase	beta
Packaging	DEB and tarball
License	GPLv3

Features added:

- **Command-line options**: WP-MIRROR 0.5 adds several new run-time options:

  The --add *language-code* option lets an user add a wikipedia to the mirror by providing the language-code on the command-line, rather than by editing /etc/wp-mirror/local.conf.

  The --dump *language-code* option lets an user dump the database, for a given wikipedia, to a file. The dump file is written to *xx*wiki.sql (where *xx* stands for the language-code) and stored under /var/lib/mediawiki/image/wp-mirror/. If the language-code is template, then the empty database wikidb will be dumped to database_farm.sql.

  The --info option lets the user see the values of all the parameters that can be configured in /etc/wp-mirror/local.conf.

  The --update *language-code* option lets an user update the database for a given wikipedia to the latest MediaWiki database schema. This option is useful after a major upgrade, because MediaWiki upgrades often involve changes to the database schema.

- **Concurrency**: WP-MIRROR 0.5 now permits concurrent adding, building, and dropping of wikipedias. In `--gui` mode, WP-MIRROR dynamically adjusts the number of windows as wikipedias are added and dropped from the mirror.
- **Interwiki**: WP-MIRROR 0.5 now has interlanguage links displayed in the browser Navigation Bar. These links allow the user to switch from an article in one language to the corresponding article in another.
- **Logotype**: WP-MIRROR 0.5 now has logotype and favicon displayed by web browser.
- **Virtual host**: WP-MIRROR 0.5 creates the virtual host `http://simple.wpmirror.site/` (for prior versions it was `http://simple.mediawiki.site/`). Consistency of naming was the motive. Now cron job, documentation, log files, logotype, packaging, program, and virtual host all have names like 'wp-mirror' or 'wpmirror'.

Features deprecated:

- `wikix` **not POSIX compliant**: The `C` language program `wikix` generates shell scripts that download image files. These scripts contain "bashisms" that do not work with the Debian Almquist SHell `dash`. `dash` is a POSIX-compliant implementation of `/bin/sh`, which has become the default `/bin/sh` for Debian. Using the built-in alternative to `wikix` is now recommended.

Features removed:

- **Verbose error messages**: It can happen that one or more assertions fail. This was especially true for WP-MIRROR 0.3 and earlier versions, when the configuration of dependencies was not yet automated. At that time it was thought that highly verbose error messages would be an aid to early adopters and to software development. See Chapter D, Error Messages.

  In WP-MIRROR 0.4, these error messages were relegated to the log files `/var/log/wp-mirror.log`. In WP-MIRROR 0.5, they were eliminated entirely.

  Verbose error messages were removed for two reasons: 1) Given that user configuration is no longer needed, the user interface should be far less verbose, ideally one line per checkpoint; and 2) Given the availability of the Reference Manual (this document), the log files would no longer need to be highly verbose.

1.5.3 What Is New in WP-MIRROR 0.4

Phase	alpha
Packaging	`DEB` and `tarball`
License	`GPLv3`

Features added:

- **Assert minimum disk space (periodically)**: WP-MIRROR 0.4 adds protection against low disk space. Disk space is periodically asserted. If the assert fails, WP-MIRROR exits gracefully.
- **Automated default configuration for dependencies**: Before WP-MIRROR 0.4, the user was required to do quite a bit of post-install configuration of the binary dependencies, such as `MySQL`, `MediaWiki`, and `apache2`. The advantage was that power users could obtain an order-of-magnitude performance increase. The disadvantage was that it deterred potential users who want everything to 'just work' without having to read documentation. This is a valid concern for those who learned English as a second language (ESL). WP-MIRROR 0.4 installation now provides default configuration of its dependencies.
- **Command-line options**: WP-MIRROR 0.4 adds a new run-time option `--restore-default`. This option is intended for users who have messed up their configuration (e.g. those trying to mirror the en wikipedia). Running this option let the user start over with the default installation, namely, a mirror of the simple wikipedia.

  This option is a bit dangerous and comes with a warning message because it: drops all WP-MIRROR related databases, deletes all WP-MIRROR working files, and deletes all WP-MIRROR related configuration files (including those intended for WP-MIRROR's dependencies).

- **Documentation in LaTeX and PDF format**: WP-MIRROR 0.4 includes a Reference Manual in PDF format (this document), built from LaTeX source. This new document features improved browsing using hyperrefs, and improved content and layout. Before WP-MIRROR 0.4, the documentation was available only in text format—a lengthy README, and a man page.
- **Shell commands and scripts now POSIX compliant**: WP-MIRROR gets much of the work done by forking shells to run shell commands or even large shell scripts. Some of these commands and scripts previously contained "bashisms" that did not work with the Debian Almquist SHell dash. dash is a POSIX-compliant implementation of /bin/sh, which has become the default /bin/sh for Debian.
- **Warnings**: WP-MIRROR 0.4 issues warnings if MySQL is insecure (e.g. has a root account with no password) or if disk space is inadequate. These warnings are intended as a convenience for the system administrator.

Features deprecated:

- wikix **not POSIX compliant**: The C language program wikix generates shell scripts that download image files. These scripts contain "bashisms" that do not work with the Debian Almquist SHell dash. dash is a POSIX-compliant implementation of /bin/sh, which has become the default /bin/sh for Debian. Using the built-in alternative to wikix is now recommended.

Features removed:

- **Command-line options**: Before WP-MIRROR 0.4, some command-line options existed for the purpose of generating files during the build process. For WP-MIRROR 0.4, we have:
  - **Build options removed**: --config-default, and --config-local. The files, previously generated by WP-MIRROR using these build options, are now simply included in the package.
  - **Build options retained**: --copyright, --help, and --version are still used to generate files during the build process.

1.5.4 What Is New in WP-MIRROR 0.3

Phase	alpha
Packaging	DEB, RPM, and tarball
License	GPLv3

Features added:

- **Build from source using pbuilder clean-room**: Debian imposes a number of requirements that must be satisfied before a DEB package will be considered for inclusion in any Debian distributions (such as sid). One such requirement is that the package be built from source in a 'clean room' using pbuilder. WP-MIRROR 0.3 implements a Makefile that passes this test.
- **Documentation**: README updated.
  - **Dependency**: Build dependencies and binary dependencies are listed.
  - **Installation**: Installation and post-install configuration instructions for Debian GNU/Linux 6.0 (squeeze), and tarball.
- **Scheduling**: WP-MIRROR forks a great number of tasks. While many can be run concurrently, some must be done in a particular order. Prior to WP-MIRROR 0.3, the sequencing of tasks was handled by hundreds of lines of 'spaghetti code'. This was a debugging and maintenance issue. WP-MIRROR 0.3 implements a new scheduling algorithm, using very little code, that looks up the sequence in a LISP list named *type-state-priority*. Basically, it is a Finite State Machine (FSM).

Features deprecated: none.
Features removed:

- **Command-line options**: Before WP-MIRROR 0.3, many command-line options existed for the purpose of generating files during the build process. WP-MIRROR 0.3
  - **Build options removed**: `--changelog`, `--cron`, `--changelog-Debian`, `--debian-control`, `--localsettings-wpmirror`, `--logrotate`, `mw-farm-importdump`, `--mw-farm-rebuildimages`, `--mw-farm-update`, `--thanks`, and `--virtual-host`. The files, previously generated by WP-MIRROR using these build options, are now simply included in the package.
  - **Build options retained**: `--config-default`, `--config-local`, `--copyright`, `--help`, and `--version` are still used to generate files during the build process.

1.5.5 What Is New in WP-MIRROR 0.2

Phase	pre-alpha
Packaging	`tarball`
License	`GPLv3`

Features added:

- **Images**: WP-MIRROR 0.2 adds a built-in alternative to `wikix`.
  - **PRO**: The built-in alternative generates smaller shell scripts that capture more image files and throw fewer `HTTP` 400 and 404 errors than `wikix`.
  - **CON**: The build-in alternative takes longer to run than `wikix`, and `wikix` does download some image files that the alternative does not.
- **Mirror farm**: WP-MIRROR 0.2 adds the capability to mirror a set of wikipedias (e.g. `meta`, `simple`, `zu`). The choice of wikipedias may be set in a configuration file, and otherwise defaults to the simple wikipedia.

Features deprecated: none.
Features removed: none.

1.5.6 What Is New in WP-MIRROR 0.1

Phase	proof of concept
Packaging	`tarball`
License	was never released

Features added:

- **Assert hardware and software prerequisites (at start)**: WP-MIRROR asserts hardware prerequisites and software prerequisites. Hardware prerequisites include: adequate DRAM, adequate disk space, and Internet connectivity. Software prerequisites include dependencies, such as `MySQL`, `MediaWiki`, and `apache2`, as well as their configuration.
- **Check-pointing**: WP-MIRROR stores its state information in an ACID compliant database. This state information is read at each point where WP-MIRROR must decide what task to perform next. This state information is Durable (the 'D' in ACID), which permits WP-MIRROR to resume after interruption.
- **Concurrency**: WP-MIRROR stores its state information in an ACID compliant database. This state information is read at each point where WP-MIRROR must decide what task to perform next. Multiple instances of WP-MIRROR accessing the state information are Isolated (the 'I' in ACID) from each other, which permits concurrency. Concurrency is used to accelerate the mirroring of the largest wikipedias (such as the en wikipedia).
- **Mirror**: WP-MIRROR can run in two modes. In mirror mode, it builds a mirror of a single wikipedia, the choice of which may be set in a configuration file.
- **Monitor**: WP-MIRROR can run in two modes. In monitor mode, it reports on the state of the mirror building process and the mirror. Information may be displayed in three ways:
  - **GUI**: State information is displayed as a set of colored progress bars in a separate window, if the host has a suitable windowing system such as `X`.
  - **Screen**: State information is displayed as a set of ASCII art progress bars in a terminal (or console).

  - **Text**: State information is displayed in a terminal (or console) and scrolls upwards. This last display method is most often used for debugging purposes.
- **Retry failed tasks**: WP-MIRROR keeps track of the failed tasks (an example of the state information kept by WP-MIRROR in an ACID compliant database). After all other tasks have run to completion, WP-MIRROR retries the failed tasks.
- **Skip over bad file names**: Many image files have names containing control characters. Such file names pose a security risk to shell scripts (because they may insert malicious shell commands) and to SQL commands (SQL injection). Such files are not downloaded.
- **Skip over unparsable articles**: Dump files of each of wikipedia, are posted online by the Wikimedia Foundation. These dump files are in XML format, and are updated monthly, more or less. Unfortunately, about one per million articles are corrupt, and cause the XML parser to fail. WP-MIRROR copes by splitting each dump file into small chunks of a thousand articles each. These chunks are called xchunks. The processing of each xchunk is forked as a separate process. So, when an unparsable article is encountered, the failure is limited to that xchunk.
- **Validate image files**: Many images files fail to download completely, and some are corrupt to begin with. If your Internet access passes through a web-caching proxy (such as polipo), a great number of bad image files turn out to be error messages written by the proxy. So, WP-MIRROR validates every downloaded image file, and sequesters the corrupt ones into a bad-images directory where the user may later inspect them.
- **Wait for Internet access**: WP-MIRROR performs a number of tasks that require Internet access (and many that do not). When WP-MIRROR encounters a task that requires Internet access, it first asserts connectivity. If the assert fails, WP-MIRROR sleeps for 10 seconds, and tries again. When Internet connectivity is restored, WP-MIRROR resumes.
- **Warn if Proxy Detected**: WP-MIRROR examines the configuration files of bash, cURL, and wget for evidence of a proxy. If such evidence is found, a warning is issued.

Features deprecated: none.
Features removed: none.

1.6 WP-MIRROR Information Sources

1.6.1 Project Home Page for WP-MIRROR

The WP-MIRROR project home page can be found at http://www.nongnu.org/wp-mirror/.

1.6.2 Downloading WP-MIRROR

WP-MIRROR packages can be found online on the main GNU server at http://download.savannah.gnu.org/releases/wp-mirror/.

1.6.3 Documentation for WP-MIRROR

Documentation for WP-MIRROR can be found online on the main GNU server at http://download.savannah.gnu.org/releases/wp-mirror/manual/.

1.6.4 Mailing Lists for WP-MIRROR

WP-MIRROR has several mailing lists to which you may subscribe or unsubscribe.
Definitions:

upstream, a., of or pertaining to the code, documentation, and features of WP-MIRROR.
downstream, a., of or pertaining to the packaging of WP-MIRROR.

WP-MIRROR has the following upstream mailing lists:

- https://lists.gnu.org/mailman/listinfo/wp-mirror-announce/ is used to announce releases.

- https://lists.gnu.org/mailman/listinfo/wp-mirror-devel/ is a closed list for developers and testers.
- https://lists.gnu.org/mailman/listinfo/wp-mirror-list/ is used to discuss most aspects of WP-MIRROR, including development and enhancement requests, as well as bug reports.

WP-MIRROR has the following downstream mailing lists:

- TBD

The author may be contacted directly using <wpmirrordev@gmail.com>.

1.7 How to Report Bugs or Problems

Before posting a bug report about a problem, please try to verify that it is a bug and that it has not been reported already:

- Start by reading the online documentation at http://download.savannah.gnu.org/releases/wp-mirror/manua
  In particular, please take a look at §1.5, What Is New in WP-MIRROR, because your problem may have been solved in a recent version.

Upstream bugs and feature requests may be reported to https://lists.gnu.org/mailman/listinfo/wp-mirror-list
Downstream bugs may be reported as follows:

- **Debian.** Bugs should be reported using the reportbug program (see http://www.debian.org/Bugs/Reporting for documentation).
- **RPM.** TBD
- **tarball.** Bugs should be reported to https://lists.gnu.org/mailman/listinfo/wp-mirror-list/.

1.8 Credits

1.8.1 Contributors to WP-MIRROR

The following people have helped with WP-MIRROR code and/or features.

- **Guy Castagnoli**. Testing HTTP redirect. Submitting bug reports.
- **Jason Cooper**. Testing on virtual machine. Submitting bug report and patch.
- **Benjamin Goldsmith.** Testing on Debian GNU/Linux 6.0 (squeeze). Submitting bug reports.
- **Tylery Khai.** Recommending color scheme for monitor mode (--gui option).
- **Jason Skomorowski.** Testing on Ubuntu 12.10 (quantal). Submitting bug reports. Recommending performance enhancements.

The following people have helped with access to wikimedia dump files.

- **Christian Aistleitner.** Advising on dump file access.
- **Jeremy Baron.** Advising on dump file access. Advising on git repository access.
- **Kevin Day.** Improving accessibility of http://ftpmirror.your.org/. Providing rsync access to image dump tarballs. Fixing HTTP redirect issues with IPv4 and IPv6.
- **Ariel Glenn.** Providing dump files. Advising on dump file access. Writing and maintaining mwxml2sql. Advising on git repository access.
- **Platonides.** Advising on image file validation.

1.8.2 Documenters and Translators

The following people have helped with WP-MIRROR documentation.

- **Tylery Khai.** Proofreading early drafts of this manual.

Chapter 2

Installing and Upgrading WP-MIRROR

2.1 General Installation Guidance

This section describes how to choose, download, and verify your WP-MIRROR package. Subsequent sections describe how to install your choice of WP-MIRROR package on different platforms.

2.1.1 Operating Systems Supported by WP-MIRROR

This section lists the operating systems on which WP-MIRROR is known to run.

Table 2.1: History of WP-MIRROR Support for Operating Systems

WP-MIRROR Version	Operating System	Architecture			
		amd64	armel	i386	ia64
0.6	Linux	3.2.0			
0.5	Linux	3.2.0			
0.4	Linux	3.2.0			
0.3	Linux	2.6.32			

WP-MIRROR has been reported to build successfully on the above operating systems. See Table 2.1, History of WP-MIRROR Support for Operating Systems.

2.1.2 GNU/Linux Distributions Supported by WP-MIRROR

This section lists the GNU/Linux distributions on which WP-MIRROR is known to run.

Table 2.2: History of WP-MIRROR Support for GNU/Linux Distributions

WP-MIRROR Version	Debian		Ubuntu	
	6.0	7.0	12.10	Raring
0.6		Y	Y	
0.5		Y	Y	
0.4		Y		
0.3	Y			

WP-MIRROR has been reported to install and run successfully on the above GNU/Linux distributions. See Table 2.2, History of WP-MIRROR Support for GNU/Linux Distributions.

2.1.3 How to Get WP-MIRROR

Check the home page at http://www.nongnu.org/wp-mirror/ for information about the current version of WP-MIRROR. The home page has a link to the downloads page at

http://download.savannah.gnu.org/releases/wp-mirror/ where one may find recent releases in several file formats: tarball .tar.gz file, DEB package, and RPM package.

2.1.4 Choosing Which WP-MIRROR Distribution to Install

You have three decisions to make: which version, which package format, and when to upgrade.

2.1.4.1 Choosing Which Version of WP-MIRROR to Install

Choosing the most recent version is recommended, as it offers more features and bug fixes.

2.1.4.2 Choosing a Distribution Format

After choosing which version to install, you should decide whether to install a binary distribution (e.g. DEB, RPM) or a source distribution (e.g. .tar.gz).

Binary packages are the easiest to install. You may prefer binary distributions if:

- If you need dependency issues taken care of for you.
- If you have multiple platforms and, to reduce maintenance cost, you have decided that all platforms should run the same major distribution, such as Debian GNU/Linux or RedHat.
- If you want the default configuration which 'just works', without having to read much documentation.

Source distributions are more challenging to install. You may want the source if:

- You want to configure things yourself (e.g. if you need different install locations).
- You want to port WP-MIRROR to a new platform.

2.1.4.3 Choosing When to Upgrade

If you are using a major distribution, such as Debian GNU/Linux or RedHat, then it is best to upgrade your WP-MIRROR distribution at the same time you upgrade your major distribution (e.g. from Debian GNU/Linux 6.0 (squeeze) to Debian GNU/Linux 7.0 (wheezy)). This is because upgrading a major distribution entails upgrading clisp, MediaWiki, MySQL, and many other dependencies in a consistent fashion. Some of these upgrades may be incompatible. In particular, MediaWiki upgrades usually involve changes to its database schema, the historical record of which is shown at http://www.mediawiki.org/wiki/Manual:Database_layout.

2.1.5 Verifying Package Integrity Using Checksums or GnuPG

After downloading a WP-MIRROR package and before installing it, you should verify its integrity. This is to protect you agains partial downloads and alteration. All WP-MIRROR packages are cryptographically signed using GnuPG, the GNU Privacy Guard.

2.1.5.1 How to Get the Author's Public Build Key

All tarballs and documentation are signed using GnuPG. DEB packages are signed as well (see §2.1.5.2, Checksums for a DEB Package). See http://www.gnupg.org/ and http://www.opengpg.org/ for documentation.

To verify a signed package, you must first obtain a copy of the author's public gpg build key. Public keys are available from key servers such as http://pgpkeys.mit.edu/. The desired key can be found by searching by KeyID 382FBD0C or by User ID WP-MIRROR.

First, determine if you already have the key by executing:

```
root-shell# gpg --list-keys 382FBD0C
pub   2048R/382FBD0C 2011-12-25
uid                  WP-MIRROR <wpmirrordev@gmail.com>
sub   2048R/E5A8E0CB 2011-12-25
```

If you get the above, then you have the key, and may proceed to §2.1.5.2, Checksums for a DEB Package. On the other hand, if you get the following:

```
root-shell# gpg --list-keys 382FBD0C
gpg: error reading key: public key not found
```

then you do not have the key, and must import it into gpg.

There are two ways of obtaining the public key:

2.1.5.1.1 Download the Public Key from Public Keyserver First, confirm that gpg is configured with a known public keyserver:

```
root-shell# cd /root/.gnupg/
root-shell# cat gpg.conf | grep ^keyserver
keyserver hkp://pgp.mit.edu:11371
```

Second, download the key directly from a public keyserver using the public KeyID 382FBD0C:

```
root-shell# gpg --recv-keys 382FBD0C
gpg: requesting key 382FBD0C from hkp server pgp.mit.edu
gpg: key 382FBD0C: "WP-MIRROR <wpmirrordev@gmail.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
gpg: no ultimately trusted keys found
```

Finally, since the above steps were executed under the root account, you may wish to log into your personal shell account and repeat the process (and this is recommended).

2.1.5.1.2 Cut and Paste the Public Key from Public Keyserver Alternatively, you may try the following 'cut-and-paste' method.

First, find the desired key by searching public key server, such as http://pgpkeys.mit.edu/, by KeyID 382FBD0C or by User ID WP-MIRROR.

Second, cut and paste the resulting PGP PUBLIC KEY BLOCK into a file named wpmirror_pubkey.asc. Visually confirm that it is the following:

```
root-shell# cat wpmirror_pubkey.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: SKS 1.1.0

mQENBE72h9MBCADqNJkAJs1jj9OxqM6m7iTk211V7g2U+PiLujnEmvg/BokAtoFzLHIgBYw8
oemQGfp0Aq0zYirQKnzBFTL/Ba0SNv/TVGHXgK349GzJZl/MY8rf9hvoSoNSV7RIGEBoU5a6
Bv47k4CYCDCk++jAifTpk3yAgyW7JOZ3uzwEYrvIdEW/4JZuli4ZMmZIja+o6Q4lbXylwzoq
KKBtLWcofdpNKoS+/3Mq1wENN+d3c8SjqH8NL4tMCyF4LXC0CpSxfxUiMY1d8ifl+BjpguDl
jod+UqKEKigXeP7kiwdE09t0lRLFYLwBcRuw+qbfUXi0o6tTnQPiEO6xqr3f6HywnTxrABEB
AAG0IVdQLU1JUlJPUiA8d3BtaXJyb3JkZXZAZ21haWwuY29tPokBOAQTAQIAIgUCTvaH0wIb
AwYLCQgHAwIGFQgCCQoLBBYCAwECHgECF4AACgkQMgr8nTgvvQyK5QgAv62RuSJ/VB8V5ALj
1+J3Q4CRQjmWNncUbc7xvSJPbRjl3CeZSd7MflEjevA10japTGwrUboPeOyYPupxin2I4jMY
rWHTIfqtgh03YkxYaoK2MTrNKbYUBmzM5I9jQ49aNYfM3ikhaHMUIUFFzTV6nRmekEo1L2o2
wlVWmw2j/naF8KaHXL3X+dnNBZDz6kuBTo3MI3oIR5mRuVhj9ppbS6qYF3pmVmS2agj2186D
thSrD4JBBUhzqPriPS43JPEs1L696LcNgBOQq7O964ZiNrTyu/FBIkGGkl6Oly4zALApeA8V
XmaYMwnJQwSVtt8WLD250S/Trb+C05WQfbvMybkBDQRO9ofTAQgA6n4xWXt52PfWEYsXPuDb
La57M1KiyFPKHrViFR5ic9xNIy4H5P3Iryp1plKMBkUCI6TdKNXcMYR1X4tugYtyq/LyxYgt
82f2eJdVFq3wbFPt/eMOtMn05n+K+4J8ptu+qkwyrlVkAaPofbtQGlZwb9wQmrNMZdJg6i9B
7vPvqjGaZARtrn9Gcqu8ytt/OMc/Pc9Y14iuCpkL4QeBMhAuKuBBOAqGGyCovIP7wlRmvGO5
ofKmtY1zATqGXknZpo0HSbaVcn3GxkrUNpcE2SZDFUOC82EJ7So3tzCCIRJvJUh0YE+QP1IY
/XYV/uwpcxAHma36E9u/a+o3eIXft8AVxwARAQABiQEfBBgBAgAJBQJO9ofTAhsMAAoJEDIK
/J04L70M81UIAJVHyeWQvU6EAZ62twjmNBfcwnoOsL/VHrTsUe0oY+CjZj6p2EsGQio8wC97
wwol7fc9/8UUcqdgVxIFY3OuyHsSjbSYqQ58o7e0P6hulBPc/zfw2jhpX3J4kTkyX7CgPWvd
C+WHHnmzdyBPuv+sPhsJTziVb/gr+Jzti3v85a+YtYwiGXV/7o2R373UlJFLl58veV4gF5N8
IMrXP1U+CKTXvwS9qQjULy7yRfrKiBiJ70kNkA2m+T79NXNbnUB4GHw8tS7T7bnA0c/E0Ajs
6JQCCEF3xSPS3Q8aEp4mxDUBZe7FWTlJJbvFlAV7PGmcMLZLj64j6zrbSIkF+kxbtEI=
=wNKp
-----END PGP PUBLIC KEY BLOCK-----
```

Third, import the public key into your GPG keyring by executing:

```
root-shell# gpg --import wpmirror_pubkey.asc
gpg: key 382FBD0C: "WP-MIRROR <wpmirrordev@gmail.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
gpg: no ultimately trusted keys found
```

Finally, since the above steps were executed under the root account, you may wish to log into your personal shell account and repeat the process (and this is recommended).

2.1.5.2 Checksums for a DEB Package

Checksums for a DEB package are stored in a separate .changes file which is cryptographically signed with the author's secret key. Both files may be downloaded as described in §2.1.3, How to Get WP-MIRROR.

First, verify the .changes file by executing:

```
shell$ gpg --verify wp-mirror_0.5-1_amd64.changes
gpg: Signature made Fri 14 Dec 2012 05:41:50 AM EST using RSA key ID 382FBD0C
gpg: Good signature from "WP-MIRROR <wpmirrordev@gmail.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 9F55 7EEA 08B6 A87E 3070  FAD4 320A FC9D 382F BD0C
```

The Good signature message indicates two things: 1) integrity, the .changes file was downloaded successfully (i.e. without transmission errors), and 2) authenticity, the file was signed by someone with access to the author's secret key (only the author, one may hope).

Second, the .changes file contains three types of checksum: sha1sum, sha256sum, and md5sum. To verify the .deb file execute:

```
shell$ sha1sum wp-mirror_0.5-1_all.deb
1c313fe66de0d74e4e6f74b6d0fa6a6efb0d68eb  wp-mirror_0.5-1_all.deb
shell$ cat wp-mirror_0.5-1_amd64.changes | grep all | grep deb | head -n 1
 1c313fe66de0d74e4e6f74b6d0fa6a6efb0d68eb 3296430 wp-mirror_0.5-1_all.deb
```

and then visually confirm that the sha1sum checksums match. A match indicates two things: 1) integrity, that the DEB package was downloaded successfully (i.e. without transmission error), and 2) authenticity, since the checksums were provided by a .changes file for which integrity and authenticity have been shown.

2.1.5.3 Checksums for a RPM Package

Checksums for a RPM package are conveniently included within the package. The RPM package may be downloaded as described in §2.1.3, How to Get WP-MIRROR.

To verify the RMP file execute:

```
shell$ rpm --checksig wp-mirror-0.3-2.noarch.rpm
wp-mirror-0.3-2.noarch.rpm: sha1 md5 OK
```

The OK message indicates one thing: integrity, that the RPM package was downloaded successfully (i.e. without transmission error).

2.1.5.4 Signature Checking for a Tarball

After downloading and importing the public build key, download the latest tarball .tar.gz file and its associated signature .sig file. Both files may be downloaded as described in §2.1.3, How to Get WP-MIRROR. Be sure to put both files into the same directory, then execute:

```
shell$ gpg --verify wp-mirror-0.5.tar.gz.sig
gpg: Signature made Fri 14 Dec 2012 05:50:18 AM EST using RSA key ID 382FBD0C
gpg: Good signature from "WP-MIRROR <wpmirrordev@gmail.com>"
```

The Good signature message indicates two things: 1) integrity, that the tarball was downloaded successfully (i.e. without transmission errors), and 2) authenticity, that the tarball was signed by someone with access to the author's secret key (only the author, one may hope).

2.2 Installing WP-MIRROR on Linux

Table 2.3: History of WP-MIRROR Packages

Version	Package	Distributions
0.6	DEB	Debian GNU/Linux 7.0 (wheezy) Ubuntu 12.10 (quantal)
0.5	DEB	Debian GNU/Linux 7.0 (wheezy) Ubuntu 12.10 (quantal)
0.4	DEB	Debian GNU/Linux 7.0 (wheezy)
0.3	DEB	Debian GNU/Linux 6.0 (squeeze)

2.2.1 Installing from a DEB Package

A DEB package includes a debian/control file that lists all of the build and binary dependencies. Naturally, the build and installation processes will be easiest when you have a platform using a Linux distribution containing the dependencies listed. WP-MIRROR 0.5 and later versions were packaged for use with the Debian GNU/Linux 7.0 (wheezy) distribution and Ubuntu 12.10 (quantal) distributions.

First, download the latest DEB packages for mediawiki-mwxml2sql and wp-mirror as described above in §2.1.3, How to Get WP-MIRROR.

Second, install the DEB packages. Beginning with version 0.6, this is done by executing:

```
root-shell# dpkg --install mediawiki-mwxml2sql_0.0.2-1_amd64.deb
root-shell# dpkg --install wp-mirror_0.6-1_all.deb
```

For WP-MIRROR 0.4 and 0.5, this is done by executing:

```
root-shell# dpkg --install wp-mirror_0.6-1_all.deb
```

For WP-MIRROR 0.3, install the DEB package by executing:

```
root-shell# dpkg --install --force-overwrite wp-mirror_0.3-2_all.deb
root-shell# cp /etc/wp-mirror/local.conf.template /etc/wp-mirror/local.conf
```

where the --force-overwrite option is required because the WP-MIRROR 0.3 package overwrites a file that belongs to another package (the mediawiki_1.15.5-2squeeze4_all.deb package). Specifically, WP-MIRROR 0.3 provides a patched version of /usr/share/mediawiki/includes/Import.php.

For WP-MIRROR 0.4 and later versions, configuration of WP-MIRROR and its dependencies is entirely automated. Just execute:

```
root-shell# wp-mirror --mirror
```

Then, to monitor the build process open another terminal and execute:

```
root-shell# wp-mirror --gui &
```

For WP-MIRROR 0.3, post-installation configuration is required. See §2.4, Postinstallation Configuration and Testing.

2.2.2 Installing from a RPM Package

TBD

For WP-MIRROR 0.4 and later versions, configuration of WP-MIRROR and its dependencies is entirely automated. Just execute:

```
root-shell# wp-mirror --mirror
```

Then, to monitor the build process open another terminal and execute:

```
root-shell# wp-mirror --gui &
```

For WP-MIRROR 0.3, post-installation configuration is required. See §2.4, Postinstallation Configuration and Testing.

2.3 Installing WP-MIRROR from Source

This approach is not for the faint of heart. Please consider installing from a DEB or RPM package.

There are two kinds of dependencies that you will have to install and configure:

- **build dependencies** are software utilities required to build WP-MIRROR from source; and
- **binary dependencies** are software utilities required to install and run WP-MIRROR.

2.3.1 Installing Dependencies

Building WP-MIRROR from source code has the advantage of letting you customize everything. The disadvantage, of course, is that you must first install and configure its software dependencies.

2.3.1.1 Installing Build Dependencies

For WP-MIRROR 0.4 and later versions, the following packages must be installed before you can build.

```
Build-Depends: clisp (>= 1:2.48-3), common-lisp-controller (>= 7.9),
 cl-getopt (>= 1.2.0), cl-md5 (>= 1:1.8.5), clisp-module-clx (>= 1:2.49-8.1),
 debhelper (>= 7.0.50~), help2man (>= 1.38.2)
```

The above dependency information is copied from the debian/control file in the DEB package.

2.3.1.2 Verifying clisp Version

Table 2.4: History of clisp Version

Version	Configuration
≥ 0.1	clisp 2.48

Check that you have clisp 2.48 or higher.

```
shell$ clisp --version | head -n 1
GNU CLISP 2.48 (2009-07-28) (built 3487543663) (memory 3534965158)
```

Earlier distributions, such as Debian GNU/Linux 5.0 (lenny), and its derivatives, such as Ubuntu 10.04 LTS (lucid), provide older versions of clisp that lack some of the functions called by WP-MIRROR.

2.3.1.3 Configuring common-lisp-controller

Table 2.5: History of common-lisp-controller Configuration

Version	Configuration
≥ 0.4	no user configuration is necessary
≤ 0.3	N/A

All modern language systems come with libraries that provide functionality greatly in excess of that needed for standards compliance. Often these libraries are provided by third parties. Common Lisp systems are no exception.

For Debian distributions, third-party libraries for Common Lisp are installed under /usr/share/common-lisp/source/, and symbolic links to these source files are collected under /usr/share/common-lisp/systems/. The common-lisp-controller makes use of cl-asdf (see below).

For WP-MIRROR 0.4 and later versions, common-lisp-controller is used to manage libraries. No user configuration of common-lisp-controller is necessary.

2.3.1.4 Configuring cl-asdf

Table 2.6: History of cl-asdf Configuration

Version	Configuration
≥ 0.4	no user configuration is necessary
≤ 0.3	user configuration required (see below)

Another System Definition Facility (cl-asdf), is the link between a Common Lisp system and any third-party libraries that it calls. cl-asdf is not immediately usable upon installation. Your Common Lisp system (clisp in this case) must first be made aware of its location. cl-asdf comes with documentation that discusses configuration.

```
shell$ less /usr/share/doc/cl-asdf/README.Debian
shell$ lynx /usr/share/doc/cl-asdf/asdf/index.html
```

For WP-MIRROR 0.4 and later versions, common-lisp-controller manages cl-asdf. No user configuration of cl-asdf is necessary.

For WP-MIRROR 0.3 and earlier versions, it is necessary for the user to configure cl-asdf.

But, before configuring, first note that WP-MIRROR will be run as root, so the configuration file .clisprc, must be put in the root directory, rather than the user's home directory. To configure clisp to use cl-asdf, append the following line to /root/.clisprc.

```
(load #P"/usr/share/common-lisp/source/cl-asdf/asdf.lisp")
(push #P"/usr/share/common-lisp/systems/" asdf:*central-registry*)
```

clisp should now be ready to use asdf. Check this by running

```
shell$ clisp -q
[1]>*features*
(:ASDF2 :ASDF ...
[2]>(asdf:asdf-version)
"2.011"
[3]>asdf:*central-registry*
(#P"/usr/share/common-lisp/systems/")
```

2.3.1.5 Installing Binary Dependencies

The following packages must be installed before you can run WP-MIRROR 0.6 to build a mirror.

```
Depends:
 apache2 (>= 2.2.16), bzip2 (>= 1.0.5), cjk-latex (>= 4.8.3+git20120621-1),
 cl-getopt (>= 1.2.0), cl-md5 (>= 1:1.8.5), clisp (>= 1:2.48-3),
 clisp-module-clx (>= 1:2.49-8.1), common-lisp-controller (>= 7.9),
 coreutils (>= 8.5), curl (>= 7.21.0),
 graphicsmagick (>= 1.3.12), gv (>= 1:3.7.1),
 hdparm (>= 9.32), inkscape (>= 0.48.3.1),
 librsvg2-bin (>= 2.26.3), mediawiki (>= 1:1.19.1),
 mediawiki-extensions (>= 2.6+wheezy1),
 mediawiki-extensions-math (>= 2:1.0+git20120528-5),
 mediawiki-mwxml2sql (>=0.0.2),
 mysql-client (>= 5.5.24+dfsg-9), mysql-server (>= 5.5.24+dfsg-9),
 openssl (>= 0.9.8o), rsync (>=3.0.9), tar (>= 1.26),
 texlive-latex-base (>= 2009-11), tidy (>= 20091223cvs), tzdata (>= 2012),
 wget (>= 1.12)
```

The above dependency information is copied from the `debian/control` file in the `DEB` package.

2.3.2 Installing WP-MIRROR from a Standard Source Distribution

For WP-MIRROR 0.4 and later versions, installation from a standard source distribution is done by executing:

```
shell$ tar --extract --gzip --preserve-permissions --file wp-mirror-0.4.tar.gz
shell$ cd wp-mirror-0.4
shell$ make build
root-shell# make install
```

For WP-MIRROR 0.3 and earlier versions, also execute:

```
root-shell# cp /etc/wp-mirror/local.conf.template /etc/wp-mirror/local.conf
```

For WP-MIRROR 0.3, post-installation configuration is required. See §2.4, Postinstallation Configuration and Testing.

2.3.3 WP-MIRROR Build Options

WP-MIRROR has command line options that may be invoked to generate files during the build and install process. These command line options are listed in Table 2.7, WP-MIRROR Build Options Reference.

Table 2.7: WP-MIRROR Build Options Reference

Build Options	Description	Intr	Rem
`--copyright`	generate `copyright`	0.1	
`--help`	used by `help2man` to generate `wp-mirror.1.gz`	0.1	
`--version`	used by `help2man` to generate `wp-mirror.1.gz`	0.1	

Many build options have been removed from recent versions. The files, previously generated by WP-MIRROR using these build options, are now simply included in the package. These obsolete command line options are listed in Table 2.8, WP-MIRROR Build Options Reference (Obsolete).

Additionally, WP-MIRROR has several run-time options for use in mirror mode, which are listed in Table 4.1, WP-MIRROR Mirror Mode Options Reference, and several more options for use in monitor mode, which are listed in Table 4.3, WP-MIRROR Monitor Mode Options Reference.

Table 2.8: WP-MIRROR Build Options Reference (Obsolete)

Build Options	Description	Intr	Rem
--changelog	generate changelog	0.1	0.3
--config-default	generate default.conf	0.1	0.4
--config-local	generate local.conf	0.1	0.4
--cron	generate cron.d/wp-mirror	0.1	0.3
--changelog-debian	generate debian/changelog	0.1	0.3
--debian-control	generate debian/control	0.1	0.3
--localsettings-wpmirror	generate LocalSettings_wpmirror.php	0.2	0.3
--logrotate	generate logrotate.d/wp-mirror	0.1	0.3
--mw-farm-importdump	generate importDump_farm.php	0.2	0.3
--mw-farm -rebuildimages	generate rebuildImages_farm.php	0.2	0.3
--mw-farm-update	generate update_farm.php	0.2	0.3
--mw-import	download Import.php from SVN	0.2	0.4
--thanks	generate thanks	0.1	0.3
--virtual-host	generate mediawiki.site.conf	0.2	0.3

2.4 Postinstallation Configuration and Testing

After installation, we note that WP-MIRROR relies on a great number of binary dependencies, such as apache2, MediaWiki, MySQL, and others. These dependencies must now be configured.

There are three cases:

- **Default configuration** is completely automated. See §2.4.1, Default Configuration.
- **Farm configuration** requires editing one line in a configuration file. See §2.4.3, Configuration of a Wiki Farm.
- **Top ten wikipedia configuration** requires quite a bit of system planning and configuration. See Chapter 3, System Planning and Configuration.

2.4.1 Default Configuration

For WP-MIRROR 0.4 and later versions, the default configuration is entirely automated. The user should execute:

```
root-shell# wp-mirror --mirror
```

and watch the stream of messages. These messages, which identify each step during the run, are explained in §5.1, How --mirror Works. If any message indicates a warning or a failure, please read that section.

It is convenient to monitor the mirror by opening a second terminal and executing one of the following:

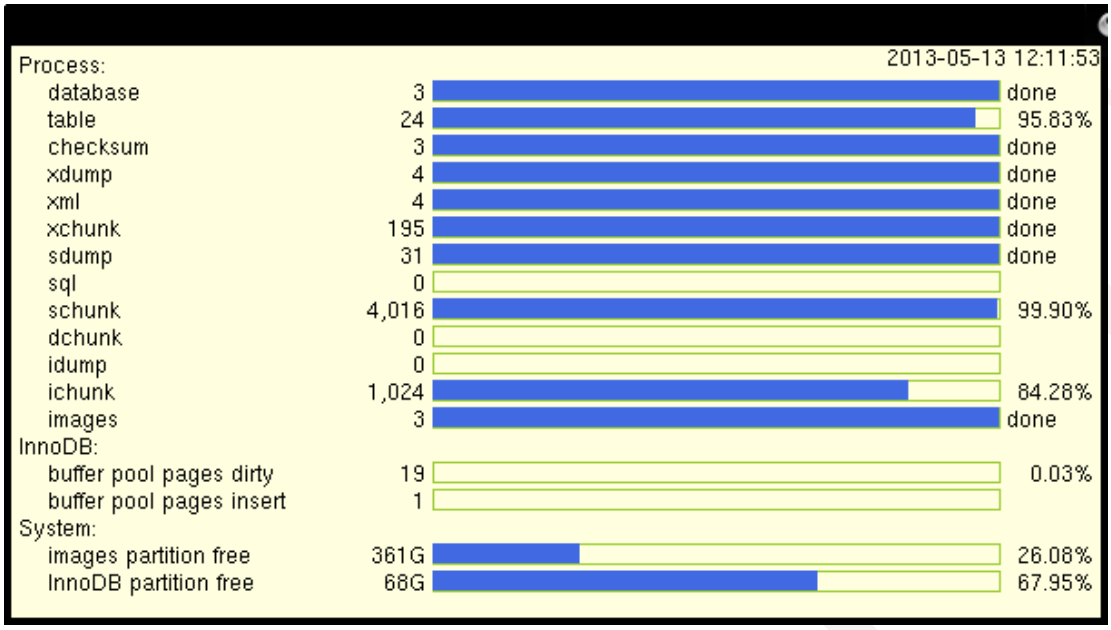
```
root-shell# wp-mirror --gui &
root-shell# wp-mirror --screen
root-shell# wp-mirror --text
```

2.4.2 Working with X

WP-MIRROR is an X client, so an user sitting at an X server should try the --gui option. A screen shot of the GUI display is shown in Figure 2.1, WP-MIRROR Monitor Mode in X.

Note that ssh can be configured to transport the X protocol, so that the X client and X server can be on separate hosts. This is easily done. On each host, edit the ssh and sshd configuration files to read:

Figure 2.1: WP-MIRROR Monitor Mode in X




```
root-shell# cat /etc/ssh/ssh_config | grep X
   ForwardX11 yes
   #ForwardX11Trusted yes
```

and

```
root-shell# cat /etc/ssh/sshd_config | grep X
X11Forwarding yes
X11DisplayOffset 10
```

If the user does not have an X server, then the `--screen` or `--text` options should be adequate.

2.4.3 Configuration of a Wiki Farm

To mirror a set of wikis, it is necessary to edit one line in `/etc/wp-mirror/local.conf`.

First, visit

http://meta.wikimedia.org/wiki/List_of_Wikipedias, and

http://meta.wikimedia.org/wiki/Table_of_Wikimedia_projects

to see which wikis are available.

Second, decide which wikis you would like to mirror, and look up their language codes.

Third, edit the configuration file `/etc/wp-mirror/local.conf` by uncommenting and editing the `*mirror-language-code-list*` parameter. Several examples are provided in the comments:

```
root-shell# cat /etc/wp-mirror/local.conf
...
;;;    This is the default.  Expect 200ks (two days) build.
;;(defparameter *mirror-language-code-list* '("simple"))
;;;
;;;    For a quick test.  Expect 5ks (one hour) build.
;;(defparameter *mirror-language-code-list* '("xh" "zu"))
;;;
;;;    Major third-world languages.
;;(defparameter *mirror-language-code-list* '("ar" "simple"))        ;arabic
;;(defparameter *mirror-language-code-list* '("hi" "simple"))        ;hindi
;;(defparameter *mirror-language-code-list* '("vi" "simple"))        ;vietnamese
;;(defparameter *mirror-language-code-list* '("zh" "simple"))        ;chinese
;;;
;;;    Classical languages.
;;(defparameter *mirror-language-code-list* '("el" "la" "simple"))   ;greek,latin
;;(defparameter *mirror-language-code-list* '("he" "yi" "simple"))   ;hebrew,yiddish
;;(defparameter *mirror-language-code-list* '("zh-classical" "simple")) ;chinese
...
```

Fourth, consider also uncommenting and editing the `*mirror-project-list*` parameter.

```
root-shell# cat /etc/wp-mirror/local.conf
...
;;;    Project.  Default is 'wikipedia' and 'wiktionary'.  If you want all
;;;    the projects, uncomment the following:
;;;(defparameter *mirror-project-list*
;;;  '("wikibooks"  "wikimedia"   "wikinews"   "wikipedia"  "wikiquote"
;;;    "wikisource" "wikiversity" "wikivoyage" "wiktionary"
...
```

Finally, launch the mirror building process, and monitor its progress as above in §2.4.1, Default Configuration.

Chapter 3

System Planning and Configuration

The Wikimedia Foundation offers wikipedias in nearly 300 languages. However, if you intend to mirror any of the ten largest wikipedias, namely, the en wikipedia (which is the most demanding case), followed by the de, nl, fr, it, ru, es, sv, pl, and ja wikipedias (as of year 2013), then reading this chapter is required. Why?

- **Size.** Currently (year 2013), the top ten wikipedias are judged to be too large to fit on a laptop PC equipped with a single 500G hard disk drive (HDD). A desktop PC with extra hard disk drives (HDD) and memory (DRAM) is required, unless you can make do without the images (and this is configurable).
- **Performance.** An order of magnitude improvement in system performance can be achieved by customizing the configuration of `MediaWiki` and `MySQL`. Indeed this performance gain is necessary, because processing a top ten wikipedia is very demanding in terms of disk access, CPU utilization, memory utilization, and Internet bandwidth. The Wikimedia Foundation publishes dump files on a monthly basis (more or less), and, with merely default performance, the processing of a large dump file would take longer than a month.

3.1 General Information

3.1.1 Storage at the Wikimedia Foundation

The Wikimedia Foundation maintains about a thousand wiki's, including wikipedias for nearly 300 languages.

3.1.1.1 `XML` dump files

The Foundation provides compressed dump files of each wiki. These files have names like
`enwiki-yyyymmdd-pages-articles.xml.bz2`,
`enwiktionary-yyyymmdd-pages-articles.xml.bz2`,
`simplewiki-yyyymmdd-pages-articles.xml.bz2`, etc.
The leading characters are a language code: de for German, en for English, fr for French, ja for Japanese, ru for Russian, simple for Simple English, etc. The next few characters are the project: wikibooks, wiktionary, wiki for Wikipedia, etc. The string *yyyymmdd* is replaced by a date.

New dump files appear every month, more or less. These dump files are posted on one of the Foundation's web sites. You can expect URL's like
`http://dumps.wikimedia.org/enwiki/yyyymmdd/...`,
`http://dumps.wikimedia.org/enwiktionary/yyyymmdd/...`,
`http://dumps.wikimedia.org/simplewiki/yyyymmdd/...`, etc.
The dump files of greatest interest are the `pages-articles` dumps, which contain the latest revisions of articles, templates, image descriptions, and the primary meta-pages. They do not contain talk, old revisions, or user pages. You can expect URL's like
`http://dumps.wikimedia.org/enwiki/yyyymmdd/enwiki-yyyymmdd-pages-articles.xml.bz2`,

http://dumps.wikimedia.org/enwiktionary/yyyymmdd/enwiktionary-yyyymmdd-pages-articles.xml.bz2,
http://dumps.wikimedia.org/simplewiki/yyyymmdd/simplewiki-yyyymmdd-pages-articles.xml.bz2,
etc.

3.1.1.2 SQL dump files

Also posted are dump files of individual database tables, and they have URL's like
http://dumps.wikimedia.org/enwiki/yyyymmdd/enwiki-yyyymmdd-category.sql.gz,
http://dumps.wikimedia.org/enwiki/yyyymmdd/enwiki-yyyymmdd-categorylinks.sql.gz,
http://dumps.wikimedia.org/enwiki/yyyymmdd/enwiki-yyyymmdd-externallinks.sql.gz,
etc.

3.1.1.3 Images

Images are stored separately from everything else. Images that are used by more than one wiki are stored in the commons directory tree with URL's like
http://upload.wikimedia.org/wikipedia/commons/0/00/Arc_en_ciel.png.
Images that are unique to a given wiki have their own directory tree. You can expect URL's with names something like
http://upload.wikimedia.org/wikipedia/en/...,
http://upload.wikimedia.org/wiktionary/en/...,
http://upload.wikimedia.org/wikipedia/simple/..., etc.

The Wikimedia Foundation does not provide a dump file of images. Nor does it provide any kind of bulk download method (e.g. rsync, bit torrent files). Copyright and bandwidth issues have been cited. However, there are sites that do provide bulk download methods (see next).

3.1.2 Storage at Other Dump Sites

Mirror sites that provide dump files and image dump tarballs are listed on http://meta.wikimedia.org/wiki/Mirroring_Wikimedia_project_XML_dumps. [1]

3.1.3 Storage on Mirror

As for your computer: Articles, templates, image metadata, etc. are stored in the form of a database. We use MySQL as the database management system (and so does WMF). MySQL offers many storage engines (sub-systems that manage the data in memory and on disk). The InnoDB storage engine is preferred, because it supports transactions and is 'ACID compliant' (explained in the Design Note in §3.2.1, Plan Your Disk Space below). As already mentioned, the Wikimedia Foundation makes compressed dump files available. These dump files must be downloaded, validated, decompressed, and then imported into the database. WP-MIRROR automates all these tasks for you.

Image files for wikis, are stored in a file system, rather than a database. This is a design decision on the part of people at the Wikimedia Foundation. It recognizes that web servers are better at caching files than database contents.

[1] The mirror site http://ftpmirror.your.org/pub/wikimedia/imagedumps/tarballs/fulls/ provides image dumps, but the Wikimedia Foundation issues to following caution:

> The Wikimedia Foundation has permission to use certain images, and many of the fair use images are borderline in terms of whether they can be used or not off Wikipedia. If you choose to download the image base, you do so at your own risk and assume all liability for the use of any images on the main Wikipedia site. The Wikipedia Community vigorously police the site and remove infringing images daily, however, it is always possible that some images may escape this extraordinary level of vigilance and end up on the site for a short time. As of February of 2007, the entire collection of images produce a compressed tar.gz file of over 213 GB (gigabytes). As of November 2011 the image and other media files take up about 17T, most of it already compressed media.
> http://meta.wikimedia.org/wiki/Importing_a_Wikipedia_database_dump_into_mediawiki, accessed 2012-12-22

The author recommends using a modern journalling file system (such as ReiserFS).

3.1.4 WP-MIRROR

WP-MIRROR can read the dump file, identify the image files needed, and download them. Again WP-MIRROR automates all these tasks for you.

When you wish to access your mirror, open a browser and enter the URL http://simple.wikipedia.site/. This URL is a virtual host set up for your convenience by WP-MIRROR. Your web server (e.g. Apache2) will first resolve the virtual host name to ::1, which is your localhost. Then it will call the MediaWiki PHP script /var/lib/mediawiki/index.php. This script will in turn interact with MySQL and your file system to fetch articles and images, respectively. Then the web server will serve the articles and images to your browser, which will then render the page.

Currently (year 2013), the ten largest wikipedias are judged to be too large to fit on a laptop PC equipped with a single 500G hard disk drive (HDD). They are: the en wikipedia (the largest at about 3T), followed by the de, nl, fr, it, ru, es, sv, pl, and ja wikipedias. When HDDs of larger capacity reach the market, this list may be shortened. But for now, a desktop PC with ample disk space and memory (DRAM) is required for any of the top ten, unless you can make do without the images (and this is configurable). Most other languages should fit on a laptop. The simple wiki at about 90G (as of year 2013) should fit easily.

If you will be building your mirror on a laptop PC, please read §3.2, Laptop Planning and Configuration below. It contains detailed instructions for building a mirror of the simple wikipedia and simple wiktionary (default).

If you will be building your mirror on a desktop PC, please read both §3.2, Laptop Planning and Configuration and §3.3, Desktop Planning and Configuration below. The latter contains instructions for building a mirror of the en wikipedia. This is the most demanding case. You should expect a steeper learning curve and allow yourself more time.

3.2 Laptop Planning and Configuration

Laptops usually have a single HDD, so the configuration is simpler (albeit with reduced performance). In this section, the author assumes that you will be building a mirror of the simple wikipedia and simple wiktionary.

To reduce the labor involved in configuring dependencies such as apache2, MediaWiki, and MySQL; many installation and post-installation configuration activities have been automated in WP-MIRROR 0.4 and later versions. This is shown in Table 3.1, Automation of laptop Configuration.

3.2.1 Plan Your Disk Space

3.2.1.1 Configure HDD Write Caching

3.2.1.1.1 Purpose: Permit trade-off between database performance and Durability (the 'D' in ACID).

3.2.1.1.2 Motivation: Enabling HDD write cache improves performance (22% less time). Disabling HDD write cache helps assure Durability of individual COMMITted transactions. See §G.3, fsm-file-import and Durability.

3.2.1.1.3 Implementation: Automatic configuration of the HDD write cache was introduced with WP-MIRROR 0.1. No user configuration is necessary.

Beginning with WP-MIRROR 0.6, the HDD write cache is *enabled* (default).

For WP-MIRROR 0.5 and earlier versions, the HDD write cache was *disabled* (default).

The user who wishes to configure the HDD write cache may do so, and two parameters have been introduced in WP-MIRROR 0.6 for this purpose:

Table 3.1: Automation of `laptop` Configuration

Category	Configuration	Auto	Warn	Opt
Disk	Configure HDD Write Caching	Y		
	Assert Adequate Disk Space		Y	
	Setup Thermal Management			Y
	Setup `smart` Disk Monitoring			Y
DBMS	Secure Database Management System		Y	
	Load Time Zone Tables	Y		
	Configure `InnoDB` Storage Engine	Y		
DRAM	N/A			
Internet	N/A			
mediawiki	Configure `MediaWiki`	Y		
	Enable `MediaWiki` extensions	Y		
Images	Replace `ImageMagick` with `GraphicsMagick`	Y		
	Replace `SVG` Converter	Y		
Host	Enable Virtual Host	Y		
Proxy	Configure `bash` for Use with Caching Web Proxy			Y
	Configure `cURL` for Use with Caching Web Proxy			Y
	Configure `wget` for Use with Caching Web Proxy)			Y
	Configure Browser for Use with Caching Web Proxy			Y

Table 3.2: History of Configuring `hdparm`

Version	Configuration
≥ 0.6	automated, enabled (default), no user configuration is necessary
≤ 0.5	automated, disabled (default), no user configuration is necessary

Name	Default	Range
`*system-hdd-write-cache*`	`1`	`0, 1`
`*system-hdd-write-cache-flush*`	`nil`	`nil, t`

The first parameter `*system-hdd-write-cache*` defaults to `1` (HDD write cache enabled). When set to `0`, the HDD write cache is disabled. WP-MIRROR configures HDD write caching by executing something like:

```
root-shell# hdparm -W1 /dev/sda          <-- enable, or
root-shell# hdparm -W0 /dev/sda          <-- disable
```

The second parameter `*system-hdd-write-cache-flush*` defaults to `nil` (no flushing after each checkpoint). When set to `t`, the HDD write cache is flushed immediately after each checkpoint. WP-MIRROR flushes the HDD write cache by executing something like:

```
root-shell# hdparm -F /dev/sda
```

Write caching is configured only for the disk(s) holding your `InnoDB` data. Disabling HDD write caching helps assure Durability (the 'D' in ACID) of committed transactions.

You may manually disable write caching for an hard drive (and this is *not* recommended) by configuring `/etc/rc.local` by inserting the following lines:

```
hdparm -W0 /dev/sda
exit 0
```

But one hazard in doing so arises for systems using Linux dynamic device management, udev. There is no guarantee that /dev/sda will refer to the same HDD each time the system is rebooted, unless there is only one HDD. Instead you will need the UUID of the disk, and this is found by executing:

```
shell$ ls -l /dev/disk/by-uuid/
```

Design note. A virtual machine presents an important exception. A virtual machine provides a virtual disk. Configuration of a virtual disk's write cache is impossible (and makes no sense). In this case, a warning is issued.

The user who requires Durability for database transactions on a virtual machine, must log into the underlying host machine, and manually disable write caching for its physical hard drives.

Design note. WP-MIRROR 0.6 offers a design compromise.

- *system-hdd-write-cache*. Enabling HDD write caching can enhance performance, but sacrifices the Durability (the 'D' in ACID) of each individual transaction.
- *system-hdd-write-cache-flush*. However, this loss of Durability is mitigated somewhat by flushing the HDD write caches immediately after each checkpoint. Flushing assures that all transactions up to the last checkpoint are Durable.

In the event of a system failure that wipes the contents of the HDD write caches (e.g. power fails), the loss is not tragic because WP-MIRROR resumes from the last checkpoint, and therefore repeats any transactions that were lost.

Design note. When we say that a storage engine is 'ACID compliant', we mean that database transactions will have the following four features: Atomicity, Consistency, Isolation, and Durability.

- **Atomicity** means that database transactions follow the all-or-nothing rule—they either commit or they leave the database unchanged (there are no half-measures).
- **Consistency** means that the database is never in an invalid state, not even for a moment.
- **Isolation** means that one transaction can not access uncommitted changes made by another transaction.
- **Durability** means that committed transaction will not be lost due to system failure.

3.2.1.2 Put images Directory on Partition with Adequate Free Space

Table 3.3: History of Configuring images Directory

Version	Configuration
≥ 0.6	warning issued if /var/ has <90G free space; user configuration optional (see below)
$0.4--0.5$	warning issued if /var/ has <60G free space; user configuration optional (see below)
≤ 0.3	user configuration optional (see below)

3.2.1.2.1 Purpose: Assure adequate storage for WP-MIRROR installation.

3.2.1.2.2 Motivation: WP-MIRROR is storage intensive. Databases and image files are stored under `/var/`. Stuffing that partition would interfere with other processes.

3.2.1.2.3 Implementation: A mirror of the simple wikipedia and simple wiktionary (the default), requires about 90G (as of year 2013). Its image files are stored under `/var/lib/mediawiki/images/`.

WP-MIRROR periodically checks the amount of free disk space. If `/var/` has less than 90G free space, WP-MIRROR issues a warning. If `/var/` has less than 5G, WP-MIRROR gracefully exits.

If you mounted `/var/` on its own partition (which you should do), and if that partition has insufficient free space, then there is still hope. If may not be necessary to repartition your HDD. If you mounted `/home/` on its own partition (which is recommended), and if it has enough free space, then you can, instead, store the images there and set a symbolic link. This is done by executing:

```
root-shell# mkdir /home/images
root-shell# chown --recursive www-data:www-data /home/images
root-shell# cd /var/lib/mediawiki/
root-shell# rm --recursive images/
root-shell# ln --symbolic /home/images/ images
```

3.2.1.3 Setup Thermal Management

Table 3.4: History of Configuring `hddtemp`

Version	Configuration
$\geq$ 0.1	user configuration optional (see below)

3.2.1.3.1 Purpose: Avoid harming HDDs.

3.2.1.3.2 Motivation: Running WP-MIRROR puts quite a load on CPU, HDD, and memory. Hot HDDs (e.g. over 50°C) do not last. HDD failure is traumatic for users not in the habit of performing backups.

3.2.1.3.3 Implementation: Check the temperature of the HDD by executing:

```
root-shell# aptitude install hddtemp
root-shell# /usr/sbin/hddtemp /dev/sda
```

or perhaps

```
root-shell# aptitude install smartmontools
root-shell# smartctl -a /dev/sda | grep Celsius
```

Make sure that the airflow is unobstructed. Better yet, go to your kitchen, and look for a cookie rack (a wire frame used for letting freshly backed cookies cool). Put the cookie rack under your laptop PC.

3.2.1.4 Setup `smart` Disk Monitoring

3.2.1.4.1 Purpose: Monitor health of HDDs.

Table 3.5: History of Configuring smart Disk Monitoring

Version	Configuration
≥ 0.1	user configuration optional (see below)

3.2.1.4.2 Motivation: Disks age, wear out, and fail. It is important to monitor the health of all your HDDs.

3.2.1.4.3 Implementation: Most HDDs are equipped with on-board diagnostics—the Self-Monitoring, Analysis, and Reporting Technology (S.M.A.R.T.) System. The software utilities smartctl and smartd let you control and monitor such disks. smartd can send you an e-mail warning of imminent disk failure.

To enable smartd to run as a daemon, edit the configuration file /etc/default/smartmontools by uncommenting the line to read

```
start_smartd=yes
```

Then, to monitor all attributes, to schedule a short self-test daily at 2am, a long self-test weekly on Saturdays at 3am, and to receive an email of disk failure, edit the configuration file /etc/smartd.conf to read:

```
/dev/sda -d ata -a -o on -S on -s (S/../../.02|L/../../6/03)
/dev/sda -d ata -H -m root@localhost
```

From time-to-time you should read the test report by executing:

```
root-shell# smartctl -a /dev/sda
```

You should pay careful attention to attributes of type Pre-fail. Look also for evidence that the disk is encountering bad sectors, by finding the attributes Reallocated_Sector_Ct and Current_Pending_Sector. Check also that the scheduled tests are actually running by comparing the attribute Power_On_Hours with the LifeTime(hours) column of the Self-test log.

3.2.2 Plan Your Database Management System

It is simplest if you start with a clean database.

3.2.2.1 Secure the Database Management System

Table 3.6: History of Configuring MySQL Security

Version	Configuration
≥ 0.4	warning issued if no security; user configuration optional (see below)
≤ 0.3	user configuration optional (see below)

3.2.2.1.1 Purpose: Enhance database security.

3.2.2.1.2 Motivation: MySQL by default has no password for the root account. Some people think that this is safe. The author respectfully disagrees.

3.2.2.1.3 Implementation: Security testing of the database management system (DBMS) was introduced with WP-MIRROR 0.4. Warnings are issued as needed. User configuration is optional but highly recommended. You should secure your installation by executing:

```
root-shell# mysql_secure_installation
```

This interactive script prompts you do five things:

- set a password for root accounts,
- remove anonymous-user accounts,
- remove root accounts that are accessible from outside the local host,
- remove the `test` database,
- reload the privilege tables so that changes take effect immediately.

Alternatively, you can do this manually by executing:

```
shell$ mysql --user=root
mysql> UPDATE mysql.user SET password=PASSWORD('new_pwd') WHERE user='root';
mysql> DELETE FROM mysql.user WHERE user='root'
    ->   AND host NOT IN ('localhost', '127.0.0.1', '::1');
mysql> DELETE FROM mysql.user WHERE user='';
mysql> DELETE FROM mysql.db WHERE db='test' OR db='test_%';
mysql> DROP DATABASE test;
mysql> FLUSH PRIVILEGES;
```

where *new_pwd* is replaced by a password of your choice.

3.2.2.2 Load the DBMS `time_zone` Tables

Table 3.7: History of Configuring `MySQL` `time_zone` Tables

Version	Configuration
≥ 0.4	automated, no user configuration is necessary
≤ 0.3	user configuration required (see below)

3.2.2.2.1 Purpose: Assure correct handling of timestamps.

3.2.2.2.2 Motivation: `Mediawiki` and `MySQL` handle timestamps differently.

3.2.2.2.3 Implementation: Automatic loading of the `time_zone` tables was introduced with WP-MIRROR 0.4. No user configuration is necessary.

WP-MIRROR checks that the `time_zone` tables are populated; and, if they are not, loads them from another package (namely, the `tzdata` package which installs its time zone information under `/usr/share/zoneinfo/`).

For WP-MIRROR 0.3 and earlier versions, it is necessary to load the `time_zone` tables manually by executing:

```
shell$ mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql --host=localhost \
--user=root --password mysql
Enter password:
root-shell# /etc/init.d/mysql restart
```

where the password you enter should be your MySQL root password chosen in §3.2.2.1, Secure the Database Management System above.

Design note. MediaWiki has its own kind of timestamp, and does not use the MySQL timestamp. This can be seen when running the /usr/share/mediawiki/maintenance/update.php script, which emits the message:

```
Converting tc_time from UNIX epoch to mediawiki timestamp... done.
```

MySQL has tables that store time zone information. These tables enable recognition of named time zones.

While not required by MediaWiki, these time_zone tables are useful. This is because, several of the MediaWiki tables (e.g. image, logging, page, etc.) have a field that contains a MediaWiki timestamp. These fields are of type BINARY(14) rather than of type TIMESTAMP. According to the MySQL 5.5 Reference Manual:

> MySQL converts TIMESTAMP values from the current time zone to UTC for storage, and back from UTC to the current time zone for retrieval.
>
> Section 11.3.1, The DATE, DATETIME, and TIMESTAMP Types, accessed 2012-12-19

This is not true of type BINARY(14).

This difference in behaviour could result in converting or comparing timestamps incorrectly.

When the MySQL data directory is first installed, these tables are empty.

3.2.2.3 Configure the InnoDB Storage Engine

Table 3.8: History of Configuring InnoDB Storage Engine

Version	Configuration
≥ 0.4	automated, no user configuration is necessary
≤ 0.3	user configuration required (see below)

3.2.2.3.1 Purpose: Increase DBMS performance.

3.2.2.3.2 Motivation: With proper DBMS configuration, one can improve performance by an order of magnitude. See Figure E.1, InnoDB Experiments—Importing imagelinks Table and Figure G.1, InnoDB Experiments—Importing commonswiki.image Table.

3.2.2.3.3 Implementation: Automatic customizing of the InnoDB storage engine was introduced with WP-MIRROR 0.4. No user configuration is necessary.

For WP-MIRROR 0.3 and earlier versions, configuration is done as follows:

First, copy the configuration file for the laptop case by executing:

```
root-shell# cd /usr/share/doc/wp-mirror/examples/
root-shell# cp wp-mirror_laptop.cnf /etc/mysql/conf.d/.
root-shell# cd /etc/mysql/conf.d/
root-shell# mv wp-mirror_laptop.cnf wp-mirror.cnf
```

Second, delete the existing InnoDB log files, and restart MySQL by executing:

```
root-shell# /etc/init.d/mysql stop
root-shell# rm /var/lib/mysql/ib_logfile*
root-shell# /etc/init.d/mysql start
```

Alternatively, you can try the cut-and-paste method:
First, edit /etc/mysql/conf.d/wp-mirror.cnf to read:

```
[mysqld]
# time zone UTC
default-time-zone               = UTC
# transactional (ACID) storage engine.
default-storage-engine          = innodb
# UTF-8 character set and collation.
character-set-server            = utf8
collation-server                = utf8_general_ci

# InnoDB

# to speed disk I/O bound operations (compress tables using 'zlib')
innodb_file_per_table           =    1 # default  OFF. Create .ibd file/table
innodb_file_format              = Barracuda # default Antelope.
innodb_file_format_check        =    1 # default   ON.
innodb_strict_mode              =    1 # default    0.
# increase size of buffer pool and log files for speed
innodb_buffer_pool_size         =1024M # default 128M.
innodb_log_buffer_size          =  80M # default   8M.
innodb_log_file_size            = 512M # default   5M.
innodb_additional_mem_pool_size =   8M # default   8M. (data dict, internals)
# for Durability (the 'D' in 'ACID compliance')        (0=flush log buff 1/sec)
innodb_flush_log_at_trx_commit  =    0 # default    1. (2=write log file 1/sec)
sync_binlog                     =    0 # default    0.
# fast index creation requires large TMP directory
tmpdir                          = /var/tmp # default /tmp
# enable large page support. InnoDB will use it automatically for its
# buffer_pool and its additional_memory_pool
large_pages                     =    1 # default    0.

# Other

lower_case_table_names          =    1 # default    0. (use 1 for InnoDB)
# <meta.wikimedia.org/wiki/Data_dumps> recommends 20M.
max_allowed_packet              =  64M # default  16M.
query_cache_size                =  16M # default  16M.
query_cache_type                =    1 # default   ON.

[mysql]
default-character-set           = utf8
# <meta.wikimedia.org/wiki/Data_dumps> recommends 20M.
max_allowed_packet              =  64M # default  16M.
```

Second, delete the existing log files, and restart MySQL as above.

Design note. InnoDB has four ways of storing the data on disk:

1. Antelope (data uncompressed) with all data stored in a single file, called a data file or table space (default),
2. Antelope with each database in a separate directory and each table in a separate file,
3. Antelope with the table space written directly onto a raw partition (no file system), and
4. Barracuda (data compressed using zlib) with each database in a separate directory and each table in a separate file (as in case 2).

Barracuda (case 4) is best for a laptop PC, because data compression saves disk space and improves SELECT performance. However, it also requires much more CPU power, for compressing your data before writing it to disk, and for decompressing it after reading it back from disk. Barracuda performs poorly at concurrency (handling multiple connections). Experiments show that, when running two or more instances of WP-MIRROR in mirror mode, two out of three xchunks experience deadlocks and fail to install completely. However, the laptop user normally starts a single instance in mirror mode, lets it run overnight, and no deadlocks occur. For WP-MIRROR 0.5 and earlier versions, the author recommends Barracuda for mirroring small wikis (but not the top ten). For WP-MIRROR 0.6 and later versions, Barracuda is recommended for all wikis.

Antelope (case 3) is best for a desktop PC or a server, because it is fast, it handles concurrency well, and because disk space is cheap. Writing directly onto a raw partition is faster, because it avoids an extra layer of journalling. All modern file systems use journalling to improve crash recovery. However, InnoDB's table space already has its own journaling system. Storing the table space on top of a journaling file system (case 1) is unecessary and slow—and it makes the HDDs run harder and hotter. Antelope also handles concurrency well. It is normal for an user to start two or three instances of WP-MIRROR in mirror mode, and let them run for days or weeks. Very few deadlocks occur, and very few xchunks fail to install completely. For WP-MIRROR 0.5 and earlier versions, the author recommends it (case 3) for mirroring the top ten wikis. For WP-MIRROR 0.6 and later versions, Antelope is no longer recommended.

Third, the buffer pool is the space in memory (DRAM) where InnoDB stores data that it is currently processing. A large buffer pool improves performance, because more of your data is available in memory (fast), and less of it has to be read from disk (slow). For large wikipedias, such as the en wikipedia, it may be necessary to install extra DRAM to accommodate a large buffer pool.

Fourth, the log files are the journals where InnoDB first logs any transactions that will be written to disk. Whenever the log files fill up, InnoDB must stop and flush the data from memory to disk. Large log files can improve performance. Beginning with MySQL 5.5 it is reasonable to set innodb_log_file_size to half the value of innodb_buffer_pool_size, and innodb_log_buffer_size should be set to the size that can be written to disk in one second.

3.2.2.4 MySQL Redux

3.2.2.4.1 Purpose: Reinstall MySQL.

3.2.2.4.2 Motivation: The word 'redux' means brought back or restored. If you experiment aggressively with MySQL you might inadvertently trash your installation. If this happens to you, there is an easy way to make a fresh start.

3.2.2.4.3 Implementation: First, delete the old installation by executing:

```
root-shell# /etc/init.d/mysql stop
root-shell# rm -r /var/lib/mysql/*
root-shell# rm /etc/mysql/conf.d/wp-mirror.cnf
```

Second, make a clean install by executing:

```
root-shell# mysql_install_db --user=mysql
root-shell# /etc/init.d/mysql start
```

Third, restore the debian-sys-maint account info by executing:

```
root-shell# cat /etc/mysql/debian.cnf
[client]
host     = localhost
user     = debian-sys-maint
password = abcdefghijklmnop
socket   = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
host     = localhost
user     = debian-sys-maint
password = abcdefghijklmnop
socket   = /var/run/mysqld/mysqld.sock
basedir  = /usr
shell$ mysql --host=localhost --user=root
mysql> CREATE USER 'debian-sys-maint'@'localhost' IDENTIFIED BY 'abcdefghijklmnop';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'debian-sys-maint'@'localhost' WITH GRANT OPTION;
mysql> FLUSH PRIVILEGES;
mysql> quit;
```

Fourth, improve security by executing:

```
root-shell# mysql_secure_installation
```

Finally, restore the WP-MIRROR configuration by executing:

```
root-shell# wp-mirror --mirror
```

3.2.3 Plan Your DRAM

No configuration is required for laptops.

3.2.4 Plan Your Internet Access

3.2.4.1 Configure cURL (Obsolete)

Table 3.9: History of Configuring cURL

Version	Configuration
≥ 0.6	obsolete, no user configuration is necessary
$0.4 - 0.5$	automated, no user configuration is necessary
≤ 0.3	user configuration required (see below)

3.2.4.1.1 Obsolete: cURL no longer used for downloading dump files or image files.

3.2.4.1.2 Purpose: Prevent cURL from blocking.

3.2.4.1.3 Motivation: By default, cURL does not time-out when a file takes a long time to download. This means that cURL will block when a file fails to download completely.

3.2.4.1.4 Implementation: For WP-MIRROR 0.6 and later, cURL is no longer used for downloading dump files and image files. No user configuration is necessary.

For WP-MIRROR 0.5 and earlier versions, the utility cURL is used for downloading dump files and image files.

Automatic configuration of cURL was introduced with WP-MIRROR 0.4. No user configuration is necessary. WP-MIRROR does this by appending the command-line option `-m 1000` to all invocations of cURL.

For WP-MIRROR 0.3 or earlier versions, you must manually edit (or create) the configuration file `/root/.curlrc` by appending the lines:

```
# We want 1ks timeout (initially and for each retry)
--max-time 1000
```

3.2.5 Plan Your MediaWiki

3.2.5.1 Configure MediaWiki

Table 3.10: History of Configuring MediaWiki

Version	Configuration
≥ 0.4	automated, no user configuration is necessary
≤ 0.3	user configuration required (see below)

3.2.5.1.1 Purpose: Use MediaWiki as wiki engine to mirror WMF wikis.

3.2.5.1.2 Motivation: MediaWiki needs DBMS credentials to access wiki content.

3.2.5.1.3 Implementation: Automatic configuration of MediaWiki was introduced with WP-MIRROR 0.4. No user configuration is necessary.

For WP-MIRROR 0.3 and earlier version, you must manually configure MediaWiki as follows:

First, open a browser to `http://localhost/mediawiki/config/index.php` and fill in the blanks with:

```
Site config
Wiki name: Wikipedia
Contact e-mail: webmaster@localhost
Language: simple - Simple English
Copyright/license: (*) No licence metadata
Admin username: WikiSysop
Password: ********
Password confirm: ********
Object caching: (*) No caching
Memcached servers: <blank>
E-mail, e-mail notification and authentication setup
E-mail features (global): (*) Disabled
User-to-user e-mail: (*) Disabled
E-mail notification about changes: (*) Disabled
E-mail address authentication: (*) Disabled
Database config
Database type: (*) MySQL
Database host: localhost
Database name: wikidb
DB username: wikiuser
DB password: ********
DB password confirm: ********
Superuser account: [x] Use superuser account
Superuser name: root
Superuser password: ********
MySQL specific options
Database table prefix: <blank>
Storage Engine: (*) InnoDB
Database character set: (*) MySQL 4.1/5.0 binary
```

If you are installing on a laptop, and are the only user, it is reasonable to use the same password for all the database accounts (namely, `root`, `WikiSysop`, and `wikiuser`) by entering the same password that you created in §3.2.2.1, Secure the Database Management System.

Second, press the `Install mediawiki!` button. The browser may take a minute to respond while it: 1) creates and populates the database `wikidb`, and 2) writes the file `/var/lib/mediawiki/config/LocalSettings.php`.

Third, move the `LocalSettings.php` into place and set permissions to protect passwords, by executing:

```
root-shell# mv /var/lib/mediawiki/config/LocalSettings.php /etc/mediawiki/.
root-shell# chown www-data:www-data LocalSettings.php
root-shell# chmod 600 LocalSettings.php
```

Fourth, WP-MIRROR provides a supplementary configuration file `/etc/mediawiki/LocalSettings_wpmirror.php` which contains several important customizations. To include this, edit `/etc/mediawiki/LocalSettings.php` by appending the lines:

```
# wp-mirror specific include:
if (is_file( '/etc/mediawiki/LocalSettings_wpmirror.php' ) ) {
 include( '/etc/mediawiki/LocalSettings_wpmirror.php' );
}
```

Fifth, move the administrator config file into place and set permissions to protect passwords, by executing:

```
root-shell# cd /usr/share/doc/mediawiki/examples/
root-shell# cp -a AdminSettings.sample /etc/mediawiki/AdminSettings.php
root-shell# cd /etc/mediawiki/
root-shell# chmod 600 /etc/mediawiki/AdminSettings.php
```

Finally, edit the /etc/mediawiki/AdminSettings.php by appending:

```
$wgDBadminuser = 'root';
$wgDBadminpassword = 'new_pwd';
```

where *new_pwd* should be replaced with the password you set above in §3.2.2.1, Secure the Database Management System. The credentials in the administrator config file are needed by many of the scripts in /usr/share/mediawiki/maintenance/.

3.2.5.2 Enable MediaWiki Extensions

Table 3.11: History of Configuring mediawiki-extensions

Version	Configuration
≥ 0.4	automated, no user configuration is necessary
≤ 0.3	user configuration required (see below)

3.2.5.2.1 Purpose: Assure proper formatting of wiki pages.

3.2.5.2.2 Motivation: Without MediaWiki extensions enabled, most of the Wikipedia articles will look messy; and articles with citations will be almost unreadable.

3.2.5.2.3 Implementation: Automatic configuration of the mediawiki-extensions was introduced with WP-MIRROR 0.4. No user configuration is necessary.

For WP-MIRROR 0.3 and earlier versions, extensions are configured by setting a couple dozen links. This is most easily done by executing:

```
root-shell# cd /etc/mediawiki-extensions/extensions-enabled/
root-shell# cp -a ../extensions-available/* .
```

3.2.5.3 MediaWiki Redux

3.2.5.3.1 Purpose: Reinstall MediaWiki.

3.2.5.3.2 Motivation: The word 'redux' means brought back or restored. If you mess up your installation of MediaWiki, there is an easy way to make a fresh start.

3.2.5.3.3 Implementation: For WP-MIRROR 0.4, which uses MediaWiki 1.19, restart the installation as follows:

First, delete the old configuration by executing:

```
root-shell# rm /etc/mediawiki/LocalSettings.php
```

Second, open a browser to http://localhost/mediawiki/index.php. You will get message what says:

```
LocalSettings.php not found.
Please [set up the wiki] first.
```

Third, click on the link "set up the wiki", which will take you to http://localhost/mediawiki/mw-config/index.php.

Finally, on that page click on the link which says "Restart installation".

For WP-MIRROR 0.3 and earlier versions, which use MediaWiki 1.15, restarting the installation as follows:

First, drop the database by executing:

```
shell$ mysql --host=localhost --user=root --password
Enter password:
...
mysql> DROP DATABASE wikidb;
```

Second, edit /var/lib/mediawiki/config/index.php by commenting out the following lines (scroll down about 220 lines):

```
#if( file_exists( "../LocalSetings.php" ) ) {
#  $script = defined('MW_INSTALL_PHP5_EXT') ? 'index.php5 : 'index.php';
#  dieout( "<p><strong>Setup has completed, <a href='../$script'>your wiki</a>
     is configured.</strong></p>
#  <p>Please delete the /config directory for extra security.</p>" );
#}
```

Finally, redo the installation described above in §3.2.5.1, Configure MediaWiki.

3.2.6 Plan Your Image Processing

3.2.6.1 Replace ImageMagick with GraphicsMagick

Table 3.12: History of Configuring Image Processing

Version	Configuration
≥ 0.1	automated, no user configuration is necessary

3.2.6.1.1 Purpose: Prevent ImageMagick from hanging the system.

3.2.6.1.2 Motivation: By default, MediaWiki processes image files with convert from ImageMagick. However, convert often grabs too much memory, which can cause: poor performance, memory starvation of other processes, and system hang.

3.2.6.1.3 Implementation: It is better to use gm convert from GraphicsMagick.

Automatic replacement of ImageMagick with GraphicsMagick in MediaWiki's configuration was introduced with WP-MIRROR 0.1. No user configuration is necessary.

WP-MIRROR provides a supplementary configuration file /etc/mediawiki/LocalSettings_wpmirror.php that deals with the image processing issue by providing the following lines:

```
$wgUseImageMagick=false;
$wgCustomConvertCommand="/usr/bin/gm convert %s -resize %wx%h %d";
```

Instructions for installing the supplementary configuration file are found above in §3.2.5.1, Configure MediaWiki.

Design note. Most of the older and larger articles have images. Usually, the authors of such articles provide images that do not fit the page layout. So MediaWiki resizes them. The resized images, which are called 'thumb's, are stored in a directory tree under
/var/lib/mediawiki/images/wikipedia/common/thumb/,
/var/lib/mediawiki/images/wikipedia/simple/thumb/, and
/var/lib/mediawiki/images/wiktionary/simple/thumb/.

By default, MediaWiki uses convert from ImageMagick. However, convert often grabs too much memory, which can:

- cause poor performance,
- cause other processes to fail for lack of memory to allocate, and
- cause your system to hang.

To reduce your frustration, you should instead have MediaWiki resize images using gm convert from GraphicsMagick.

3.2.6.2 Replace SVG Converter

Table 3.13: History of Configuring SVG to PNG Conversion

Version	Configuration
≥ 0.1	automated, no user configuration is necessary

3.2.6.2.1 Purpose: Prevent ImageMagick from hanging the system.

3.2.6.2.2 Motivation: Images provided in SVG format are usually converted into PNG format, because some browsers do not render SVG files properly.

By default, MediaWiki converts SVG images using convert from ImageMagick. However, convert often grabs too much memory, which can cause: poor performance, memory starvation of other processes, and system hang.

3.2.6.2.3 Implementation: It is better to use inkscape or rsvg.

Automatic replacement of ImageMagick in MediaWiki's configuration was introduced with WP-MIRROR 0.1. No user configuration is necessary.

WP-MIRROR provides a supplementary configuration file /etc/mediawiki/LocalSettings_wpmirror.php that deals with the image processing issue by providing the following lines:

For WP-MIRROR 0.4 and later versions, inkscape is preferred.

```
$wgUseImageMagick=false;
$wgCustomConvertCommand="/usr/bin/gm convert %s -resize %wx%h %d";
$wgSVGConverter="inkscape";
```

For WP-MIRROR 0.3 and earlier versions, rsvg is preferred.

```
$wgUseImageMagick=false;
$wgCustomConvertCommand="/usr/bin/gm convert %s -resize %wx%h %d";
$wgSVGConverter="rsvg";
```

Instructions for installing the supplementary configuration file are found above in §3.2.5.1, Configure MediaWiki.

3.2.7 Plan Your Virtual Host

3.2.7.1 Enable Virtual Host

Table 3.14: History of Configuring apache2 Virtual Host

Version	Configuration
≥ 0.2	automated, no user configuration is necessary
≤ 0.1	user configuration required (see below)

3.2.7.1.1 Purpose: Direct web server to local mirror.

3.2.7.1.2 Motivation: By default, Apache2 will not know that your local mirror exists, and will instead try (unsuccessfully) to look outside your system.

3.2.7.1.3 Implementation: WP-MIRROR sets up virtual hosts named wikibooks.site, wikipedia.site, wiktionary.site, etc. on your own computer. This lets you access your mirror locally by giving your web browser the URL http://simple.wikipedia.site/. If you set up a mirror farm, you will have one URL for each language (replace 'simple' with the language code, e.g. http://en.wikipedia.site/).

Most laptops will not be running a Domain Name Server (DNS), such as bind. Therefore, WP-MIRROR resolves the virtual host names to the localhost, by editing /etc/hosts.

Automatic enabling of the virtual host was introduced with WP-MIRROR 0.2. No user configuration is necessary.

For WP-MIRROR 0.1, a virtual host is configured as follows:

First, create a virtual host container in /etc/apache2/sites-available/wpmirror.site.conf with the following text:

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wpmirror.site
    # for access using:  en.wpmirror.site, simple.wpmirror.site, etc.
    ServerAlias  *.wpmirror.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wikibooks.site
    # for access using:  en.wikibooks.site, simple.wikibooks.site, etc.
    ServerAlias  *.wikibooks.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wikimedia.site
    # for access using:  en.wikimedia.site, simple.wikimedia.site, etc.
    ServerAlias  *.wikimedia.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wikinews.site
    # for access using:  en.wikinews.site, simple.wikinews.site, etc.
    ServerAlias  *.wikinews.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wikipedia.site
    # for access using:  en.wikipedia.site, simple.wikipedia.site, etc.
    ServerAlias  *.wikipedia.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wikiquote.site
    # for access using:  en.wikiquote.site, simple.wikiquote.site, etc.
    ServerAlias  *.wikiquote.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wikisource.site
    # for access using:  en.wikisource.site, simple.wikisource.site, etc.
    ServerAlias  *.wikisource.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wikiversity.site
    # for access using:  en.wikiversity.site, simple.wikiversity.site, etc.
    ServerAlias  *.wikiversity.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wikivoyage.site
    # for access using:  en.wikivoyage.site, simple.wikivoyage.site, etc.
    ServerAlias  *.wikivoyage.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wiktionary.site
    # for access using:  en.wiktionary.site, simple.wiktionary.site, etc.
    ServerAlias  *.wiktionary.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

Second, enable the virtual host, and restart `apache2` by executing:

```
root-shell# a2ensite wpmirror.site.conf
root-shell# /etc/init.d/apache2 reload
```

Third, edit `/etc/hosts` by appending lines like:

```
::1 simple.wikipedia.site
::1 simple.wiktionary.site
```

Finally, for each language specified in `/etc/wp-mirror/local.conf`, append corresponding lines to `/etc/hosts`.

Design note. If you wish to configure your virtual host differently, then create a new virtual host container in a separate file `/etc/apache2/sites-available/your-virtual-host.conf`. Your new virtual host must then be enabled by executing:

```
root-shell# a2dissite wpmirror.site.conf
root-shell# a2ensite your-virtual-host.conf
root-shell# /etc/init.d/apache2 reload
```

You will also have to make corresponding edits to `/etc/hosts` in order to resolve the new names.

3.2.8 Plan Your Caching Web Proxy

If your system sits behind a caching web proxy, then it can happen that many web-access programs fail to connect to the internet.

Most web-access programs look for environment variables to determine if traffic must go through a caching web proxy. These environment variables (`ftp_proxy`, `http_proxy`, and `https_proxy`) provide the host name of the caching web proxy, and the port that the proxy is listening on (usually port 8123). The variables look something like:

```
https_proxy = http://proxy-hostname:8123/
http_proxy  = http://proxy-hostname:8123/
ftp_proxy   = http://proxy-hostname:8123/
```

where *proxy-hostname* is replaced by the host name of your caching web proxy.

3.2.8.1 Configure `bash` for Use with Caching Web Proxy

Table 3.15: History of Configuring `bash` for Caching Web Proxy

Version	Configuration
≥ 0.1	user configuration required (see below)

3.2.8.1.1 Purpose: Assure web-access programs are usable when behind a web caching proxy.

3.2.8.1.2 Motivation: Most web-access programs look for environment variables to determine whether or not traffic must go through a caching web proxy.

3.2.8.1.3 Implementation: These environment variables can be set globally by editing `/etc/bash.bashrc` by appending:

```
export HTTP_PROXY=http://proxy-hostname:8123/
export HTTPS_PROXY=http://proxy-hostname:8123/
export FTP_PROXY=http://proxy-hostname:8123/
```

where *proxy-hostname* is replaced by the host name of your proxy.

Confirm that the environment variables have been declared by executing:

```
shell$ export | grep PROXY
declare -x FTP_PROXY="http://proxy-hostname:8123/"
declare -x HTTPS_PROXY="http://proxy-hostname:8123/"
declare -x HTTP_PROXY="http://proxy-hostname:8123/"
```

If the above environment variables do not appear, then try closing and reopening your shell to load the new environment variables.

3.2.8.2 Configure `cURL` for Use with Caching Web Proxy

3.2.8.2.1 Purpose: Assure `cURL` is usable when behind a web caching proxy.

3.2.8.2.2 Motivation: If your system sits behind a web caching proxy, then it can happen that `cURL` fails to connect to the Internet.

Table 3.16: History of Configuring cURL for Caching Web Proxy

Version	Configuration
≥ 0.1	user configuration required (see below)

3.2.8.2.3 Implementation: cURL looks for environment variables. These can be provided by editing the configuration file /root/.curlrc by appending:

```
proxy "http://proxy-hostname:8123/"
```

where *proxy-hostname* is replaced by the host name of your caching web proxy.

cURL does not have a global configuration file. So your may wish to repeat the process for the shell account of each user.

3.2.8.3 Configure wget for Use with Caching Web Proxy

Table 3.17: History of Configuring wget for Caching Web Proxy

Version	Configuration
≥ 0.1	user configuration required (see below)

3.2.8.3.1 Purpose: Assure wget is usable when behind a web caching proxy.

3.2.8.3.2 Motivation: If your system sits behind a caching web proxy, then it can happen that wget fails to connect to the Internet.

3.2.8.3.3 Implementation: wget looks for environment variables. These can be provided by editing the configuration file /etc/wgetrc with the following lines:

```
# You can set the default proxies for Wget to use for http, https, and ftp.
# They will override the value in the environment.
https_proxy = http://proxy-hostname:8123/
http_proxy  = http://proxy-hostname:8123/
ftp_proxy   = http://proxy-hostname:8123/
```

where *proxy-hostname* is replaced by the host name of your caching web proxy.

3.2.8.4 Configure Browser for Use with Caching Web Proxy

Table 3.18: History of Configuring Browsers for Caching Web Proxy

Version	Configuration
≥ 0.1	user configuration required (see below)

3.2.8.4.1 Purpose: Assure browser can find local mirror.

3.2.8.4.2 Motivation: If your system sits behind a caching web proxy, then attempts to browse http://simple.wikimedia.site/ might get routed to the web caching proxy, which returns an error message.

3.2.8.4.3 Implementation: Examine your browser settings. For Firefox (and related) look under:

```
Edit->Preferences->Advanced->Network->Settings...
```

For Konqueror look under:

```
Settings->Configure Konqueror->Proxy
```

3.3 Desktop Planning and Configuration

Building a mirror of one of the large wikipedias is challenging. If this is your first attempt at building a mirror, then you can expect weeks of effort. There is, however, a learning curve effect. If you first build a mirror of a small wikipedia on a laptop PC, and afterwards build a mirror of a large wikipedia on a desktop PC, then the total time will be greatly reduced.

This is because it is far better to make all your mistakes on a project that can be done in a couple of days, than to make them on a project that takes a couple of weeks (months for WP-MIRROR 0.5 and earlier versions).

In what follows, the author assumes that you would like to build a mirror of the en wikipedia and en wiktionary on a desktop PC (or a server), and that you have already done some learning by building a mirror of the simple wikipedia and simple wiktionary on a laptop PC.

3.3.1 Procure Hardware

Table 3.19: History of Procure Hardware for Desktop

Version	Configuration
$\geq$ 0.6	procure 2x3T HDD, 3-speed case fan (see below)
$\leq$ 0.5	procure 2x2T HDD, 3-speed case fan, 4G DRAM (see below)

You may need additional hardware. We shall assume that you already have a working GNU/Linux distribution on a desktop PC with one HDD `/dev/sda`.

To this, you will add two HDDs: `/dev/sdb` and `/dev/sdc`, for use by this project, and that you will add more DRAM, also for use by this project. Here is your shopping list:

- hard disk drives - qty 2, size 3T (or more) each,
- case fan - qty 1, 3-speed (if disks run hot).

Before buying anything, be sure of the following:

- PC case has vacant bays to mount the HDDs,
- power supply has connectors to power the HDDs,
- power supply can handle another 10W per disk,
- mother-board, or extension card, has connectors and cables,

For WP-MIRROR 0.5 and earlier versions, add to your shopping list:

- memory - qty 1 set, size 4G,

and also be sure of the following:

- mother-board has a pair of empty slots for the DRAM,
- mother-board manual specifies the type of DRAM.

3.3.2 Plan Your Disk Space

3.3.2.1 Configure HDD Write Caching

No user configuration is necessary (see §3.2.1.1, Configure HDD Write Caching above).

3.3.2.2 Setup Thermal Management

Table 3.20: History of Configuring `hddtemp` for Desktop

Version	Configuration
≥ 0.1	user configuration optional (see below)

Same as §3.2.1.3, Setup Thermal Management, except that there will be more disks to manage. Check the temperature of your hard disk drives by executing:

```
root-shell# /usr/sbin/hddtemp /dev/sd[a-c]
```

or perhaps

```
root-shell# smartctl -a /dev/sda | grep Celsius
root-shell# smartctl -a /dev/sdb | grep Celsius
root-shell# smartctl -a /dev/sdc | grep Celsius
```

If your disks are running hot (e.g. over 50°C), they will not last. In which case, you will need to increase the air flow. A '3-speed case fan' comes with a three-way switch 'low-medium-high'. If you already have one, then set it to 'high'. If the noise is too much for you, then procure a case that features a large diameter fan. These provide higher air flow with less noise. Also, because a large fan rotates at a lower RPM, the noise spectrum will be shifted an octave lower, which for many people is easier to bear.

3.3.2.3 Setup `smart` Disk Monitoring

Table 3.21: History of Configuring `smart` Disk Monitoring for Desktop

Version	Configuration
≥ 0.1	user configuration optional (see below)

Same as §3.2.1.4, Setup `smart` Disk Monitoring, except that there will be extra lines to add in `/etc/smartd.conf`:

```
/dev/sdb -a -d ata -o on -S on -s (S/../../.02|L/../../6/03)
/dev/sdc -a -d ata -o on -S on -s (S/../../.02|L/../../6/03)
```

or, if you built your RAID array using an attached USB enclosure (and this is not recommended), then the above two lines might read something like:

```
/dev/sdb -a -d usbjmicron,0 -o on -S on -s (S/../../.02|L/../../6/03)
/dev/sdc -a -d usbjmicron,1 -o on -S on -s (S/../../.02|L/../../6/03)
```

Misconfiguration can easily happen, so you will need to wait a day and then confirm that the scheduled self-test(s) actually ran. This is done by executing:

```
root-shell# smartctl -a /dev/sdb
root-shell# smartctl -a /dev/sdc
```

or, in the case of an attached USB enclosure, something like

```
root-shell# smartctl -a -d usbjmicron,0 /dev/sdb
root-shell# smartctl -a -d usbjmicron,1 /dev/sdc
```

3.3.2.4 Allocate Disk Space for Articles

Articles for the en wikipedia require about 200G (as of 2013).

Articles are head in databases stored under the directory `/var/lib/mysql/`. However, given the size requirement, you would do better to store the databases on a separate disk, and set a symbolic link to it (as shown in §3.2.1, Plan Your Disk Space above).

Implementation instructions are given below in §3.3.2.7, Build Your Storage Array.

3.3.2.5 Allocate Disk Space for Image Files

Image files for the en wikipedia require about 2.3T (as of 2013) and growing.

Images are stored under the directory `/var/lib/mediawiki/images/`. However, given the size requirement, you would do better to store the images on a separate disk, and set a symbolic link to it (as shown in §3.2.1, Plan Your Disk Space above).

Implementation instructions are given below in §3.3.2.7, Build Your Storage Array.

3.3.2.6 Design Storage for Security and Robustness

If you value your data, and especially if `InnoDB` is storing other databases containing confidential records (e.g. `drupal`, `sendmail`, etc.), then security and robustness are paramount.

In §3.3.2.7, Build Your Storage Array we shall: 1) enhance security by using `LUKS` to encrypt all storage; and 2) enhance robustness by using `RAID` to mirror all data.

Design note. "There is no security without security policy." As a good start, you may wish to consider the following:

Table 3.22: Security Policy

Policy	Debian Package
encrypt all storage	`cryptsetup`
backup all valuable data	`mdadm`, `rsync`
encrypt all network traffic	`openssh-client`, `openssh-server`
block all unwanted IP addresses	`pgld`, `pglcmd`
block all unused ports	`iptables`
apply latest security patches	`aptitude`
deinstall all unused services	`aptitude`
close all unused or guest accounts	

To obtain the latest versions of all packages for your system execute:

```
root-shell# aptitude update
root-shell# aptitude safe-upgrade
```

and this should be done daily.

3.3.2.7 Build Your Storage Array

We shall build a RAID array out of whole disks, and then encrypt that.

To allow for future growth in the en wikipedia and en wiktionary, the author recommends procuring a pair of HDDs no smaller than 3T. The following instructions assume 3T (0.3T for table space, 2.7T for images).

One builds the structure from the bottom up.

For WP-MIRROR 0.6 and later versions, the author recommends the structure shown in Figure 3.1, Recommended Disk Configuration of WP-MIRROR 0.6 and later versions..

- mysql on ReiserFS over LVM2 over LUKS over RAID over whole disks
- images on ReiserFS over LVM2 over LUKS over RAID over whole disks

For WP-MIRROR 0.5 and earlier versions, the author recommends the structure shown in Figure 3.2, Recommended Disk Configuration of WP-MIRROR 0.5 and earlier versions.. Note that the ibdata0 table space is stored directly on a raw partition:

- ibdata0 on LVM2 over LUKS over RAID over whole disks
- images on ReiserFS over LVM2 over LUKS over RAID over whole disks

Figure 3.1: Recommended Disk Configuration of WP-MIRROR 0.6 and later versions.

Table 3.23: History of Configuring Whole Disks for Desktop

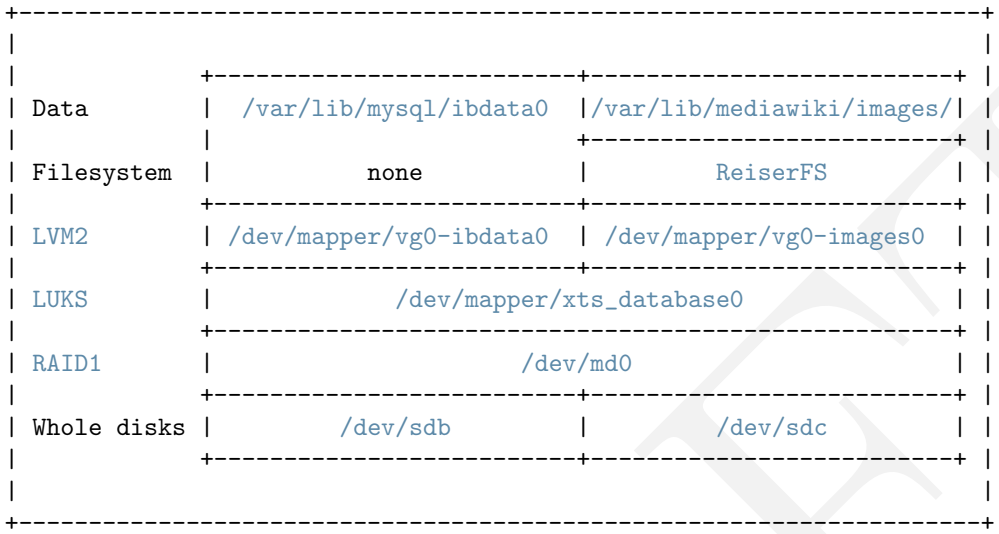
Version	Configuration
≥ 0.1	user configuration required (see below)

3.3.2.7.1 Whole Disks First, open the case of your desktop PC, and follow the case manual's and motherboard manual's instructions for installing the pair of 3T HDDs.

Second, boot up and check that the fan is working. Allow 1ks (20 min) for temperatures to settle, then check HDD temperatures by executing:

```
root-shell# aptitude install hddtemp
root-shell# hddtemp /dev/sd[a-c]
```

Figure 3.2: Recommended Disk Configuration of WP-MIRROR 0.5 and earlier versions.



Third, check each new disk for bad blocks by executing:

```
root-shell# badblocks -c 102400 -o /tmp/badblocks.txt -s -t random -v -w /dev/sdb
root-shell# badblocks -c 102400 -o /tmp/badblocks.txt -s -t random -v -w /dev/sdc
```

Each of the above `badblocks` commands should take about 75ks (21 hours) to run.

Finally, if you find even one bad block, return the disk(s) and get new ones. Beware, many disks are 'refurbished', meaning, a previous user had problems and returned them—and now they are back on the shelf at your computer store.

Note that there will be no need to partition your new disks. So there is no need to run `cfdisk`.

Table 3.24: History of Configuring `mdadm` for Desktop

Version	Configuration
≥ 0.1	user configuration required (see below)

3.3.2.7.2 `RAID` We use `RAID1` (mirrored) as the next layer.

First, create the mirror by executing:

```
root-shell# aptitude install mdadm
root-shell# mdadm --create /dev/md0 --verbose --level=1 --raid-devices=2 \
--metadata=1.0 --bitmap=internal --name=database0 --auto=md /dev/sdb /dev/sdc
```

where

- `level=1` means `RAID1` (mirrored).
- `raid-devices=2` means the raid set will have two active disks.
- `metadata=1.0` means a version-1 formatted superblock will be placed at the end of each disk. If the array will be larger than 2T, then the metadata=0.90 (default) will not do.
- `bitmap=internal` means a write-intent log is stored near the superblock. Resync is greatly optimized. A full resync takes about 100ks (1 day) if there is no bitmap.

- name=database0 means the raid array will have a name. This is possible with the version-1 format superblock.
- auto=md means the array will be non-partitionable.
- /dev/sdb and /dev/sdc are the disks.

Second, check that the RAID array is healthy by executing:

```
root-shell# cat /proc/mdstat
root-shell# mdadm --detail /dev/md0
```

Finally, set up the configuration file by executing

```
root-shell# mdadm --detail --scan
root-shell# mdadm --detail --scan >> /etc/mdadm/mdadm.conf
```

Table 3.25: History of Configuring LUKS for Desktop

Version	Configuration
≥ 0.1	user configuration required (see below)

3.3.2.7.3 LUKS Encryption is unecessary for public information like a set of wikipedias. However, InnoDB stores all tables in a single partition. This means, if you later decide to install any other database-backed service (e.g. drupal, sendmail, etc.) then its data would be stored together on the same partition. This would be a security hole.

First, create the encrypted block device by executing:

```
root-shell# aptitude install cryptsetup
root-shell# cryptsetup --cipher aes-xts-plain --key-size 512 luksFormat /dev/md0
```

You will be asked for a LUKS passphrase. Choose a phrase that you can remember, but that others will not guess. As a precaution, you may wish to write it down and store it in a safe.

```
root-shell# cryptsetup luksOpen /dev/md0 xts_database0
```

You can (optionally) add up to eight keys to the LUKS partition.

Third, in order to automate the boot process, store a second key in a file in the /etc/keys/ directory, and set the permissions to deny non-root access, by executing:

```
root-shell# emacs /etc/keys/luks_key_md0
My V3ry S3cr3t P@ssphr@s3
root-shell# chmod 600 /etc/keys/luks_key_md0
root-shell# cryptsetup luksAddKey /dev/md0 /etc/keys/luks_key_md0
```

Take a look at the meta-data by executing:

```
root-shell# cryptsetup luksDump /dev/md0
```

Fourth, to automatically unlock this partition during the cryptdisks-early phase of the boot process, edit /etc/crypttab to read:

```
#<target dev> <source dev> <key file>             <options>
xts_database0 /dev/md0     /etc/keys/luks_key_md0 luks,tries=3
```

Finally, reboot to test.

Design note.
Keeping the passphrase on /dev/sda is not a security hole, if you also encrypt /dev/sda (which you should).

Table 3.26: History of Configuring LVM2 for Desktop

Version	Configuration
≥ 0.1	user configuration required (see below)

3.3.2.7.4 LVM2 Next we wish to partition the LUKS partition: 0.3T for ibdata0, and the rest (2.7T) for images.

```
root-shell# aptitude install lvm2
root-shell# pvcreate /dev/mapper/xts_database0
root-shell# vgcreate vg0 /dev/mapper/xts_database0
root-shell# vgdisplay vg0
root-shell# lvcreate --extents 75000 --name ibdata0 vg0
root-shell# lvcreate --extents 100%FREE --name images0 vg0
root-shell# vgdisplay vg0
root-shell# lvdisplay vg0
```

Extents are 4MB in size, so the logical volume /dev/mapper/vg0-ibdata0 is about 300G (75,000 x 4MB), while /dev/mapper/vg0-images0 occupies the rest of the LUKS partition.

Table 3.27: History of Configuring File System for Desktop

Version	Configuration
≥ 0.6	format /dev/vg0/ibdata0 and /dev/vg0/images0 (see below)
≤ 0.5	format /dev/vg0/images0, leave /dev/vg0/ibdata0 raw (see below)

3.3.2.7.5 File System First, set up a modern journalling file system. There are several choices. Some, however, you may wish to avoid:

- btrfs is under development (underlying disk format may change),
- ext2 has no journal, and the file reference counter is only two bytes,
- ext4 has poor performance,
- reiser4 is under development, and
- zfs is encumbered.

For now, the author recommends ReiserFS.
For WP-MIRROR 0.6 and later versions, also format the second partition by executing:

```
root-shell# aptitude install reiserfsprogs
root-shell# mkfs.reiserfs /dev/vg0/ibdata0
root-shell# mkdir -p /database0/mysql
root-shell# mkfs.reiserfs /dev/vg0/images0
root-shell# mkdir -p /database0/images
```

Second, to automate mounting during the boot process, edit /etc/fstab by appending the lines:

```
root-shell# emacs /etc/fstab
/dev/vg0/ibdata0 /database0/mysql  reiserfs defaults 0 2
/dev/vg0/images0 /database0/images reiserfs defaults 0 2
```

Third, test the mount procedure by executing:

```
root-shell# mount /database0/mysql
root-shell# chown mysql:mysql /database0/images
root-shell# mount /database0/images
root-shell# chown www-data:www-data /database0/images
```

Finally, move the database to the new partition and establish a link by executing:

```
root-shell# cd /var/lib/
root-shell# mv /var/lib/mysql/* /database0/mysql/.
root-shell# rmdir mysql
root-shell# ln -s /database0/mysql/ mysql
root-shell# chown -R mysql:mysql /database0/mysql/
```

For WP-MIRROR 0.5 and earlier versions, leave the `/dev/vg0/ibdata0` partition unformatted (raw).

Table 3.28: History of Configuring Raw Partition for Desktop

Version	Configuration
$\geq$ 0.6	obsolete, no user configuration is necessary
$\leq$ 0.5	user configuration required (see below)

3.3.2.7.6 Raw Partition For WP-MIRROR 0.6 and later versions, there is no raw partition.

For WP-MIRROR 0.5 and earlier versions, there is nothing to do just yet. We will discuss this when we Customize the InnoDB Storage Engine (see below).

3.3.3 Plan Your Database Management System

It is simplest if you start with a clean database.

3.3.3.1 Secure the Database Management System

User configuration is optional (see §3.2.2.1, Secure the Database Management System above).

3.3.3.2 Load the DBMS time_zone Tables

Automated. No user configuration is necessary (see §3.2.2.2, Load the DBMS time_zone Tables above).

3.3.3.3 Customize the InnoDB Storage Engine

Table 3.29: History of Configuring InnoDB Storage Engine for Desktop

Version	Configuration
$\geq$ 0.6	automated, no user configuration is necessary
$\leq$ 0.5	user configuration required (see below)

For WP-MIRROR 0.6 and later versions, no user configuration is necessary (see §3.2.2, Plan Your Database Management System above).

For WP-MIRROR 0.5 and earlier versions, customization is done as follows:

First, copy /usr/share/doc/wp-mirror/examples/wp-mirror_desktop.cnf to /etc/mysql/conf.d/.:

```
root-shell# cd /usr/share/doc/wp-mirror/examples/
root-shell# cp wp-mirror_desktop.cnf /etc/mysql/conf.d/.
root-shell# cd /etc/mysql/conf.d/
```

Second, delete the existing log files and table space, and restart MySQL by executing:

```
root-shell# /etc/init.d/mysql stop
root-shell# rm /var/lib/mysql/ib*
root-shell# /etc/init.d/mysql start
```

Third, edit /etc/mysql/conf.d/wp-mirror_desktop.cnf to replace the lines:

```
innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mnewraw
#innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mraw
```

with:

```
#innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mnewraw
innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mraw
```

Fourth, restart MySQL.

```
root-shell# /etc/init.d/mysql restart
```

Finally, confirm that it retains data.

Design note.

WARNING: Creating a new table space on the raw partition is a two step process.

1) In /etc/mysql/conf.d/custom.cnf the lines

```
innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mnewraw
#innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mraw
```

create the new table space—and will do so again and again (with data loss) every time you restart mysqld.

2) You must comment the first line, and uncomment the second line, so that they read:

```
#innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mnewraw
innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mraw
```

and restart mysqld.

Alternatively, you can try the 'cut-and-paste' approach:

First, edit /etc/mysql/conf.d/wp-mirror_desktop.cnf to read:

```
[mysqld]
default-time-zone = UTC
default-storage-engine = innodb
character-set-server = utf8
collation-server = utf8_general_ci

# put table space on a disk separate from the log files
innodb_data_home_dir =

# put table space on a raw partition
innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mnewraw
#innodb_data_file_path=/dev/mapper/vg0-ibdata0:300000Mraw
innodb_file_format_check

# increase buffer pool for speed (but fit within 4000M hugepages)
innodb_buffer_pool_size =3998M     # default 128M

# increase size of log files for speed
innodb_log_file_size =250M         # default=5M.

# for Durability (the 'D' in 'ACID compliance')
innodb_flush_log_at_trx_commit = 2 # default 1. (2=write log file once per sec)
sync_binlog = 0                    # default 0.
max_allowed_packet = 64M           # default 16M.

# Enable large page support.  InnoDB will use it automatically
# for its buffer_pool and additional_memory_pool
large-pages

[mysql]
default-character-set = utf8
max_allowed_packet = 64M           # default 16M.
```

Second, delete the existing log files and `table space`, and restart `MySQL` as above.
Third, edit `/etc/mysql/conf.d/wp-mirror_desktop.cnf` as above.
Fourth, restart `MySQL` as above.

Design note. The `large-pages` setting tells `InnoDB` to make use of `hugepages`, which are discussed in §3.3.4, Plan Your DRAM next.

3.3.4 Plan Your DRAM

All major CPU's organize memory into pages (e.g. for Intel the default page size is 4KB). All major operating systems (OS) have a memory management (MM) algorithm that swaps the least recently used (LRU) pages out to disk.

However, `MySQL`'s `InnoDB` storage engine has its own MM algorithm and uses a page size of 16KB.

There is no need for the OS to swap pages used by `InnoDB`, and actually there is harm in letting the OS do so. Due to the difference in page size (OS 4KB vs. `InnoDB` 16KB), the OS swap algorithm could inadvertently break up `InnoDB` pages whenever other memory intensive processes (e.g. `gm`) run.

Fortunately there is a way to disable OS swapping. Most modern CPU's offer pages of various sizes. Intel offers `hugepages` of 2MB or 4MB (depending on the CPU model), which the OS swap algorithm will leave alone.

To run efficiently, InnoDB needs to have a large innodb-buffer-pool in DRAM, and it is best to store it on hugepages. This is why you should consider buying another 4G of DRAM and configuring that space as hugepages. Total installed DRAM should be at least 6G (more is better).

Design note. WP-MIRROR will not let you start mirroring large wikipedias without first finding adequate physical memory (at least 4G default).

3.3.4.1 Hugepages

Table 3.30: History of Configuring Hugepages for Desktop

Version	Configuration
≥ 0.6	allocate 1G, user configuration required (see below)
≤ 0.5	allocate 4G, user configuration required (see below)

For WP-MIRROR 0.6 and later versions, allocate 1G of DRAM as hugepages by configuring /etc/sysctl.conf by adding the lines:

```
root-shell# emacs /etc/sysctl.conf
vm.nr_hugepages = 512
vm.hugetlb_shm_group = 1001
kernel.shmmax = 1073741824
kernel.shmall = 262144
root-shell# sysctl -p
```

Confirm that hugepages have been allocated by executing:

```
shell$ cat /proc/meminfo | grep Huge
```

For WP-MIRROR 0.5 and earlier versions, allocate 4G of DRAM as hugepages by configuring /etc/sysctl.conf by adding the lines:

```
root-shell# emacs /etc/sysctl.conf
vm.nr_hugepages = 2048
vm.hugetlb_shm_group = 1001
kernel.shmmax = 4294967296
kernel.shmall = 1048576
root-shell# sysctl -p
shell$ cat /proc/meminfo | grep Huge
```

Confirm allocation as above.

If the hugepages were *not* allocated, then it may be necessary to reboot the system in order to allocate the hugepages. Failure to allocate happens when the desktop has been running for some time and the DRAM is fully allocated to other processes.

3.3.4.2 Permissions

Set permissions so that MySQL can access the hugepages by configuring /etc/group by appending the line:

```
hugepage:x:1001:mysql
```

Table 3.31: History of Configuring Hugepages Permissions for Desktop

Version	Configuration
≥ 0.1	user configuration required (see below)

Table 3.32: History of Configuring InnoDB Buffer Pool for Desktop

Version	Configuration
≥ 0.6	automated, no user configuration is necessary
≤ 0.5	user configuration required (see below)

3.3.4.3 Buffer Pool

Automatic configuration of the buffer pool was introduced with WP-MIRROR 0.6. No user configuration is necessary.

For WP-MIRROR 0.5 and earlier versions, you must manually configure mysql as follows:

Tell mysqld to use the hugepages and set InnoDB's buffer pool to fit within the hugepages by configuring /etc/mysql/conf.d/custom.cnf by appending the lines:

```
[mysqld]
innodb_buffer_pool_size=3998M #default=8M.
large-pages
```

This was done above in §3.3.3.3, Customize the InnoDB Storage Engine.

3.3.4.4 Process Limits

Table 3.33: History of Configuring Process Limits for Desktop

Version	Configuration
≥ 0.5	obsolete, no user configuration is necessary
≤ 0.4	user configuration required (see below)

For WP-MIRROR 0.5 and later versions, configuring mysqld process limits is no longer needed. No user configuration is necessary.

For WP-MIRROR 0.4 and earlier versions, relax mysqld process limits by configuring /usr/bin/mysqld_safe by appending the line:

```
ulimit -l unlimited
```

3.3.5 Plan Your Internet Access

No user configuration is necessary (see §3.3.5, Plan Your Internet Access above).

3.3.6 Plan Your MediaWiki

No user configuration is necessary (see §3.3.6, Plan Your MediaWiki above).

3.3.7 Plan Your Image Processing

No user configuration is necessary (see §3.3.7, Plan Your Image Processing above).

3.3.8 Plan Your Virtual Host

No user configuration is necessary (see §3.2.7, Plan Your Virtual Host above).

3.3.9 Plan Your Caching Web Proxy

If your Internet traffic must go through a caching web proxy (e.g. `polipo`), you should let your system administrator know what you are up to before you stuff his cache. You may expect problems when you Download Large Dump Files and when you Download Millions of Image Files (see below).

3.3.9.1 Download Large Dump Files

The en wikipedia dump file is about 10G (as of 2013). Some caching web proxies (e.g. `polipo`) crash if a file exceeds available memory. A crash blocks web traffic for everyone—not a great way to make friends, even if productivity soars. In this case, the dump file must be manually copied into WP-MIRROR's working directory, `/var/lib/mediawiki/images/wp-mirror/`. For example, take a laptop off-site, download the dump file, bring the laptop back on-site, and then `scp` the dump file to the working directory. Better yet, if you are the system administrator, `ssh` to your proxy, run `wget` with the `--no-proxy` option, then `scp` the downloaded dump file to the WP-MIRROR working directory like so:

```
root-shell# cd /var/lib/mediawiki/images/wp-mirror/
root-shell# ssh my-proxy
my-proxy# wget --no-proxy http://dumps.wikimedia.org/enwiki/yyyymmdd/enwiki-
yyyymmdd-pages-articles.xml.bz2
my-proxy# exit
root-shell# scp -p my-proxy:enwiki-yyyymmdd-pages-articles.xml.bz2 .
```

Then restart WP-MIRROR. It will find the dump file, and handle matters from there.

3.3.9.2 Download Millions of Image Files

Some caching proxies might not have disk space adequate for caching nearly 3T of images. Usually `/var/` is mounted on its own partition (and this is highly recommended); and usually a caching web proxy keeps its cache on that partition (e.g. `/var/cache/polipo/`). However, if the `/var/` partition fills up, then no other process will be able to write to that partition. For example,

- **rsyslogd**: `rsyslogd` writes its system log files under `/var/log/`. While the loss of log files is not too tragic and would pass unnoticed by most users, it would draw the concern of the system administrator.
- **sendmail**: If, moreover, the system administrator had the poor judgement to install a mail transfer agent (MTA) (e.g. `exim`, `sendmail`, etc.) on the same host, then stuffing `/var/` would hold up the mails, and the loss of service would be felt widely. Again, not a great way to make friends.

In this case, the cached images should be removed each day (perhaps by using a daily `cron` job) like this

```
root-shell# killall -USR1 polipo
root-shell# rm -f /var/cache/polipo/dumps.wikimedia.org/*
root-shell# rm -f /var/cache/polipo/upload.wikimedia.org/*
root-shell# killall -USR2 polipo
```

3.3.9.3 Bypass Unreliable Caching Web Proxy

If the caching web proxy is unable to handle your traffic, it may be best to by-pass the caching web proxy altogether. One good method is to use the port-forwarding feature of `SSH` to connect to a box that is outside of the network served by the caching web proxy. But try working with your system administrator first.

3.3.9.4 Configure Utilities for Use with Caching Web Proxy

Utility	Configuration
`bash`	See §3.2.8.1, Configure `bash` for Use with Caching Web Proxy above.
`cURL`	See §3.2.8.2, Configure `cURL` for Use with Caching Web Proxy above.
`wget`	See §3.2.8.3, Configure `wget` for Use with Caching Web Proxy above.
browsers	See §3.2.8.4, Configure Browser for Use with Caching Web Proxy above.

Chapter 4

WP-MIRROR Programs

4.1 WP-MIRROR in Mirror Mode

WP-MIRROR can run either in mirror mode or in monitor mode. Mirror mode can be invoked from your shell as follows:

```
root-shell# wp-mirror --mirror
```

WP-MIRROR in mirror mode, supports the options listed in Table 4.1, WP-MIRROR Mirror Mode Options Reference.

4.2 WP-MIRROR in Monitor Mode

WP-MIRROR can run in either mirror mode or in monitor mode. Monitor mode can be invoked from your shell as follows:

```
root-shell# wp-mirror --monitor
```

WP-MIRROR in monitor mode, supports the options listed in Table 4.3, WP-MIRROR Monitor Mode Options Reference. In monitor mode, WP-MIRROR displays the state of each mirror in a mirror farm. This display can be done in three ways.

4.2.1 WP-MIRROR in Monitor Mode with GUI Display

Monitor mode with GUI display can be invoked by the command:

```
root-shell# wp-mirror --gui
```

A screen shot of the GUI display is shown in Figure 4.1, WP-MIRROR in Monitor Mode with GUI Display.

4.2.2 WP-MIRROR in Monitor Mode with Screen Display

Monitor mode with screen display can be invoked by the command:

```
root-shell# wp-mirror --screen
```

A screen shot of the screen display is shown in Figure 4.2, WP-MIRROR in Monitor Mode with Screen Display.

Table 4.1: WP-MIRROR Mirror Mode Options Reference

Runtime Options	Description	Default	Intr	Rem
--add {*wiki*|*language-code*|*project*|all}	add given language to mirror, and exit.		0.5	
--copyright	display copyright, and exit.		0.1	
--debug	verbose output.		0.1	
--delete {*wiki*|*language-code*|*project*|all|template}	delete files and state information (but keep database and images), for given language, and exit.		0.4	
--drop {*wiki*|*language-code*|*project*|all|template}	drop database, delete files and state info (but keep images), for given language, and exit.		0.4	
--dump {*wiki*|*language-code*|*project*|all|template}	dump database, to file *xx*wiki.sql in working directory (where *xx* stands for the language code), for given language, and exit. If the language-code is template, then the empty database wikidb is dumped to database_farm.sql.		0.5	
--help	display help, and exit.		0.1	
--info	display configuration parameters, and exit.		0.5	
--mirror	force mirror mode (overrides any monitor mode options or defaults).		0.1	
--profile [*run-number*]	display real-time profile for mirror-building run(s), and exit.		0.6	
--restore-default	drop all databases and files (except images), and start over (dangerous).		0.4	
--update {*wiki*|*language-code*|*project*|all|template}	update database, to recent MediaWiki schema, for given language, and exit.		0.5	
--version	display version and copyright, and exit.		0.1	

Table 4.2: WP-MIRROR Mirror Mode Options Reference (Obsolete)

Runtime Options	Description	Default	Intr	Rem
--delete-dump *language*	delete dump file and state information for given language, and exit.		0.1	0.4

4.2.3 WP-MIRROR in Monitor Mode with Text Display

Monitor mode with screen display can be invoked by the command:

```
root-shell# wp-mirror --text
```

A screen shot of the text display is shown in Figure 4.3, WP-MIRROR in Monitor Mode with Text Display.

Table 4.3: WP-MIRROR Monitor Mode Options Reference

Runtime Options	Description	Default	Intr	Rem
`--gui`	operate monitor in GUI mode.		0.1	
`--monitor`	force monitor mode (if no other process in mirror mode).	`gui`	0.1	
`--screen`	operate monitor in screen mode.		0.1	
`--text`	operate monitor in text mode.		0.1	

Figure 4.1: WP-MIRROR in Monitor Mode with GUI Display

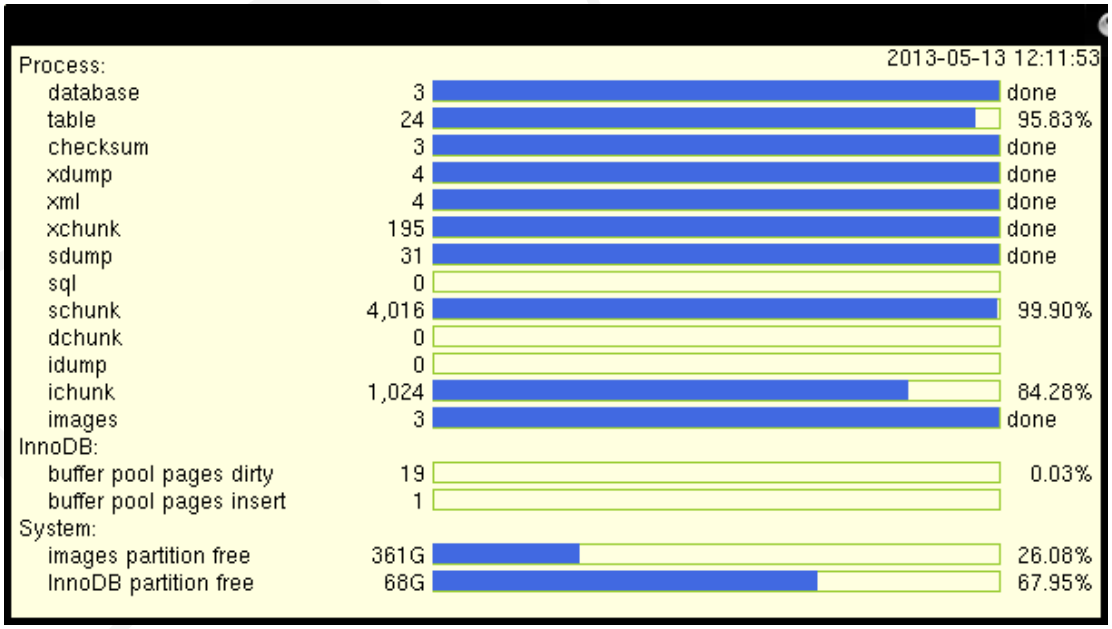


Figure 4.2: WP-MIRROR in Monitor Mode with Screen Display

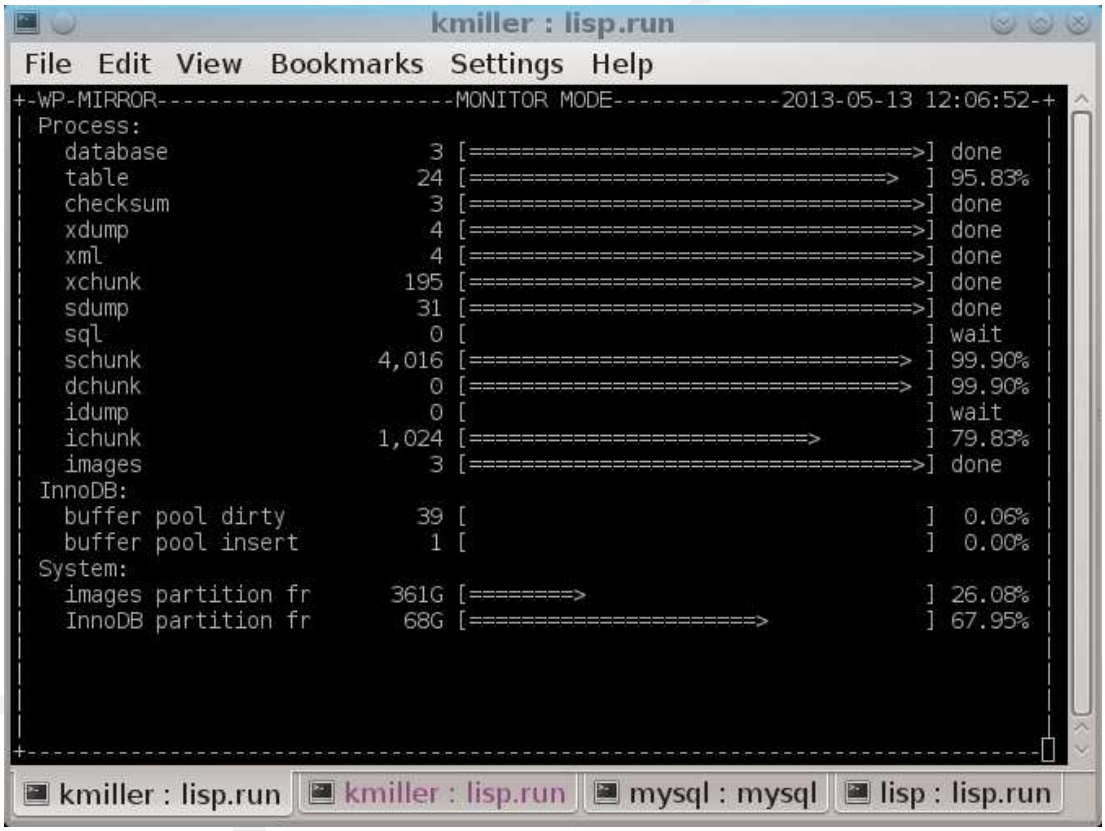
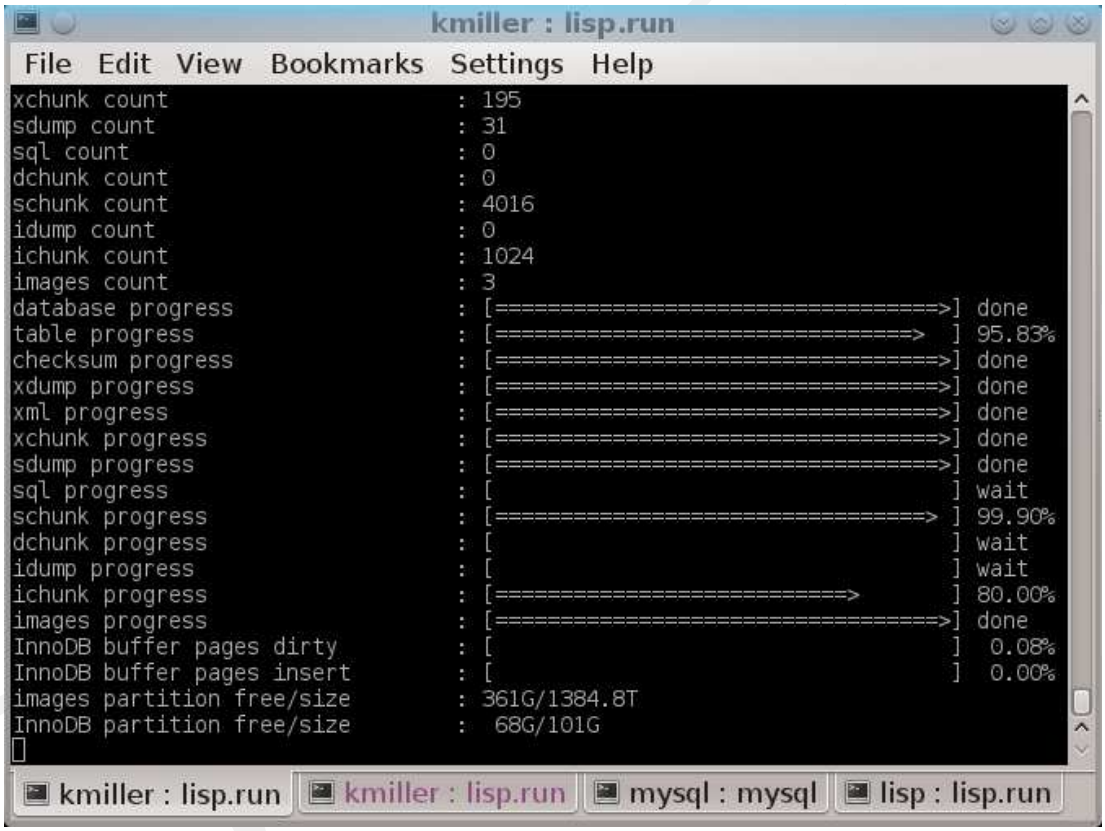


Figure 4.3: WP-MIRROR in Monitor Mode with Text Display



Chapter 5

How WP-MIRROR Works

This chapter describes in great detail, how WP-MIRROR works. This is meant as an aid to developers, but may be also of use to users who are interested in the how and why of each task that WP-MIRROR performs.

5.1 How `--mirror` Works

5.1.1 Start

Processing begins with the command:

```
root-shell# wp-mirror --mirror
```

5.1.1.1 `process-command-line-arguments-or-die`

WP-MIRROR first reads the command-line arguments. If there an error is found, you may expect:

```
root-shell# wp-mirror --foo
Error: command line ...
unexpected option                : (foo)                                [fail]
For help, please try:
    $ wp-mirror --help
```

5.1.2 Initializing

```
root-shell# wp-mirror --mirror
-----initializing-begin-----------------------
[ ok ]clear-pidfile
[ ok ]set-pidfile
[info]set mode of operation to: FIRST-MIRROR
[ ok ]log-start
[ ok ]assert-clisp-features-p
[ ok ]assert-utilities-p
[ ok ]assert-images-directory-or-create-p
[ ok ]assert-working-directory-or-create-p
[ ok ]assert-dbms-mysql-p
[ ok ]assert-dbms-mysql-install-db-p
[ ok ]assert-dbms-mysql-config-debian-p
[ ok ]assert-dbms-credentials-debian-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-debian-p
[ ok ]assert-dbms-time-zone-or-load
[ ok ]assert-configuration-files-or-restore-default
[ ok ]process-configuration-files-or-die
[ ok ]put-parameters
```

These messages show the progress of initializing. We now describe each of them:

5.1.2.1 clear-pidfile

```
root-shell# wp-mirror --mirror
...
[ ok ]clear-pidfile
```

WP-MIRROR establishes whether or not it is the only instance of WP-MIRROR running. There are a couple of reasons for doing this:

- cron. One is to be sure that the weekly cron job starts only if there are no other instances of WP-MIRROR running at that time. When mirroring large wikipedias, it can happen that last week's instance is still running.
- **Concurrency**. Another is to distinguish between the instances running as :first-mirror and any others running as :next-mirror. Tasks that cannot be run concurrently are allowed to run only on the :first-mirror.

Determining whether or not there are other instances of WP-MIRROR, is done by looking for the file /var/run/wp-mirror.pid. This is called the PID file, and it contains the process ID assigned by the operating system. As a precaution, if WP-MIRROR lacks read-write privilege for /var/run/wp-mirror.pid, it generates an error message and exits. Now, if a PID file is found for which there is no corresponding process, then probably a previous process was terminated prematurely, and so the stale PID file is cleared.

5.1.2.2 set-pidfile

```
root-shell# wp-mirror --mirror
...
[ ok ]set-pidfile
```

When WP-MIRROR runs in :first-mirror mode, it sets a PID file at /var/run/wp-mirror.pid.

5.1.2.3 set mode of operation to: FIRST-MIRROR

```
root-shell# wp-mirror --mirror
...
[info]set mode of operation to: FIRST-MIRROR
```

Every WP-MIRROR instance must determine its main mode of operation, and this is announced to the user with one of these messages:

```
[info]set mode of operation to: ADD
[info]set mode of operation to: DELETE
[info]set mode of operation to: DROP
[info]set mode of operation to: DUMP
[info]set mode of operation to: FIRST-MIRROR
[info]set mode of operation to: NEXT-MIRROR
[info]set mode of operation to: MONITOR
[info]set mode of operation to: RESTORE-DEFAULT
[info]set mode of operation to: UPDATE
```

Only the instance running as :first-mirror may set the PID file.

Which mode a particular instance of WP-MIRROR chooses depends upon: 1) whether or not there is a valid PID file, and 2) the command-line options, if any. There are several cases to consider, and these are tabulated in Table 5.1, WP-MIRROR PID File Logic. For this purpose, the command-line options --gui, --screen, and --text, all count as the --monitor option.

Table 5.1: WP-MIRROR PID File Logic

Case	Process exists	Cmd-line options	Functions	*main-mode*
1	N	none	clear-pid, set-pidfile	:first-mirror
2	N	--mirror	clear-pid, set-pidfile	:first-mirror
3	N	--monitor	none	:monitor
4	Y	none	none	:monitor
5	Y	--mirror	none	:next-mirror
6	Y	--monitor	none	:monitor
7	N/A	--add	none	:add
8	N/A	--delete	none	:delete
9	N/A	--drop	none	:drop
10	N/A	--dump	none	:dump
11	N/A	--info	none	:info
12	N/A	--profile	none	:time
13	N/A	--restore-default	none	:restore-default
14	N/A	--update	none	:update

When an instance of WP-MIRROR runs in :next-mirror mode, that means multiple instances of WP-MIRROR are running concurrently. Most mirror-building tasks can be executed concurrently (and to great advantage when building a large wikipedia on a desktop). However, there is a small number of tasks that can be run safely only by the :first-mirror instance.

5.1.2.4 log-start

```
root-shell# wp-mirror --mirror
...
[ ok ]log-start
```

WP-MIRROR turns on logging. Messages are written to /var/log/wp-mirror.log.

5.1.2.5 assert-clisp-features-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-clisp-features-p
```

WP-MIRROR is written in Common Lisp (clisp 2.48). WP-MIRROR tests (asserts) that all the required libraries are loaded. For example, the CLX library is needed for running WP-MIRROR in --gui mode.

5.1.2.6 assert-utilities-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-utilities-p
```

WP-MIRROR relies on many utilities to do most of the work. WP-MIRROR asserts that the executable files exist.

5.1.2.7 assert-images-directory-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-images-directory-or-create-p
```

WP-MIRROR creates a directory tree in which MediaWiki stores downloaded image files. This directory is /var/lib/mediawiki/images/ (default).

5.1.2.8 assert-images-math-directory-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-images-math-directory-or-create-p
```

The MediaWiki extension Math.php converts math equations from LaTeX format into PNG image files.

WP-MIRROR creates a directory tree in which Math.php stores the image files it creates. This directory is /var/lib/mediawiki/images/*project*/*language-code*/math/ (default).

5.1.2.9 assert-images-thumb-directory-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-images-thumb-directory-or-create-p
```

WP-MIRROR creates a directory tree in which MediaWiki stores resized images (thumbs). This directory is /var/lib/mediawiki/images/*project*/*language-code*/thumb/ or /var/lib/mediawiki/images/wikipedia/commons/thumb/ (default).

5.1.2.10 assert-images-tmp-directory-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-images-tmp-directory-or-create-p
```

WP-MIRROR creates a directory which the MediaWiki extension Math.php uses as a scratch space, while converting math equations from LaTeX format into PNG image files. This directory is /var/lib/mediawiki/images/*project*/*language-code*/tmp/ (default).

5.1.2.11 assert-working-directory-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-working-directory-or-create-p
```

WP-MIRROR creates a directory in which all its working files will be kept (checksum, ichunk, idump, schunk, sdump, sql, xchunk, xdump, xml, etc.). This directory is /var/lib/mediawiki/images/wp-mirror/ (default).

5.1.2.12 assert-dbms-mysql-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-mysql-p
```

MediaWiki is able to use: MySQL, postgres, sqlite, mssql, and ibm_db2. WP-MIRROR, however, is only able to use the MySQL database management system (DBMS). If MySQL is not available, then WP-MIRROR exits.

5.1.2.13 assert-dbms-mysql-install-db-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-mysql-install-db-p
```

WP-MIRROR looks for the /var/lib/mysql/ directory. If it does not exist, WP-MIRROR attempts to install it by executing:

```
root-shell# mysql_install_db --user=mysql
```

If /var/lib/mysql/ still does not exist, then WP-MIRROR exits. In this event, please reinstall MySQL manually (see §3.2.2.4, MySQL Redux).

5.1.2.14 assert-dbms-mysql-config-debian-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-mysql-config-debian-p
```

WP-MIRROR looks for Debian's database credentials, which are stored in /etc/mysql/debian.cnf. Debian's database account has root privileges.

5.1.2.15 assert-dbms-credentials-debian-or-scrape-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-credentials-debian-or-scrape-p
```

WP-MIRROR reads /etc/mysql/debian.cnf, and parses the file for Debian's MySQL user account and password. The file looks something like:

```
# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host     = localhost
user     = debian-sys-maint
password = abcdefghijklmnop
socket   = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
host     = localhost
user     = debian-sys-maint
password = abcdefghijklmnop
socket   = /var/run/mysqld/mysqld.sock
basedir  = /usr
```

5.1.2.16 assert-dbms-connect-with-credentials-debian-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-connect-with-credentials-debian-p
```

WP-MIRROR tests (asserts) if it can use the Debian database credentials to access MySQL. If it cannot, WP-MIRROR exits.

5.1.2.17 assert-dbms-time-zone-or-load

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-time-zone-or-load
```

WP-MIRROR checks if the MySQL time zone tables are populated. If they are not, the time zone data will be loaded from /usr/share/zoneinfo/, and you will see messages like:

```
root-shell# wp-mirror --mirror
...
[....]assert-dbms-time-zone-or-load
looking for time-zone             : UTC                                [fail]
Warning: Unable to load '/usr/share/zoneinfo/Asia/Riyadh87' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/Asia/Riyadh88' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/Asia/Riyadh89' as time zone. Skipping it.
...
[ ok ]assert-dbms-time-zone-or-load
```

5.1.2.18 assert-configuration-files-or-restore-default

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-configuration-files-or-restore-default
```

If all the configuration files are in place, you will see the above.

If any configuration files are missing, you will see additional messages. If all configuration files are missing, which can happen after you use the --restore-default option, you will see:

```
root-shell# wp-mirror --mirror
...
[....]assert-configuration-files-or-restore-default
[info]restoring default    : /etc/mediawiki/LocalSettings.php
[info]restoring default    : /etc/mediawiki/LocalSettings_wpmirror.php
[info]restoring default    : /etc/mysql/conf.d/wp-mirror.cnf
[info]restoring default    : /etc/wp-mirror/default.conf
[info]restoring default    : /etc/wp-mirror/local.conf
[info]restoring default    : /usr/share/mediawiki/maintenance/database_farm.sql
[info]restoring default    : /var/lib/mediawiki/favicon.ico
[info]restoring default    : /var/lib/mediawiki/wp-mirror.png
[ ok ] Stopping MySQL database server: mysqld.
[ ok ] Starting MySQL database server: mysqld . . ..
[info] Checking for tables which need an upgrade, are corrupt or were
not closed cleanly..
[ ok ]assert-configuration-files-or-restore-default
```

5.1.2.19 process-configuration-files-or-die

```
root-shell# wp-mirror --mirror
...
[ ok ]process-configuration-files-or-die
```

WP-MIRROR reads the local configuration file /etc/wp-mirror/local.conf. All WP-MIRROR parameters have default values, which may be seen in /etc/wp-mirror/default.conf. Parameter values set in /etc/wp-mirror/local.conf override the default values.

The parameter that most users will want to edit is,

```
(defparameter *mirror-language-code-list* '("simple"))
```

Users should make their configurations in /etc/wp-mirror/local.conf only. The /etc/wp-mirror/default.conf is for reference, and should never be edited.

5.1.2.20 put-parameters

```
root-shell# wp-mirror --mirror
...
[ ok ]put-parameters
```

WP-MIRROR writes all parameter values to the log file /var/log/wp-mirror.log. This information is used for debugging.

5.1.3 Asserting Prerequisite Software

```
...
-----asserting-prerequisite-software-begin----
[ ok ]assert-dbms-accounts-or-create-p
[ ok ]assert-dbms-credentials-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
[ ok ]assert-dbms-grant-for-wikiadmin-p
[ ok ]assert-dbms-connect-with-credentials-wikiuser-p
[ ok ]assert-dbms-grant-for-wikiuser-p
[ ok ]warn-if-dbms-root-account-has-no-password
[ ok ]warn-if-dbms-has-anonymous-user-account
[ ok ]warn-if-dbms-has-root-accounts-accessible-from-outside-localhost
[ ok ]warn-if-dbms-has-test-database
[ ok ]assert-database-wpmirror-or-create-p
[ ok ]assert-database-template-and-wikidb-p
[ ok ]assert-mediawiki-localsettings-p
[ ok ]assert-mediawiki-localsettings-account-p
[ ok ]assert-mediawiki-localsettings-wpmirror-p
[ ok ]assert-mediawiki-localsettings-image-p
[ ok ]assert-mediawiki-localsettings-tidy-p
[ ok ]assert-mediawiki-favicon-p
[ ok ]assert-mediawiki-logo-p
[ ok ]assert-mediawiki-dbms-credentials-p
[ ok ]assert-concurrency-limit-xchunk-p
[ ok ]assert-virtual-host-p
[ ok ]assert-virtual-host-name-resolution-p
[ ok ]warn-if-detect-proxy
```

These messages show the progress of asserting software prerequisites. We now describe each of them:

5.1.3.1 assert-dbms-accounts-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-accounts-or-create-p
```

WP-MIRROR asserts:

- the DBMS has an account for wikiadmin,
- the DBMS has an account for wikiuser,
- the file /etc/mediawiki/LocalSettings_account.php exists.

If the assertion fails, WP-MIRROR then:

- performs a DROP USER on any old wikiadmin and wikiuser accounts,
- generates random passwords,
- performs a CREATE USER to make new wikiadmin and wikiuser accounts, and
- writes the credentials to /etc/mediawiki/LocalSettings_account.php.

5.1.3.2 assert-dbms-credentials-or-scrape-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-credentials-or-scrape-p
```

WP-MIRROR reads /etc/mediawiki/LocalSettings_account.php and parses it to extract database credentials. The file looks something like:

```
root-shell# cat /etc/mediawiki/LocalSettings_account.php
<?php
# Automatically generated by WP-MIRROR for mediawiki scripts. DO NOT TOUCH!
$wgDBadminuser      = 'wikiadmin';
$wgDBadminpassword  = 'abcdefghijklm';
$wgDBuser           = 'wikiuser';
$wgDBpassword       = 'abcdefghijklm';
```

5.1.3.3 assert-dbms-connect-with-credentials-wikiadmin-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
```

WP-MIRROR, using the wikiadmin credentials, asserts that it is possible to connect to the MySQL server, and execute a simple query.

5.1.3.4 assert-dbms-connect-with-credentials-wikiuser-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-connect-with-credentials-wikiuser-p
```

WP-MIRROR, using the wikiuser credentials, asserts that it is possible to connect to the MySQL server, and execute a simple query.

5.1.3.5 assert-dbms-grant-for-wikiadmin-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-grant-for-wikiadmin-p
```

WP-MIRROR, using the Debian credentials, GRANTs elevated database privileges to wikiadmin. Namely,

- GRANT ALL PRIVILEGES ON each of the databases: wikidb, wpmirror, simplewiki (and any other database in a wikipedia farm), and
- GRANT PROCESS globally.

This last privilege is needed for monitoring the status of the InnoDB storage engine.

5.1.3.6 assert-dbms-grant-for-wikiuser-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-grant-for-wikiuser-p
```

WP-MIRROR, using Debian credential, GRANTs limited privileges to wikiuser. Namely, SELECT, INSERT, UPDATE, and DELETE privileges on wikidb, wpmirror, simplewiki (and any other database in a wikipedia farm).

5.1.3.7 warn-if-dbms-root-account-has-no-password

```
root-shell# wp-mirror --mirror
...
[ ok ]warn-if-dbms-root-account-has-no-password
```

MySQL, in its 'out-of-the-box' configuration, is insecure. WP-MIRROR, using Debian credentials, emits a warning if it detects a root account with no password.

5.1.3.8 warn-if-dbms-has-anonymous-user-account

```
root-shell# wp-mirror --mirror
...
[ ok ]warn-if-dbms-has-anonymous-user-account
```

MySQL, in its 'out-of-the-box' configuration, is insecure. WP-MIRROR, using Debian credentials, emits a warning if it detects an anonymous user account.

5.1.3.9 warn-if-dbms-has-root-accounts-accessible-from-outside-localhost

```
root-shell# wp-mirror --mirror
...
[ ok ]warn-if-dbms-has-root-accounts-accessible-from-outside-localhost
```

MySQL, in its 'out-of-the-box' configuration, is insecure. WP-MIRROR, using Debian credentials, emits a warning if it detects a root account that is accessible from an host outside localhost.

5.1.3.10 warn-if-dbms-has-test-database

```
root-shell# wp-mirror --mirror
...
[ ok ]warn-if-dbms-has-test-database
```

MySQL, in its 'out-of-the-box' configuration, is insecure. WP-MIRROR, using Debian credentials, emits a warning if it detects a test database.

5.1.3.11 assert-database-wpmirror-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-database-wpmirror-or-create-p
```

WP-MIRROR maintains state information that must be shared with one or more mirror and monitor processes, each of which must be Isolated (the 'I' in ACID) from each other. This state information also serves as a checkpoint, and must therefore be Durable (the 'D' in ACID) to facilitate resuming after interruption.

WP-MIRROR asserts (and if necessary creates) a database wpmirror, managed by the ACID compliant storage engine InnoDB, to hold its state information.

Beginning with WP-MIRROR 0.5, this function also carries the responsibility for upgrading the wpmirror database schema. If a schema update occurs, it will be noted in the log file, /var/log/wp-mirror.log.

WP-MIRROR 0.5 introduced a change: the schema for the wpmirror.file database table was altered to support keeping track of databases. This was motivated by desire to be able to add and drop wikipedias concurrently with building yet other wikipedias; which in turn required the introduction of a new file type (i.e. database); and hence, a new schema for the wpmirror.file database table.

The SQL command that effects the change for WP-MIRROR 0.5 is:

```
mysql> ALTER TABLE file
    -> MODIFY COLUMN type ENUM('database','checksum','dump','xml','xchunk',
    ->                         'ichunk','images','error')
    ->                     NOT NULL DEFAULT 'error';
```

WP-MIRROR 0.6 introduced further changes:

1. the wpmirror.file database table was altered again in order to support: fast index creation (table), the splitting of large SQL files (sdump, sql, schunk, dchunk), and to support downloading image dump tarballs (idump);
2. the wpmirror.image database table was removed now that we no longer need to validate image files;
3. the wpmirror.priority database table was introduced to support the prioritization of tasks managed by the finite state machine; and
4. the wpmirror.time database table was introduced to support profiling;

The SQL command that effects the changes for WP-MIRROR 0.6 are:

```
mysql> ALTER TABLE file
    -> ADD    COLUMN project VARCHAR(20) NOT NULL DEFAULT 'error' AFTER timestamp,
    -> ADD    COLUMN wiki    VARCHAR(30) NOT NULL DEFAULT 'error' AFTER project,
    -> MODIFY COLUMN type    ENUM('database','table','checksum','xdump','xml','xchunk',
    ->                            'sdump','sql','schunk','dchunk',
    ->                            'idump','ichunk','images','error')
    ->                       NOT NULL DEFAULT 'error';
```

or,

```
mysql> SHOW CREATE TABLE wpmirror.file\G
*************************** 1. row ***************************
       Table: file
Create Table: CREATE TABLE `file` (
  `timestamp`     timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
                  ON UPDATE CURRENT_TIMESTAMP,
  `project`       varchar(20) NOT NULL DEFAULT 'error',
  `wiki`          varchar(30) NOT NULL DEFAULT 'error',
  `language_code` varchar(20) NOT NULL DEFAULT 'error',
  `date`          varchar(8)  NOT NULL DEFAULT 'error',
  `name`          varchar(80) NOT NULL DEFAULT 'error',
  `size`          bigint(20) unsigned NOT NULL DEFAULT '0',
  `md5sum`        varchar(32) NOT NULL DEFAULT 'error',
  `type`          ENUM('database','table','checksum','xdump','xml','xchunk',
                       'sdump','sql','schunk','dchunk',
                       'idump','ichunk','images','error')
                  NOT NULL DEFAULT 'error',
  `state`         enum('start','created','valid','pending','done','fail','error')
                  NOT NULL DEFAULT 'error',
  `page`          int(10) unsigned NOT NULL DEFAULT '0',
  `pages`         int(10) unsigned NOT NULL DEFAULT '0',
  `images`        int(10) unsigned NOT NULL DEFAULT '0',
  `updates`       int(10) unsigned NOT NULL DEFAULT '0',
  `semaphore`     int(10) unsigned NOT NULL DEFAULT '1',
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

as well as

```
mysql> SHOW CREATE TABLE wpmirror.priority\G
*************************** 1. row ***************************
       Table: priority
Create Table: CREATE TABLE `priority` (
  `id`         int(10) unsigned NOT NULL AUTO_INCREMENT,
  `type`       enum('database','table','checksum','xdump','xml','xchunk',
                    'sdump','sql','schunk','dchunk',
                    'idump','ichunk','images','error')
               NOT NULL DEFAULT 'error',
  `state`      enum('start','created','valid','pending','done','fail','error')
               NOT NULL DEFAULT 'error',
  `concurrent` int(10) unsigned NOT NULL DEFAULT '0',
  `image`      int(10) unsigned NOT NULL DEFAULT '0',
  `commons`    int(10) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=52 DEFAULT CHARSET=utf8
1 row in set (0.02 sec)
```

and

```
mysql> SHOW CREATE TABLE wpmirror.time\G
*************************** 1. row ***************************
       Table: time
Create Table: CREATE TABLE `time` (
  `id`            int(10) unsigned NOT NULL AUTO_INCREMENT,
  `timestamp`     timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
                  ON UPDATE CURRENT_TIMESTAMP,
  `run`           int(10) unsigned NOT NULL DEFAULT '0',
  `function_name` varchar(80) NOT NULL DEFAULT 'error',
  `file_name`     varchar(80) NOT NULL DEFAULT 'error',
  `real_time`     bigint(20) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8631 DEFAULT CHARSET=utf8
1 row in set (0.02 sec)
```

5.1.3.12 assert-database-template-and-wikidb-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-database-template-and-wikidb-p
```

WP-MIRROR must be able to mirror wikipedias of more than one language. It must be possible to create or delete one wikipedia without touching others. The author made a design decision to have MediaWiki store each wikipedia in a separate database. This means that each database must have the identical table structure.

WP-MIRROR, provides a database template for wikipedia databases. This template is /usr/share/mediawiki/maintenance/database_farm.sql. Almost all of the tables in the template are empty. This template is later used to create one MediaWiki compatible database for each wikipedia, but first WP-MIRROR uses it into the wikidb database.

Actually, for robustness; WP-MIRROR, using Debian credentials, will do one of the following:

- if the database template exists, and the wikidb database does not; then the database template is LOADed to create the wikidb database; and, conversely,
- if the wikidb database exists, and the database template does not; then the wikidb database is DUMPed to create the database template.

The first case is the usual one. However, the adventurous user who inadvertently deletes the database template will find it back again if wikidb had been created during a previous run.

5.1.3.13 assert-mediawiki-localsettings-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-mediawiki-localsettings-p
```

MediaWiki expects to find user configuration in /etc/mediawiki/LocalSettings.php. WP-MIRROR provides this file.

However, it often happens that the user edits this file, and then forgets to set the file ownership and permissions. In particular, it is important that the file not be world readable, because it may contain database credentials.

WP-MIRROR, therefore, asserts that this file exists, and sets its ownership and permissions.

5.1.3.14 assert-mediawiki-localsettings-account-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-mediawiki-localsettings-account-p
```

The file is /etc/mediawiki/LocalSettings_account.php is created by WP-MIRROR to hold the database credentials for wikiadmin and wikiuser.

However, because the file contains database credentials, it is important that the file not be world readable.

WP-MIRROR, therefore, asserts that this file exists, and sets its ownership and permissions.

5.1.3.15 assert-mediawiki-localsettings-wpmirror-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-mediawiki-localsettings-wpmirror-p
```

The file is /etc/mediawiki/LocalSettings_wpmirror.php is created by WP-MIRROR to hold code for the wikipedia farm as well as additional configuration.

However, because the file may contain database credentials, it is important that the file not be world readable.

WP-MIRROR, therefore, asserts that this file exists, and sets its ownership and permissions.

5.1.3.16 assert-mediawiki-localsettings-image-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-mediawiki-localsettings-image-p
```

MediaWiki, by default, processes images with convert (ImageMagick). ImageMagick overcommits main memory and can cause the system to hang.

The author, therefore, replaces ImageMagick with:

- gm convert (GraphicsMagick) for resizing images (making thumbs); and
- inkscape for converting files in SVG format into PNG image files.

The configuration for that is written to /etc/mediawiki/LocalSettings_wpmirror.php, and looks like:

```
## Images - local
$wgEnableUploads        = true;
...
## Images - thumb
$wgUseImageMagick       = false;
$wgCustomConvertCommand = '/usr/bin/gm convert %s -resize %wx%h %d';
$wgSVGConverter         = 'inkscape';
```

WP-MIRROR reads and parses the file for the last three lines shown above.

5.1.3.17 assert-mediawiki-localsettings-tidy-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-mediawiki-localsettings-tidy-p
```

Many wikipedia articles contain complicated templates that, without tidy, will produce badly formatted pages. In particular, you will find <p> and <pre> tags in the HTML after every citation. The resulting mess will be hard to read. Tidy is an HTML syntax checker and reformatter, and is needed for generating readable pages.

In /etc/mediawiki/LocalSettings_wpmirror.php there should be a line that reads:

```
$wgUseTidy = true;
```

5.1.3.18 assert-mediawiki-favicon-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-mediawiki-favicon-p
```

A favicon is a small icon, usually 16x16 or 32x32 pixels, that is displayed by a web browser in the Address Bar (the text box where you type the URL). The favicon is displayed just to the left of the URL.

WP-MIRROR provides a favicon, /var/lib/mediawiki/favicon.ico, which is a 16x16 pixel version of the WP-MIRROR logo.

5.1.3.19 assert-mediawiki-logo-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-mediawiki-favicon-p
```

WP-MIRROR provides a 135x135 pixel logo /var/lib/mediawiki/wp-mirror.png that is displayed by a web browser in the upper left corner of the page.

MediaWiki offers a number of 'skins' (cascading style sheets mostly) which govern the layout of each web page. WP-MIRROR uses the vector skin (default). The logo dimension, 135x135 pixel, is chosen for compatibility the legacy skin Monobook.

5.1.3.20 assert-mediawiki-dbms-credentials-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-mediawiki-dbms-credentials-p
```

Before WP-MIRROR can request MediaWiki maintenance scripts to import pages and images, all the database credentials must be in hand. WP-MIRROR asserts that they are. If they are not, WP-MIRROR exits.

5.1.3.21 assert-concurrency-limit-xchunk-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-concurrency-limit-xchunk-p
```

InnoDB has two table space formats (as of 2013)

- Antelope is uncompressed and handles concurrency well. WP-MIRROR limits concurrency to the lesser of the number of CPUs and three.
- Barracuda is compressed using zlib and does *not* handle concurrency well (frequent deadlocks). WP-MIRROR limits concurrency to just one. That is, xchunks are processed one-by-one.

5.1.3.22 assert-virtual-host-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-virtual-host-p
```

WP-MIRROR lets you access your mirror locally using a web browser. For this to happen, a web server (such as apache2) running on localhost needs to know where to look.

WP-MIRROR sets up virtual hosts like wikipedia.site and wiktionary.site. The virtual host containers can be found in /etc/apache2/sites-enabled/wpmirror.site.conf, and look like:

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wikipedia.site
    # for access using:  en.wikipedia.site, simple.wikipedia.site, etc.
    ServerAlias  *.wikipedia.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

and

```
<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    ServerName   wiktionary.site
    # for access using:  en.wiktionary.site, simple.wiktionary.site, etc.
    ServerAlias  *.wiktionary.site
    DocumentRoot /var/lib/mediawiki
    # for logs using:  NCSA Combined Log Format
    CustomLog    $APACHE_LOG_DIR/wpmirror.site-access.log combined
    ErrorLog     $APACHE_LOG_DIR/wpmirror.site-error.log
</VirtualHost>
```

5.1.3.23 assert-virtual-host-name-resolution-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-virtual-host-name-resolution-p
```

For each language to be mirrored, WP-MIRROR creates a virtual host name. Each such host name must be resolved to an IP address.

WP-MIRROR scans /etc/hosts for virtual host names like simple.wikipedia.site, where simple might be replaced with any other language code. For each missing virtual host name, WP-MIRROR appends to /etc/hosts lines like:

```
::1 simple.wikipedia.site
::1 simple.wiktionary.site
```

5.1.3.24 warn-if-detect-proxy

```
root-shell# wp-mirror --mirror
...
[ ok ]warn-if-detect-proxy
```

Some caching web proxies (such as polipo) are not able to download files larger than the available system memory (DRAM). And some proxies do not have sufficient cache to handle terabytes of image files.

WP-MIRROR does not attempt to configure your caching web proxy (which is usually on a different host). Rather, WP-MIRROR scans several configuration files (those for bash, curl, and wget) looking for evidence of a proxy. If such evidence is found, WP-MIRROR issues a warning:

```
root-shell# wp-mirror --mirror
...
[warn]warn-if-detect-proxy
```

and continues processing.

5.1.4 Asserting Prerequisite Hardware

```
...
-----asserting-prerequisite-hardware-begin----
[ ok ]count-cpu
[ ok ]assert-disk-space-if-large-wikipedia-p
[ ok ]assert-physical-memory-if-large-wikipedia-p
[ ok ]assert-partition-free-images
[ ok ]warn-if-disk-space-low-p
[ ok ]warn-if-database-on-virtual-disk-p
[ ok ]assert-hdd-write-cache-p
[ ok ]assert-internet-access-to-wikimedia-site-p
```

These messages show the progress of asserting hardware prerequisites. We now describe each of them:

5.1.4.1 count-cpu

```
root-shell# wp-mirror --mirror
...
[ ok ]count-cpu
```

WP-MIRROR counts the number of CPUs as part of deciding how much concurrent processing (if any) can be permitted.

5.1.4.2 assert-disk-space-if-large-wikipedia-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-disk-space-if-large-wikipedia-p
```

WP-MIRROR initially (and periodically) determines the amount of free disk space available for storing databases and images.

If you are attempting to mirror any of the largest wikipedias, WP-MIRROR will require that you have >100G of free disk space before you can start. While this is not nearly enough to hold a large wikipedia, it is enough to hold the working files (checksum, xdump, etc.).

5.1.4.3 assert-physical-memory-if-large-wikipedia-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-physical-memory-if-large-wikipedia-p
```

WP-MIRROR initially determines the amount of main memory (DRAM) installed.

If you are attempting to mirror any of the largest wikipedias, WP-MIRROR will require that you have ≥4G of main memory before you can start. While this is not enough to build a mirror, it is enough to create all the working files (checksum, xdump, etc.).

5.1.4.4 assert-partition-free-images

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-partition-free-images
```

WP-MIRROR initially (and periodically) determines the amount of free disk space available for storing images. If the amount falls below 5G, WP-MIRROR gracefully exits.

5.1.4.5 warn-if-disk-space-low-p

```
root-shell# wp-mirror --mirror
...
[ ok ]warn-if-disk-space-low-p
```

WP-MIRROR initially (and periodically) determines the amount of free disk space available for storing images and the InnoDB table space.

WP-MIRROR initially warns when there is <90G of free disk space, in which case, the message looks like:

```
root-shell# wp-mirror --mirror
...
[....]warn-if-disk-space-low-p
[info]disk space below threshold ( 17G <  90G)
[warn]warn-if-disk-space-low-p
```

The value 90G was chosen because, WP-MIRROR in its default configuration, mirrors the simple wikipedia and simple wiktionary, which occupy 90G.

5.1.4.6 warn-if-database-stored-on-virtual-disk-p

```
root-shell# wp-mirror --mirror
...
[....]warn-if-database-stored-on-virtual-disk-p
[info]/dev/sda: disk identification found
[ ok ]warn-if-database-stored-on-virtual-disk-p
```

WP-MIRROR initially warns if the underlying disk(s) do not provide identification. This happens when WP-MIRROR is installed on a virtual machine, in which case, hdparm throws an error message like:

```
root-shell# hdparm -I /dev/vda

/dev/vda:
 HDIO_DRIVE_CMD(identify) failed: Inappropriate ioctl for device
```

and WP-MIRROR issues:

```
root-shell# wp-mirror --mirror
...
[....]warn-if-database-stored-on-virtual-disk-p
[info]/dev/vda: disk identification NOT found
[warn]warn-if-database-stored-on-virtual-disk-p
```

and continues processing.

5.1.4.7 assert-hdd-write-cache-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-hdd-write-cache-p
```

MediaWiki stores its articles in InnoDB, which is MySQL's ACID compliant storage engine that is used for transactions. The issue here is Durability (the 'D' in ACID). A committed transaction should be stored in a way that is resistant to many kinds of failure, including power outage. Transactions that COMMIT must actually be written to disk, and not retained in the HDDs write cache, where it may be lost during system failure. It is therefore important that the disk's write cache either be disabled or flushed immediately after each checkpoint (and both are configurable).

WP-MIRROR first identifies the disk(s) that hold the InnoDB table space. Next WP-MIRROR attempts to configure the write cache for each such disk. Finally, WP-MIRROR asserts that the write cache(s) have been successfully configured.

If hdparm cannot obtain identification from the disk(s), as is the case when WP-MIRROR is installed on a virtual machine, then this assertion is allowed to fail; in which case, WP-MIRROR issues:

```
root-shell# wp-mirror --mirror
...
[fail]assert-hdd-write-cache-p
```

and continues processing.

5.1.4.8 assert-internet-access-to-wikimedia-site-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-internet-access-to-wikimedia-site-p
```

WP-MIRROR requires Internet access for some of its tasks. Therefore, WP-MIRROR initially (and periodically) asserts Internet access. If initially there is no access, WP-MIRROR exits. However, in all other circumstances, WP-MIRROR sleeps for 10s (default) and tries again until access is restored.

5.1.5 The Finite State Machine

Most of what WP-MIRROR does is file processing.

Depending on circumstances, files may be: downloaded, validated, parsed, decompressed, counted, split into smaller files, imported into MediaWiki, have their permissions changed, deleted, etc. In most cases, files can be processed concurrently, while in other cases, they must be processed in a specific order.

For WP-MIRROR 0.2 and before, process scheduling was handled by hundreds of lines of 'spaghetti code', which soon became a maintenance headache. Whereas the human mind struggles with large blocks of code, and whereas the human mind readily grasps data in tabular format, the experienced engineer looks for ways to replace the former with the later.

For WP-MIRROR 0.3 and later versions: 1) Every file is assigned a:

- `type` (one of `database`, `table`, `checksum`, `xdump`, `xml`, `xchunk`, `sdump`, `sml`, `schunk`, `idump`, `ichunk`, `images`, and `error`), and a
- `state` (one of `start`, `created`, `valid`, `pending`, `done`, `fail`, and `error`).

This information is stored in the `wpmirror.file` database table.

2) A Finite State Machine (FSM) is employed. This requires very few lines of code, because all decision-making is lodged in a set of three tables:

- `*state-transition*` describes how a file changes state according to events (`start`, `done`, or `fail`);
- `*type-state-function*` describes what function (`fsm-file-download`, `fsm-file-decompress`, etc.) is applied to a file of a given `type` and `state`; and
- `*type-state-priority*` describes the order in which files are processed, and if they may be processed concurrently.

These FSM tables are shown below:

- For `*state-transition*` see Figure 5.1, FSM State Transition Diagram and the top part of Table 5.2, FSM Tables `*state-transition*` and `*type-state-function*`;
- For `*type-state-function*` see the bottom part of Table 5.2, FSM Tables `*state-transition*` and `*type-state-function*`; and
- For `*type-state-priority*` see Table 5.3, FSM Priority Table `*type-state-priority*`.

Figure 5.1: FSM State Transition Diagram

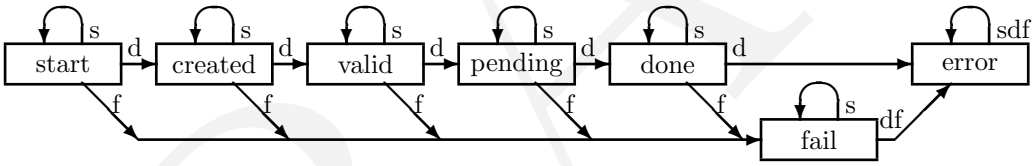


Table 5.2: FSM Tables *state-transition* and *type-state-function*

Event	**State**			
	start	created	valid	pending
start	start	created	valid	pending
done	created	valid	pending	done
fail	fail	fail	fail	fail

File type	**State**			
	start	created	valid	pending
database	fsm-database-grant	fsm-database-create	fsm-database-checksum	fsm-database-interwiki
table	fsm-table-block-size	fsm-table-drop-index	fsm-table-add-index	fsm-no-op
checksum	fsm-file-download	fsm-file-digest	fsm-file-parse	fsm-no-op
xdump	fsm-file-download	fsm-file-validate	fsm-file-decompress	fsm-no-op
xml	fsm-file-count	fsm-file-digest	fsm-file-split-xml	fsm-file-remove
xchunk	fsm-file-count	fsm-file-digest	fsm-file-xml2sql	fsm-file-remove
sdump	fsm-file-download	fsm-file-validate	fsm-file-split-sdump	fsm-no-op
sql	fsm-no-op	fsm-file-digest	fsm-file-split-sql	fsm-file-remove
schunk	fsm-no-op	fsm-no-op	fsm-file-load-insert	fsm-file-remove
dchunk	fsm-no-op	fsm-no-op	fsm-file-load-insert	fsm-file-remove
idump	fsm-file-download	fsm-no-op	fsm-file-extract	fsm-file-remove
ichunk	fsm-file-list-missing	fsm-no-op	fsm-file-wget	fsm-file-remove
images	fsm-images-directory	fsm-no-op	fsm-no-op	fsm-no-op

Event	**State**		
	done	fail	error
start	done	fail	error
done	done	fail	error
fail	error	error	error

File type	**State**		
	done	fail	error
database	fsm-no-op	fsm-abort	fsm-abort
table	fsm-no-op	fsm-abort	fsm-abort
checksum	fsm-no-op	fsm-abort	fsm-abort
xdump	fsm-no-op	fsm-abort	fsm-abort
xml	fsm-no-op	fsm-abort	fsm-abort
xchunk	fsm-no-op	fsm-abort	fsm-abort
sdump	fsm-no-op	fsm-abort	fsm-abort
sql	fsm-no-op	fsm-abort	fsm-abort
schunk	fsm-no-op	fsm-abort	fsm-abort
dchunk	fsm-no-op	fsm-abort	fsm-abort
idump	fsm-no-op	fsm-abort	fsm-abort
ichunk	fsm-no-op	fsm-abort	fsm-abort
images	fsm-no-op	fsm-abort	fsm-abort

Table 5.3: FSM Priority Table *type-state-priority*

Priority	Type	State	Concurrent	Image	Commons	Function
highest	database	pending	t	t	t	fsm-database-interwiki
	database	valid	t	t	t	fsm-database-checksum
	database	created	t	t	t	fsm-database-create
	database	start	t	t	t	fsm-database-grant
	table	pending	t	t	t	fsm-no-op
	table	start	nil	t	t	fsm-table-block-size
	checksum	pending	t	t	t	fsm-no-op
	checksum	valid	t	t	t	fsm-file-parse
	checksum	created	t	t	t	fsm-file-digest
	checksum	start	nil	t	t	fsm-file-download
	xdump	pending	t	t	nil	fsm-file-remove
	xdump	valid	t	t	nil	fsm-file-decompress
	xdump	created	t	t	nil	fsm-file-validate
	xdump	start	nil	t	nil	fsm-file-download
	xml	pending	t	t	nil	fsm-file-remove
	xml	valid	t	t	nil	fsm-file-split-xml
	xml	created	t	t	nil	fsm-file-digest
	xml	start	t	t	nil	fsm-file-count
	xchunk	pending	t	t	nil	fsm-file-remove
	xchunk	valid	t	t	nil	fsm-file-xml2sql
	xchunk	created	t	t	nil	fsm-file-digest
	xchunk	start	t	t	nil	fsm-file-count
	sdump	pending	t	t	t	fsm-file-remove
	sdump	valid	t	t	t	fsm-file-split-sdump
	sdump	created	t	t	t	fsm-file-validate
	sdump	start	nil	t	t	fsm-file-download
	sql	pending	t	t	t	fsm-file-remove
	sql	valid	t	t	t	fsm-file-split-sql
	sql	created	t	t	t	fsm-file-digest
	sql	start	t	t	t	fsm-no-op
	table	created	t	t	t	fsm-table-drop-index
	schunk	pending	t	t	t	fsm-file-remove
	schunk	valid	nil	t	t	fsm-file-load-insert
	schunk	created	t	t	t	fsm-no-op
	schunk	start	t	t	t	fsm-no-op
	dchunk	pending	t	t	t	fsm-file-remove
	dchunk	valid	nil	t	t	fsm-file-load-insert
	dchunk	created	t	t	t	fsm-no-op
	dchunk	start	t	t	t	fsm-no-op
	table	valid	nil	t	t	fsm-table-add-index
	idump	pending	t	nil	nil	fsm-file-remove
	idump	valid	nil	nil	nil	fsm-file-extract
	idump	created	t	nil	nil	fsm-no-op
	idump	start	nil	nil	nil	fsm-file-download
	images	start	t	nil	t	fsm-images-directory
	ichunk	pending	t	nil	nil	fsm-file-remove
	ichunk	valid	nil	nil	nil	fsm-file-wget
	ichunk	created	t	nil	nil	fsm-no-op
	ichunk	start	t	nil	nil	fsm-file-list-missing
	images	created	t	nil	t	fsm-no-op
	images	valid	t	nil	t	fsm-no-op
lowest	images	pending	t	nil	t	fsm-no-op

5.1.6 The Finite State Machine—Step by Step

```
root-shell# wp-mirror --mirror
...
-----mirror-mode-begin-----------------------
[ ok ]fsm-boot
[ ok ]release-all-file-semaphores
[ ok ]fsm-database-grant simplewiki
[ ok ]fsm-database-create simplewiki
[ ok ]fsm-database-checksum simplewiki
[ ok ]fsm-database-interwiki simplewiki
[ ok ]fsm-file-download simplewiki-20121106-md5sums.txt
[ ok ]fsm-file-digest simplewiki-20121106-md5sums.txt
[ ok ]fsm-file-parse simplewiki-20121106-md5sums.txt
[ ok ]fsm-no-op simplewiki-20121106-md5sums.txt
```

The FSM works as follows:

Step 0: Boot FSM. The function `fsm-boot` inserts into the `wp-mirror.file` table, one row for each language you desire. Each row contains the name of a database that holds the pages and articles (the default is `simplewiki`), its `type` (`database`), its `state` (`start`), and a semaphore (set to `1`).

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+-------+-----------+
| name       | type     | state | semaphore |
+------------+----------+-------+-----------+
| simplewiki | database | start |         1 |
+------------+----------+-------+-----------+
```

Step 1: Prioritize rows. The FSM orders these rows according to the `*type-state-priority*` table. For any given row, the values of `concurrent`, `image`, and `commons` control whether of not the function in that row will be executed.

Concurrent	Image	Commons	Function
t	t	t	runs unconditionally
nil	t	t	runs if `*main-mode*` is `:first-mirror`
t	nil	t	runs if `*mirror-image-download-p*`
t	t	nil	runs if `wiki` is not `commonswiki`

The default is to have image files for the selected wikis, but not the entire `commons` collection which occupies nearly 20T (as of 2013):

```
(defparameter *mirror-image-download-p*          t)
```

The default can be changed in two ways:

- **Images**: If an user wants no images, then edit `/etc/wp-mirror/local.conf` to contain:

  ```
  (defparameter *mirror-image-download-p*         nil)
  ```

- **Commons**: If an user wants the entire `commons` image file collection (and has the storage), then edit `/etc/wp-mirror/local.conf` to contain:

  ```
  (defparameter *mirror-language-code-list*  '(``commonswiki'' ``simple''))
  ```

Step 2: Grab semaphore. The FSM grabs the semaphore for the highest priority row.

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+-------+-----------+
| name       | type     | state | semaphore |
+------------+----------+-------+-----------+
| simplewiki | database | start |         0 |
+------------+----------+-------+-----------+
```

Caveat: This is true only for the :first-mirror instance. The :next-mirror instance (if any) grabs the semaphore of the highest priority row for which the concurrent value in the *type-state-priority* table is t. See §5.1.8, Concurrency for details.

Step 3: Look-up function. The FSM looks up the appropriate function from the *type-state-function* table. In this case, the function is fsm-database-create.

File type	State			
	start	created	valid	...
database	fsm-database-grant	fsm-database-create	fsm-database-checksum	...
checksum	fsm-file-download	fsm-file-digest	fsm-file-parse	...
...	...	...	...	...

Step 4: Apply function. The FSM applies the function to the file. In this case, fsm-database-create loads a (nearly) empty database from the database template file /usr/share/mediawiki/maintenance/database_farm.sql.

Step 5: Capture return value. The FSM captures the return value of the function (either done or fail).

Step 6: Look-up state transition. The FSM looks up the next state from the *state-transition* table. In this case, if the function returned done, then the next state will be created.

Event	State			
	start	created	valid	...
start	start	created	valid	...
done	created	valid	pending	...
fail	fail	fail	fail	...

File type	State			
	start	created	valid	...
database	fsm-database-grant	fsm-database-create	fsm-database-checksum	...
...	...	...	...	...

Step 7: Change state. The FSM updates the row in the wpmirror.file database table with the value of the next state.

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+---------+-----------+
| name       | type     | state   | semaphore |
+------------+----------+---------+-----------+
| simplewiki | database | created |         0 |
+------------+----------+---------+-----------+
```

Step 8: Release semaphore. The FSM releases the semaphore.

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+---------+-----------+
| name       | type     | state   | semaphore |
+------------+----------+---------+-----------+
| simplewiki | database | created |         1 |
+------------+----------+---------+-----------+
```

Step 9: Repeat until done. The FSM then returns to **Step 1**, and continues processing until all rows in the table are in an end `state` (`done` or `fail`). The `error state` is never reached unless the FSM itself contains an error.

Some functions insert additional rows into the `wpmirror.file` database (and these are colored in Table 5.2, FSM Tables *state-transition* and *type-state-function*). They are: `fsm-database-checksum`, `fsm-file-parse`, `fsm-file-decompress`, `fsm-file-split-sql`, and, `fsm-file-split-xml`, `fsm-file-scrape`, and `fsm-file-xml2sql`. See §5.1.9, The fsm-* Functions for details.

5.1.7 Check-pointing

Process interruptions happen (power failure, laptops closed, cat walks across the keyboard).

Whenever WP-MIRROR resumes after an interruption, the FSM finds the state of every file in the `wpmirror.file` database table just as it was before the interruption. In other words, every time the FSM inserts or updates a row in the `wpmirror.file` database table, the FSM is making a check-point.

Due to check-pointing, no completed task (meaning, for which the semaphore was released) is ever repeated. This is critical for mirroring the large wikipedias, where the initial build could take weeks, and where starting over would be unacceptable.

5.1.8 Concurrency

Experiments show that, for the desktop case, concurrent processing can reduce total run time (wall clock time) by a factor of two or three. For the desktop case, the recommended number of WP-MIRROR instances that may run concurrently is:

- one instance per CPU in `:first-mirror` or `:next-mirror` mode, for importing `xchunk`s,
- one or two instances in `:next-mirror` mode, for downloading images, and
- one instance in `:monitor` mode;

Although, for the laptop case, experiments show that the recommended number is only:

- one instance in `:first-mirror` mode, and
- one instance in `:monitor` mode.

Concurrent processing does however require a mechanism to prevent two WP-MIRROR instances from attempting to process the same file at the same time. Each WP-MIRROR instance must be isolated (the 'I' in ACID) from every other instance.

Here is how we enforce isolation:

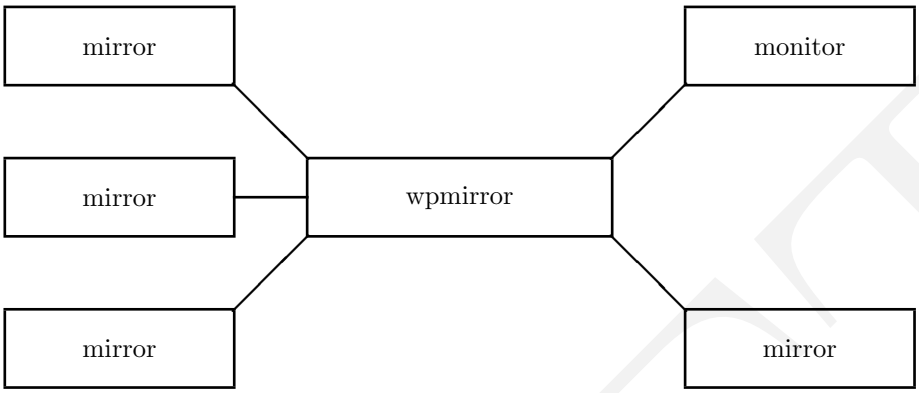
First, we require each file to have a `type` and a `state` recorded in a row in the `wpmirror.file` table.

Second, in Table 5.3, FSM Priority Table *type-state-priority* there is a column labeled `concurrent`, which indicates whether or not that file may be processed concurrently with other files. For example, an `xml` file may be processed concurrently with any other file.

Third, we require every WP-MIRROR instance to communicate state *only* via an ACID compliant transactional database storage engine, such as `InnoDB`. See Figure 5.2, WP-MIRROR Instances Must Communicate State *Only* via the `wpmirror` Database.

Fourth, every row in the `wpmirror.file` database table must have a value in its semaphore field. The schema for `wpmirror.file` is:

Figure 5.2: WP-MIRROR Instances Must Communicate State *Only* via the wpmirror Database




```
mysql> SHOW CREATE TABLE `wpmirror`.`file`\G
*************************** 1. row ***************************
       Table: file
Create Table: CREATE TABLE `file` (
  `timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
              ON UPDATE CURRENT_TIMESTAMP,
  `project`       varchar(20) NOT NULL DEFAULT 'error',     -- e.g. 'wikipedia'
  `wiki`          varchar(30) NOT NULL DEFAULT 'error',     -- e.g. 'simplewiki'
  `language_code` varchar(20) NOT NULL DEFAULT 'error',     -- e.g. 'simple'
  `date`          varchar(8)  NOT NULL DEFAULT 'error',     -- e.g. '20121106'
  `name`          varchar(80) NOT NULL DEFAULT 'error',
  `size`          bigint(20) unsigned NOT NULL DEFAULT '0', -- [0-18e18]
  `md5sum`        varchar(32) NOT NULL DEFAULT 'error',
  `type`          enum('database','checksum','xdump','idump','xml','sql',
                       'xchunk','schunk','ichunk','images','error')
                  NOT NULL DEFAULT 'error',
  `state`         enum('start','created','valid','pending','done','fail','error')
                  NOT NULL DEFAULT 'error',
  `page`          int(10) unsigned NOT NULL DEFAULT '0',
  `pages`         int(10) unsigned NOT NULL DEFAULT '0',    -- usually 1000
  `images`        int(10) unsigned NOT NULL DEFAULT '0',
  `updates`       int(10) unsigned NOT NULL DEFAULT '0',
  `semaphore`     int(10) unsigned NOT NULL DEFAULT '1',    -- [0-1]
  PRIMARY KEY (`name`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;                      -- ACID compliant
```

Fifth, we require every WP-MIRROR instance to obey the following semaphore rules:

- **Rule 1**: Before a WP-MIRROR instance is allowed to process a file, it must first:
  - look up the corresponding row in the wpmirror.file table,
  - determine if the semaphore's value is '1', and if so,
  - set the semaphore's value to '0'.
- **Rule 2**: After a WP-MIRROR instance finishes processing a file, it must:
  - capture the fsm-function's return value (done or fail),
  - look up the next state in the *state-transition* table,
  - update the state's value, and
  - set the semaphore's value to '1'.

The (colored) items in **Rule 1** above must be done together as a single transaction (otherwise two processes could simultaneously grab the semaphore and begin processing the corresponding file). This is done using the following atomic (the 'A' in ACID) transaction:

```
START TRANSACTION;                                  -- atomic transaction

SELECT @name := `name` AS `name`,@semaphore := `semaphore` AS `semaphore`,`id`
FROM `wpmirror`.`file`,`wpmirror`.`priority`
WHERE `file`.`type`=`priority`.`type`
AND   `file`.`state`=`priority`.`state`
AND (`wiki`<>'commonswiki' OR `commons`=1)
AND `semaphore`=1                                   -- semaphore is free
ORDER BY `id`,`name`
LIMIT 1
FOR UPDATE;                                         -- get IX lock on row

UPDATE `wpmirror`.`file` SET `semaphore`=0          -- semaphore is grabbed
WHERE name=@name;

COMMIT;                                             -- atomic transaction
```

When InnoDB sees the SELECT ...FOR UPDATE command, it gets an IX lock on the row. This lock is *exclusive*, meaning, it prevents any other process from accessing that row until the lock is released.

5.1.9 The fsm-* Functions

```
root-shell# wp-mirror --mirror
...
-----mirror-mode-begin-------------------------
[ ok ]fsm-boot
[ ok ]release-all-file-semaphores
[ ok ]fsm-database-grant simplewiki
[ ok ]fsm-database-create simplewiki
[ ok ]fsm-database-checksum simplewiki
[ ok ]fsm-database-interwiki simplewiki
[ ok ]fsm-file-download simplewiki-20121106-md5sums.txt
[ ok ]fsm-file-digest simplewiki-20121106-md5sums.txt
[ ok ]fsm-file-parse simplewiki-20121106-md5sums.txt
[ ok ]fsm-no-op simplewiki-20121106-md5sums.txt
...
```

This section contains detailed descriptions of each of the functions that are applied to files by the FSM.

5.1.9.1 fsm-abort

Purpose: Abort the FSM.

Motivation: The fsm-abort function would only be invoked if a file ended up in the error state. This can only happen if there is an error in the FSM itself (either in its code or in one of its three tables).

5.1.9.2 fsm-boot

Purpose: Initialize the Finite State Machine.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
Empty set (0.05 sec)
```

FSM function: Insert into the wp-mirror.file table, one row for each language you desire. Each row contains the name of the database that will hold the pages and articles (the default is simplewiki), its type (database), its state (start), and a semaphore (set to 1).

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+-------+-----------+
| name       | type     | state | semaphore |
+------------+----------+-------+-----------+
| simplewiki | database | start |         1 |
+------------+----------+-------+-----------+
```

5.1.9.3 fsm-database-checksum

Purpose: Insert record for the checksum file, the idump files, and for each table that will be stored with a compressed page size.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+-------+-----------+
| name       | type     | state | semaphore |
+------------+----------+-------+-----------+
| simplewiki | database | valid |         1 |
+------------+----------+-------+-----------+
```

FSM function: fsm-database-checksum inserts a record for the checksum, idump, and table files (the default values are shown below).

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+----------------------------------------+----------+---------+-----------+
| name                                   | type     | state   | semaphore |
+----------------------------------------+----------+---------+-----------+
| simplewiki                             | database | pending |         1 |
| simplewiki-20121106-md5sums.txt        | checksum | start   |         1 |
| simplewiki-20121201-local-media-1.tar  | idump    | start   |         1 |
| simplewiki-20121201-remote-media-1.tar | idump    | start   |         1 |
| simplewiki.categorylinks               | table    | start   |         1 |
| simplewiki.externallinks               | table    | start   |         1 |
| simplewiki.image                       | table    | start   |         1 |
| simplewiki.imagelinks                  | table    | start   |         1 |
| simplewiki.langlinks                   | table    | start   |         1 |
| simplewiki.pagelinks                   | table    | start   |         1 |
| simplewiki.templatelinks               | table    | start   |         1 |
| simplewiki.text                        | table    | start   |         1 |
+----------------------------------------+----------+---------+-----------+
```

5.1.9.4 fsm-database-create

Purpose: Create a (nearly) empty database from the database template file /var/lib/mediawiki/maintenance/database_farm.sql.

Motivation: WP-MIRROR must be able to mirror wikipedias of more than one language. Each wikipedia must be Isolated (the 'I' in ACID) from every other, so that one may CREATE or DROP a wikipedia without touching any other.

For each wikipedia, WP-MIRROR asserts (and if necessary CREATEs) a separate database with a name like *xx*wiki, where *xx* is stands for a language code. Each of these databases have the same table structure and are indeed CREATEd from the same database template.

WP-MIRROR uses Debian credentials for CREATE these databases, and to GRANT privileges on them, to the wikiadmin and wikiuser database accounts.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+---------+-----------+
| name       | type     | state   | semaphore |
+------------+----------+---------+-----------+
| simplewiki | database | created |         1 |
+------------+----------+---------+-----------+
```

FSM function: fsm-database-create first checks if the database already exists. If it does not, then fsm-database-create creates it using the database template file /var/lib/mediawiki/maintenance/database_farm.sql.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+-------+-----------+
| name       | type     | state | semaphore |
+------------+----------+-------+-----------+
| simplewiki | database | valid |         1 |
+------------+----------+-------+-----------+
```

5.1.9.5 fsm-database-grant

Purpose: GRANT database PRIVILEGES to the user accounts wikiadmin and wikiuser.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+-------+-----------+
| name       | type     | state | semaphore |
+------------+----------+-------+-----------+
| simplewiki | database | start |         1 |
+------------+----------+-------+-----------+
```

FSM function: fsm-database-grant, using the Debian credentials:

- GRANTs elevated database privileges to the user wikiadmin. Namely, GRANT ALL PRIVILEGES ON the database.
- GRANTs limited database privileges to the user wikiuser. Namely, SELECT, INSERT, UPDATE, and DELETE privileges on the database.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+------------+----------+---------+-----------+
| name       | type     | state   | semaphore |
+------------+----------+---------+-----------+
| simplewiki | database | created |         1 |
+------------+----------+---------+-----------+
```

5.1.9.6 fsm-database-interwiki

Purpose: Enter Interwiki links (interlanguage links actually) into the interwiki database table. That is, enter one row for each wikipedia in the mirror. That way, interlanguage links will: 1) be treated as *magical connectors*, 2) appear in the Navigation Sidebar under the 'Languages' menu, and 3) when clicked, carry the user to the corresponding article in another language.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------------+----------+---------+-----------+
| name                           | type     | state   | semaphore |
+--------------------------------+----------+---------+-----------+
| simplewiki                     | database | pending |         1 |
| simplewiki-20121106-md5sums.txt | checksum | start   |         1 |
| simplewiki.categorylinks       | table    | start   |         1 |
| ...                            | table    | start   |         1 |
+--------------------------------+----------+---------+-----------+
```

FSM function: Enter one row in the interwiki database table for each wikipedia. Each row specifies the language-code and URL of a wikipedia in the mirror.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------------+----------+-------+-----------+
| name                           | type     | state | semaphore |
+--------------------------------+----------+-------+-----------+
| simplewiki                     | database | done  |         1 |
| simplewiki-20121106-md5sums.txt | checksum | start |         1 |
| simplewiki.categorylinks       | table    | start |         1 |
| ...                            | table    | start |         1 |
+--------------------------------+----------+-------+-----------+
```

5.1.9.7 fsm-file-count

Purpose: Estimate the number of pages and image files.

Motivation: Estimate the work load early in the mirror building process. This information is displayed in :monitor mode, so as to give the user early notice as the work required to build the mirror.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-------------------------------------------+----------+---------+-----------+
| name                                      | type     | state   | semaphore |
+-------------------------------------------+----------+---------+-----------+
| simplewiki                                | database | done    |         1 |
| simplewiki-20121106-md5sums.txt           | checksum | done    |         1 |
| simplewiki-20121106-pages-articles.xml    | xml      | start   |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump    | pending |         1 |
+-------------------------------------------+----------+---------+-----------+
```

FSM function: fsm-file-count scans the decompressed xdump file to:

- count the number of articles (that may be inserted into the database), and
- to estimate the number of image files (that may be downloaded).

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+---------+-----------+
| name                                    | type     | state   | semaphore |
+-----------------------------------------+----------+---------+-----------+
| simplewiki                              | database | done    |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done    |         1 |
| simplewiki-20121106-pages-articles.xml  | xml      | created |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | pending |         1 |
+-----------------------------------------+----------+---------+-----------+
```

5.1.9.8 fsm-file-decompress

Purpose: To decompress a xdump file using bunzip2 and thereby generate an xml file.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+-------+-----------+
| name                                    | type     | state | semaphore |
+-----------------------------------------+----------+-------+-----------+
| simplewiki                              | database | done  |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done  |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | valid |         1 |
+-----------------------------------------+----------+-------+-----------+
```

FSM function: The fsm-file-decompress function decompresses the xdump file, thereby creating an xml file, which is then entered into the wpmirror.file database table.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+---------+-----------+
| name                                    | type     | state   | semaphore |
+-----------------------------------------+----------+---------+-----------+
| simplewiki                              | database | done    |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done    |         1 |
| simplewiki-20121106-pages-articles.xml  | xml      | start   |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | pending |         1 |
+-----------------------------------------+----------+---------+-----------+
```

5.1.9.9 fsm-file-digest

Purpose: Compute the md5sum of a file.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------------+----------+---------+-----------+
| name                           | type     | state   | semaphore |
+--------------------------------+----------+---------+-----------+
| simplewiki                     | database | done    |         1 |
| simplewiki-20121106-md5sums.txt | checksum | created |         1 |
+--------------------------------+----------+---------+-----------+
```

FSM function: The fsm-file-digest function computes the md5sum of the file, which is then entered into the wpmirror.file database table (in a field that is not shown in the example here).

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+---------------------------------+----------+-------+-----------+
| name                            | type     | state | semaphore |
+---------------------------------+----------+-------+-----------+
| simplewiki                      | database | done  |         1 |
| simplewiki-20121106-md5sums.txt | checksum | valid |         1 |
+---------------------------------+----------+-------+-----------+
```

5.1.9.10 fsm-file-download

Purpose: Used to download checksum, xdump, sdump, and idump files.
FSM before: Here we are about to download a checksum file.

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+---------------------------------+----------+-------+-----------+
| name                            | type     | state | semaphore |
+---------------------------------+----------+-------+-----------+
| simplewiki                      | database | done  |         1 |
| simplewiki-20121106-md5sums.txt | checksum | start |         1 |
+---------------------------------+----------+-------+-----------+
```

FSM function: fsm-file-download invokes wget to download the checksum file.
FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+---------------------------------+----------+---------+-----------+
| name                            | type     | state   | semaphore |
+---------------------------------+----------+---------+-----------+
| simplewiki                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt | checksum | created |         1 |
+---------------------------------+----------+---------+-----------+
```

5.1.9.11 fsm-file-extract

Purpose: Used to extract idump (image dump tarball).
FSM before: Here we are about to extract the idump file.

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+-------+-----------+
| name                                    | type     | state | semaphore |
+-----------------------------------------+----------+-------+-----------+
| simplewiki                              | database | done  |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done  |         1 |
| simplewiki-20121106-local-media-1.tar   | idump    | valid |         1 |
| simplewiki-20121106-remote-media-1.tar | idump    | valid |         1 |
+-----------------------------------------+----------+-------+-----------+
```

FSM function: fsm-file-extract invokes tar to extract the idump file.
FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+---------+-----------+
| name                                    | type     | state   | semaphore |
+-----------------------------------------+----------+---------+-----------+
| simplewiki                              | database | done    |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done    |         1 |
| simplewiki-20121106-local-media-1.tar   | idump    | pending |         1 |
| simplewiki-20121106-remote-media-1.tar | idump    | valid   |         1 |
+-----------------------------------------+----------+---------+-----------+
```

5.1.9.12 fsm-file-list-missing

Purpose: Identify any image files that are missing and make a list of URLs from which these files may be downloaded.

Motivation: Minimize the number of missing images.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+---------+-----------+
| name                                    | type     | state   | semaphore |
+-----------------------------------------+----------+---------+-----------+
| simplewiki                              | database | done    |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done    |         1 |
| simplewiki-20121106-local-media         | ichunk   | start   |         1 |
| simplewiki-20121106-remote-media        | ichunk   | start   |         1 |
| simplewiki-20121106-pages-articles.xml  | xml      | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | done    |         1 |
+-----------------------------------------+----------+---------+-----------+
```

FSM function: The fsm-file-list-missing function has two cases:

- for a local-media ichunk, it forms a list by taking the filenames found both in simplewiki.imagelinks.il_to and in simplewiki.image.img_name, less the filenames found in the directory tree under ./wikipedia/simple/, or
- for a remote-media ichunk, it forms a list by taking the filenames found both in simplewiki.imagelinks.il_to and in commonswiki.image.img_name, less the filenames of images stored in the directory tree under ./wikipedia/commons/.

The function outputs a file, the ichunk, containing URLs where the missing image files may be found.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+---------+-----------+
| name                                    | type     | state   | semaphore |
+-----------------------------------------+----------+---------+-----------+
| simplewiki                              | database | done    |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done    |         1 |
| simplewiki-20121106-local-media         | ichunk   | created |         1 |
| simplewiki-20121106-remote-media        | ichunk   | start   |         1 |
| simplewiki-20121106-pages-articles.xml  | xml      | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | done    |         1 |
+-----------------------------------------+----------+---------+-----------+
```

5.1.9.13 fsm-file-parse

Purpose: Scrape a checksum file for the names of xdump and sdump files.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------------+----------+-------+-----------+
| name                           | type     | state | semaphore |
+--------------------------------+----------+-------+-----------+
| simplewiki                     | database | done  |         1 |
| simplewiki-20121106-md5sums.txt | checksum | valid |         1 |
+--------------------------------+----------+-------+-----------+
```

FSM function: The fsm-file-parse function reads the checksum file and looks up the name of the sdump and xdump files, which are then entered into the wpmirror.file database table.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+---------------------------------------------+----------+---------+-----------+
| name                                        | type     | state   | semaphore |
+---------------------------------------------+----------+---------+-----------+
| simplewiki                                  | database | done    |         1 |
| simplewiki-20121106-categorylinks.sql.gz    | sdump    | start   |         1 |
| simplewiki-20121106-category.sql.gz         | sdump    | start   |         1 |
| simplewiki-20121106-externallinks.sql.gz    | sdump    | start   |         1 |
| simplewiki-20121106-imagelinks.sql.gz       | sdump    | start   |         1 |
| simplewiki-20121106-image.sql.gz            | sdump    | start   |         1 |
| simplewiki-20121106-interwiki.sql.gz        | sdump    | start   |         1 |
| simplewiki-20121106-iwlinks.sql.gz          | sdump    | start   |         1 |
| simplewiki-20121106-langlinks.sql.gz        | sdump    | start   |         1 |
| simplewiki-20121106-md5sums.txt             | checksum | pending |         1 |
| simplewiki-20121106-oldimage.sql.gz         | sdump    | start   |         1 |
| simplewiki-20121106-pagelinks.sql.gz        | sdump    | start   |         1 |
| simplewiki-20121106-page_props.sql.gz       | sdump    | start   |         1 |
| simplewiki-20121106-page_restrictions.sql.gz | sdump    | start   |         1 |
| simplewiki-20121106-pages-articles.xml.bz2  | xdump    | start   |         1 |
| simplewiki-20121106-protected_titles.sql.gz | sdump    | start   |         1 |
| simplewiki-20121106-redirect.sql.gz         | sdump    | start   |         1 |
| simplewiki-20121106-stub-articles.xml.gz    | xdump    | start   |         1 |
| simplewiki-20121106-templatelinks.sql.gz    | sdump    | start   |         1 |
+---------------------------------------------+----------+---------+-----------+
```

5.1.9.14 fsm-file-remove

Purpose: Delete a file from the file system.

Motivation: Files are deleted to save disk space.

Type	State	Comment
checksum	pending	Deleted to save disk space
xdump	pending	Deleted to save disk space
xml	pending	Deleted to save disk space
xchunk	pending	Deleted to save disk space
sdump	pending	Deleted to save disk space
sql	pending	Deleted to save disk space
schunk	pending	Deleted to save disk space
dchunk	pending	Deleted to save disk space
idump	pending	Deleted to save disk space
ichunk	pending	Deleted to save disk space

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+---------+-----------+
| name                                                            | type     | state   | semaphore |
+-----------------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml | xchunk   | start   |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml | xchunk   | start   |         1 |
| ...                                                             | ...      | ...     |         1 |
| simplewiki-20121106-pages-articles.xml                          | xml      | pending |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                      | xdump    | done    |         1 |
+-----------------------------------------------------------------+----------+---------+-----------+
```

FSM function: The fsm-file-remove function delete the file from the file system.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+-------+-----------+
| simplewiki                                                      | database | done  |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done  |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml | xchunk   | start |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml | xchunk   | start |         1 |
| ...                                                             | ...      | ...   |         1 |
| simplewiki-20121106-pages-articles.xml                          | xml      | done  |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                      | xdump    | done  |         1 |
+-----------------------------------------------------------------+----------+-------+-----------+
```

5.1.9.15 fsm-file-split-sdump

Purpose: Divide a large sdump file into smaller schunk files (of 10 INSERTs each).

Motivation: The sdumps for the largest wikipedias contain millions of records, which if processed all in one go, would take weeks to complete—if nothing were to go wrong. In fact one always encounters various kinds of failure along the way.

Therefore, for robustness and simplicity, both for mirroring and monitoring, WP-MIRROR splits large sql files into smaller schunks, each containing just 10 INSERTs (default), where each INSERT statement may contain VALUES for 1000 records. The successful processing of all but a few schunks is achievable.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+-------+-----------+
| name                                    | type     | state | semaphore |
+-----------------------------------------+----------+-------+-----------+
| simplewiki                              | database | done  |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done  |         1 |
| simplewiki-20121106-categorylinks.sql.gz | sdump    | valid |         1 |
+-----------------------------------------+----------+-------+-----------+
```

FSM function: The fsm-file-split-sdump function divides the sql.gz file into smaller schunk files, which are then entered into the wpmirror.file database table:

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+---------+-----------+
| name                                                            | type     | state   | semaphore |
+-----------------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done    |         1 |
| simplewiki-20121106-categorylinks-p000000000-c000000010.sql     | schunk   | start   |         1 |
| simplewiki-20121106-categorylinks.sql.gz                        | sdump    | pending |         1 |
+-----------------------------------------------------------------+----------+---------+-----------+
```

5.1.9.16 fsm-file-split-xml

Purpose: Divide a large xdump file into smaller xchunk files (of 1000 articles each).

Motivation: The xdumps for the largest wikipedias contain millions of articles, which if processed all in one go, would take weeks to complete—if nothing were to go wrong. In fact one always encounters various kinds of failure along the way. In part, this is because, for many years (as of 2012), the Wikimedia Foundation has been unable to generate error free xdumps.

Therefore, for robustness and simplicity, both for mirroring and monitoring, WP-MIRROR splits xdump files into thousands of smaller xml files, called xchunks, each containing just 1000 articles (default). The successful processing of all but a few xchunks is achievable.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+-------+-----------+
| name                                    | type     | state | semaphore |
+-----------------------------------------+----------+-------+-----------+
| simplewiki                              | database | done  |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done  |         1 |
| simplewiki-20121106-pages-articles.xml  | xml      | valid |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | done  |         1 |
+-----------------------------------------+----------+-------+-----------+
```

FSM function: The fsm-file-split-xml function divides the xml file into about 160 (as of 2012) smaller xchunk files, which are then entered into the wpmirror.file database table:

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+---------+-----------+
| name                                                            | type     | state   | semaphore |
+-----------------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml    | xchunk   | start   |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml    | xchunk   | start   |         1 |
| ...                                                             | xchunk   | start   |         1 |
| simplewiki-20121106-pages-articles.xml                          | xml      | pending |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                      | xdump    | done    |         1 |
+-----------------------------------------------------------------+----------+---------+-----------+
```

When mirroring the en wikipedia, the fsm-file-split-xml function will insert over ten thousand rows, which the FSM will process *seriatim*.

5.1.9.17 fsm-file-validate

Purpose: Compute the md5sum of the xdump file and compare it to the md5sum listed in the checksum file.

Motivation: The largest xdump files are several gigabytes. It is important to verify the integrity of the download process.

FSM before: The last row will be processed with the fsm-file-validate function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+---------+-----------+
| name                                    | type     | state   | semaphore |
+-----------------------------------------+----------+---------+-----------+
| simplewiki                              | database | done    |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | created |         1 |
+-----------------------------------------+----------+---------+-----------+
```

FSM function: The fsm-file-validate function computes the md5sum of the xdump file, and compares it to the md5sum that was provided in the checksum file.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+-------+-----------+
| name                                    | type     | state | semaphore |
+-----------------------------------------+----------+-------+-----------+
| simplewiki                              | database | done  |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done  |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | valid |         1 |
+-----------------------------------------+----------+-------+-----------+
```

5.1.9.18 fsm-file-wget

Purpose: Download image files from the Wikimedia Foundation.

Motivation: The largest wikipedias reference millions of image files. These are listed in the *xx*wiki.imagelinks table. If processed all in one go, downloading would take weeks to complete—if nothing were to go wrong. In fact one always encounters various kinds of failure along the way. In part, this is because many image file names contain control characters (such as ampersand, apostrophe, asterisk, backquote, brackets, braces, percent, quote, etc.) which make trouble for shell scripts and for database management systems.

Images are stored under a directory tree like:

```
root-shell# cd /var/lib/mediawiki/images/
root-shell# find -type d
./wikipedia
./wikipedia/commons
./wikipedia/commons/0
./wikipedia/commons/0/00
...
./wikipedia/commons/thumb
./wikipedia/simple
./wikipedia/simple/0
./wikipedia/simple/0/00
...
./wikipedia/simple/tmp
./wikipedia/simple/math
./wikipedia/simple/thumb
./wiktionary
./wiktionary/simple
./wiktionary/simple/0
./wiktionary/simple/0/00
...
./wiktionary/simple/tmp
./wiktionary/simple/math
./wiktionary/simple/thumb
```

For robustness, both for mirroring and monitoring, WP-MIRROR creates one ichunk for each image subdirectory, and then runs them one at a time, by forking a separate sub-process for each ichunk. The successful processing of all but a few ichunks is achievable.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+---------+-----------+
| name                                    | type     | state   | semaphore |
+-----------------------------------------+----------+---------+-----------+
| simplewiki                              | database | done    |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done    |         1 |
| simplewiki-20121106-local-media         | ichunk   | valid   |         1 |
| simplewiki-20121106-remote-media        | ichunk   | valid   |         1 |
| simplewiki-20121106-pages-articles.xml  | xml      | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | done    |         1 |
+-----------------------------------------+----------+---------+-----------+
```

FSM function: The fsm-file-wget function is quite simple. It simply invokes wget to download image files from the URLs listed in the ichunk.

In this example, the ichunk simplewiki-yyyymmdd-local-media-0-00 stores images under ./wikipedia/simple/0/00/, whereas the ichunk simplewiki-yyyymmdd-remote-media-0-00 stores images under ./wikipedia/commons/0/00/.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+---------+-----------+
| name                                    | type     | state   | semaphore |
+-----------------------------------------+----------+---------+-----------+
| simplewiki                              | database | done    |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done    |         1 |
| simplewiki-20121106-local-media         | ichunk   | pending |         1 |
| simplewiki-20121106-remote-media        | ichunk   | valid   |         1 |
| simplewiki-20121106-pages-articles.xml  | xml      | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2 | xdump | done    |         1 |
+-----------------------------------------+----------+---------+-----------+
```

5.1.9.19 fsm-file-xml2sql

Purpose: Convert an xchunk file into schunk files.

Motivation: An xchunk can not be directly loaded into the database. fsm-file-xml2sql uses mwxml2sql which quickly converts an xchunk into three schunks.

FSM before: The third row will be processed with the fsm-file-xml2sql function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------+----------+-------+-----------+
| name                                                      | type     | state | semaphore |
+-----------------------------------------------------------+----------+-------+-----------+
| simplewiki                                                | database | done  |         1 |
| simplewiki-20121106-md5sums.txt                           | checksum | done  |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml | xchunk | valid |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml | xchunk | valid |         1 |
| ...                                                       | xchunk   | valid |         1 |
| simplewiki-20121106-pages-articles.xml                    | xml      | done  |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                | xdump    | done  |         1 |
+-----------------------------------------------------------+----------+-------+-----------+
```

FSM function: The fsm-file-xml2sql function converts the xchunk file, into three schunk files.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------+----------+---------+-----------+
| name                                                      | type     | state   | semaphore |
+-----------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                           | checksum | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml | xchunk | pending |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml | xchunk | valid   |         1 |
| ...                                                       | xchunk   | valid   |         1 |
| simplewiki-20121106-pages-articles.xml                    | xml      | pending |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                | xdump    | done    |         1 |
| simplewiki-20121106-page-p000000000-c000001000.sql        | schunk   | start   |         1 |
| simplewiki-20121106-revision-p000000000-c000001000.sql    | schunk   | start   |         1 |
| simplewiki-20121106-text-p000000000-c000001000.sql        | schunk   | start   |         1 |
+-----------------------------------------------------------+----------+---------+-----------+
```

5.1.9.20 fsm-images-directory

Purpose: Create directory tree for images.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------------+----------+---------+-----------+
| name                           | type     | state   | semaphore |
+--------------------------------+----------+---------+-----------+
| simplewiki                     | database | done    |         1 |
| simplewiki-20121106-md5sums.txt | checksum | done    |         1 |
| simplewiki-images              | ichunk   | pending |         1 |
+--------------------------------+----------+---------+-----------+
```

FSM function: The fsm-images-directory function creates /var/lib/mediawiki/images/wikipedia/simple/ and several sub-directories.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------------+----------+-------+-----------+
| name                           | type     | state | semaphore |
+--------------------------------+----------+-------+-----------+
| simplewiki                     | database | done  |         1 |
| simplewiki-20121106-md5sums.txt | checksum | done  |         1 |
| simplewiki-images              | images   | valid |         1 |
+--------------------------------+----------+-------+-----------+
```

5.1.9.21 fsm-no-op

Purpose: Occupy an otherwise empty space in the *type-state-function* table. fsm-no-op simply returns with the return code done.

5.1.9.22 fsm-table-add-index

Purpose: ADD secondary indices (KEYs) to a database table.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------+----------+---------+-----------+
| name                     | type     | state   | semaphore |
+--------------------------+----------+---------+-----------+
| simplewiki               | database | pending |         1 |
| simplewiki.categorylinks | table    | valid   |         1 |
| ...                      | table    | ...     |         1 |
+--------------------------+----------+---------+-----------+
```

FSM function: fsm-table-add-index ADDs secondary indices by executing an SQL command like:

```
mysql> ALTER TABLE `simplewiki`.`categorylinks`
    -> ADD KEY `cl_sortkey` (`cl_to`,`cl_type`,`cl_sortkey`,`cl_from`),
    -> ADD KEY `cl_timestamp` (`cl_to`,`cl_timestamp`),
    -> ADD KEY `cl_collation` (`cl_collation`);
```

Secondary indices (KEYs) can greatly improve the SELECT performance of a table.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------+----------+---------+-----------+
| name                     | type     | state   | semaphore |
+--------------------------+----------+---------+-----------+
| simplewiki               | database | pending |         1 |
| simplewiki.categorylinks | table    | pending |         1 |
| ...                      | table    | ...     |         1 |
+--------------------------+----------+---------+-----------+
```

5.1.9.23 fsm-table-block-size

Purpose: Compress on-disk storage for given table. See §G.8, Experiments with InnoDB data compression.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------+----------+---------+-----------+
| name                     | type     | state   | semaphore |
+--------------------------+----------+---------+-----------+
| simplewiki               | database | pending |         1 |
| simplewiki.categorylinks | table    | start   |         1 |
| ...                      | table    | ...     |         1 |
+--------------------------+----------+---------+-----------+
```

FSM function: fsm-table-block-size compresses on-disk storage to fit into smaller InnoDB page sizes. This presupposes that InnoDB storage format is Barracuda. fsm-table-block-size executes an SQL command like:

```
mysql> ALTER TABLE `simplewiki`.`categorylinks`
    -> ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4;
```

Compression improve the SELECT performance of a table.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------+----------+---------+-----------+
| name                     | type     | state   | semaphore |
+--------------------------+----------+---------+-----------+
| simplewiki               | database | pending |         1 |
| simplewiki.categorylinks | table    | created |         1 |
| ...                      | table    | ...     |         1 |
+--------------------------+----------+---------+-----------+
```

5.1.9.24 fsm-table-drop-index

Purpose: DROP secondary indices (KEYs) to a database table.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------+----------+---------+-----------+
| name                     | type     | state   | semaphore |
+--------------------------+----------+---------+-----------+
| simplewiki               | database | pending |         1 |
| simplewiki.categorylinks | table    | created |         1 |
| ...                      | table    | ...     |         1 |
+--------------------------+----------+---------+-----------+
```

FSM function: fsm-table-drop-index DROPs secondary indices (KEYs) by executing an SQL command like:

```
mysql> ALTER TABLE `simplewiki`.`categorylinks`
    -> DROP KEY `cl_sortkey`,
    -> DROP KEY `cl_timestamp`,
    -> DROP KEY `cl_collation`;
```

DROPping secondary indices (KEYs) can greatly improve the INSERT performance of a table.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------+----------+---------+-----------+
| name                     | type     | state   | semaphore |
+--------------------------+----------+---------+-----------+
| simplewiki               | database | pending |         1 |
| simplewiki.categorylinks | table    | valid   |         1 |
| ...                      | table    | ...     |         1 |
+--------------------------+----------+---------+-----------+
```

5.1.10 Finalizing

```
-----finalizing-begin-------------------------
[ ok ]clear-pidfile
[ ok ]log-stop
```

5.1.10.1 clear-pidfile

```
...
[ ok ]clear-pidfile
...
```

WP-MIRROR deletes the file /var/run/wp-mirror.pid.

5.1.10.2 log-stop

```
...
[ ok ]log-stop
...
```

WP-MIRROR stops writing messages to /var/log/wp-mirror.log.

5.1.11 Initializing (Obsolete)

5.1.11.1 assert-images-bad-directory-or-create-p (Obsolete)

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-images-bad-directory-or-create-p
```

Thousands of downloaded image files are corrupt. The two leading causes appear to be: incomplete download and proxy errors. These images must be culled from the images directory tree. However, as a matter of safety, it is better to sequester than to delete corrupt files.

WP-MIRROR asserts (and if necessary creates) a directory tree in which the corrupt image files are sequestered for later inspection. This directory is /var/lib/mediawiki/images/bad-images/ (default).

5.1.12 The fsm-* Functions (Obsolete)

5.1.12.1 fsm-file-import (Obsolete)

Obsolete: WP-MIRROR 0.6 saw the introduction of fsm-file-xml2sql which invokes mwxml2sql—a utility announced in 2013-Feb by Ariel T. Glenn at the WMF. The new utility proved to have some performance advantage over importDump.php which fsm-file-import invokes.

Purpose: To have MediaWiki import the articles into the *xx*wiki database (where *xx* stands for the language-code).

Motivation: The xdumps for the largest wikipedias contain millions of articles, which if processed all in one go, would take weeks to complete—if nothing were to go wrong. In fact one always encounters various kinds of failure along the way. In part, this is because, for many years, the Wikimedia Foundation has been unable to generate error free dumps.

Table 5.4: Finite State Machine Functions (Obsolete)

Function	Intr	Rem
fsm-file-import	0.3	0.6
fsm-file-scrape	0.3	0.6
fsm-file-shell	0.3	0.6
fsm-file-split-sql	0.3	0.6
fsm-images-count	0.3	0.6
fsm-images-chown	0.3	0.6
fsm-images-rebuild	0.3	0.6
fsm-images-validate	0.3	0.6

Therefore, for robustness and simplicity, both for mirroring and monitoring, WP-MIRROR splits xdump files into thousands of smaller xml files, called xchunks, each containing just 1000 articles (default). WP-MIRROR runs them by forking a separate sub-process for each xchunk. The successful processing of all but a few xchunks is achievable.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+---------+-----------+
| name                                                            | type     | state   | semaphore |
+-----------------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.sh     | ichunk   | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml    | xchunk   | pending |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.sh     | ichunk   | done    |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml    | xchunk   | pending |         1 |
| ...                                                             | ...      | ...     |         1 |
| simplewiki-20121106-pages-articles.xml                          | xml      | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                      | xdump    | done    |         1 |
+-----------------------------------------------------------------+----------+---------+-----------+
```

FSM function: The fsm-file-import invokes

```
root-shell# /usr/bin/php /usr/share/mediawiki/maintenance/importDump_farm.php \
simplewiki-20121106-pages-articles-p000000000-c000001000.xml
```

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+---------+-----------+
| name                                                            | type     | state   | semaphore |
+-----------------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.sh     | ichunk   | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml    | xchunk   | done    |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.sh     | ichunk   | done    |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml    | xchunk   | pending |         1 |
| ...                                                             | ...      | ...     |         1 |
| simplewiki-20121106-pages-articles.xml                          | xml      | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                      | xdump    | done    |         1 |
+-----------------------------------------------------------------+----------+---------+-----------+
```

5.1.12.2 fsm-file-scrape (Obsolete)

Obsolete: WP-MIRROR 0.6 saw the introduction of fsm-file-list-missing which has performance and reliability advantages over fsm-file-scrape.

Purpose: Given an xchunk, scrape it for the names of image files (if any), then write a shell script (to be used later for downloading the image files) to an ichunk.

FSM before: The first row below will be processed with the fsm-file-scrape function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+---------+-----------+
| name                                                            | type     | state   | semaphore |
+-----------------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml | xchunk   | pending |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml | xchunk   | start   |         1 |
| ...                                                             | ...      | ...     |         1 |
| simplewiki-20121106-pages-articles.xml                          | xml      | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                      | xdump    | done    |         1 |
+-----------------------------------------------------------------+----------+---------+-----------+
```

FSM function: The fsm-file-scrape function reads the xchunk file and scrapes out the names of any image files. fsm-file-scrape then generates a shell script (to be used for downloading these image files) and writes it into an ichunk file, which is then entered into the wpmirror.file database table:

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+-------+-----------+
| name                                                            | type     | state | semaphore |
+-----------------------------------------------------------------+----------+-------+-----------+
| simplewiki                                                      | database | done  |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done  |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.sh  | ichunk   | start |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml | xchunk   | done  |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml | xchunk   | start |         1 |
| ...                                                             | ...      | ...   |         1 |
| simplewiki-20121106-pages-articles.xml                          | xml      | done  |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                      | xdump    | done  |         1 |
+-----------------------------------------------------------------+----------+-------+-----------+
```

5.1.12.3 fsm-file-shell (Obsolete)

Obsolete: WP-MIRROR 0.6 saw the introduction of fsm-file-wget which has performance and reliability advantages over fsm-file-shell.

Purpose: Execute ichunks to download image files from the Wikimedia Foundation.

Motivation: The xdumps for the largest wikipedias reference millions of image files, which if processed all in one go, would take weeks to complete—if nothing were to go wrong. In fact one always encounters various kinds of failure along the way. In part, this is because many image file names contain control characters (such as ampersand, apostrophe, asterisk, backquote, brackets, braces, percent, quote, etc.) which make trouble for shell scripts and for database management systems.

Therefor, for robustness, both for mirroring and monitoring, WP-MIRROR creates one ichunk for each xchunk, and then runs them one at a time, by forking a separate sub-process for each ichunk. The successful processing of all but a few ichunks is achievable.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+---------+-----------+
| name                                                            | type     | state   | semaphore |
+-----------------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.sh     | ichunk   | valid   |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml    | xchunk   | pending |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml    | xchunk   | pending |         1 |
| ...                                                             | ...      | ...     |         1 |
| simplewiki-20121106-pages-articles.xml                          | xml      | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                      | xdump    | done    |         1 |
+-----------------------------------------------------------------+----------+---------+-----------+
```

FSM function: The fsm-file-shell function is quite simple. It simply runs a shell script which in this case is an ichunk.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+---------+-----------+
| name                                                            | type     | state   | semaphore |
+-----------------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done    |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.sh     | ichunk   | pending |         1 |
| simplewiki-20121106-pages-articles-p000000000-c000001000.xml    | xchunk   | pending |         1 |
| simplewiki-20121106-pages-articles-p000001000-c000001000.xml    | xchunk   | pending |         1 |
| ...                                                             | ...      | ...     |         1 |
| simplewiki-20121106-pages-articles.xml                          | xml      | done    |         1 |
| simplewiki-20121106-pages-articles.xml.bz2                      | xdump    | done    |         1 |
+-----------------------------------------------------------------+----------+---------+-----------+
```

Design note. Introduced with version 0.2, WP-MIRROR offers a built-in alternative (default) to wikix. Compared to wikix, it takes much longer to generate ichunks. However the resulting ichunks are shorter, run faster, download more image files, and return fewer HTTP '400' and '404' errors. Moreover, whenever the Wikimedia Foundation posts a new xdump file and WP-MIRROR processes it, the built-in alternative first checks to see which image files are already downloaded before generating the ichunks. Thus the ichunks trend smaller and faster over time (containing just the new images and the old problem cases). Then again wikix does download some files that the built-in alternative does not.

Beginning with version 0.4, the built-in alternative generates shell scripts that are POSIX compliant. Previously, the generated shell scripts contained "bashisms" that do not work with the Debian Almquist SHell dash. dash is a POSIX-compliant implementation of /bin/sh, that has become the default /bin/sh for Debian distributions.

The use of wikix is deprecated, because it generates shell scripts that are not POSIX compliant.

The presence of "bashisms" can be detected by invoking

```
shell$ checkbashisms simplewiki-20121106-pages-articles-p000000000-c000001000.sh
```

WP-MIRROR 0.6 (development) saw the removal of wikix. The built-in alternative is now called fsm-file-scrape.

WP-MIRROR 0.6 (production) saw the replacement of fsm-file-scrape with fsm-file-list-missing which has performance and reliability advantages.

5.1.12.4 fsm-file-split-sql (Obsolete)

Obsolete: WP-MIRROR 0.6 saw the introduction of fsm-file-split-sdump which has the combined functionality of fsm-file-decompress and fsm-file-split-sql, but with some performance advantage (runs faster and uses less storage).

Purpose: Divide a large decompressed sdump file into smaller schunk files (of 10 INSERTs each).

Motivation: The sdumps for the largest wikipedias contain millions of records, which if processed all in one go, would take weeks to complete—if nothing were to go wrong. In fact one always encounters various kinds of failure along the way.

Therefore, for robustness and simplicity, both for mirroring and monitoring, WP-MIRROR splits large sql files into smaller schunks, each containing just 10 INSERTs (default), where each INSERT may contain VALUES for 1000 records. The successful processing of all but a few schunks is achievable.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------+----------+-------+-----------+
| name                                    | type     | state | semaphore |
+-----------------------------------------+----------+-------+-----------+
| simplewiki                              | database | done  |         1 |
| simplewiki-20121106-md5sums.txt         | checksum | done  |         1 |
| simplewiki-20121106-categorylinks.sql   | sql      | valid |         1 |
| simplewiki-20121106-categorylinks.sql.gz | sdump    | done  |         1 |
+-----------------------------------------+----------+-------+-----------+
```

FSM function: The fsm-file-split-sql function divides the sql file into smaller schunk files, which are then entered into the wpmirror.file database table:

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----------------------------------------------------------------+----------+---------+-----------+
| name                                                            | type     | state   | semaphore |
+-----------------------------------------------------------------+----------+---------+-----------+
| simplewiki                                                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt                                 | checksum | done    |         1 |
| simplewiki-20121106-categorylinks-p000000000-c000001000.sql | schunk   | start   |         1 |
| simplewiki-20121106-categorylinks.sql                           | sql      | pending |         1 |
| simplewiki-20121106-categorylinks.sql.gz                        | sdump    | done    |         1 |
+-----------------------------------------------------------------+----------+---------+-----------+
```

5.1.12.5 fsm-images-count (Obsolete)

Obsolete: WP-MIRROR 0.6 saw a rewrite of the :monitor modes, which no longer present image counts and therefore no longer invoke fsm-images-count.

Purpose: Count the images actually downloaded.

Motivation: This information is displayed by the :monitor instance. It give the user information regarding the image download process.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------------------------------+----------+-------+-----------+
| name                           | type     | state | semaphore |
+--------------------------------+----------+-------+-----------+
| simplewiki                     | database | done  |         1 |
| simplewiki-20121106-md5sums.txt | checksum | done  |         1 |
| simplewiki-images              | images   | start |         1 |
+--------------------------------+----------+-------+-----------+
```

FSM function: The fsm-images-count function counts files under /var/lib/mediawiki/images/wikipedia/simple
FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+---------------------------------+----------+---------+-----------+
| name                            | type     | state   | semaphore |
+---------------------------------+----------+---------+-----------+
| simplewiki                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt | checksum | done    |         1 |
| simplewiki-images               | images   | created |         1 |
+---------------------------------+----------+---------+-----------+
```

5.1.12.6 fsm-images-chown (Obsolete)

Obsolete: WP-MIRROR 0.6 saw the retirement of fsm-file-import which invoked importDump.php. importDump.php generated thumbs with root:root ownership. fsm-images-chown was used to change thumb ownership to www-data:www-data. Thumbs are now generated during browsing, and hence are created with www-data:www-data ownership.

Purpose: Set the ownership of image files, including the resized images (called thumbs) to www-data:www-data.

Motivation: WP-MIRROR runs as root, so any files created will initially have root:root ownership. However, root:root ownership prevents image resizing (making thumbs) during web browsing. If your browser renders a gray box where you expected a thumb, root:root ownership is the reason why.

FSM before:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+---------------------------------+----------+---------+-----------+
| name                            | type     | state   | semaphore |
+---------------------------------+----------+---------+-----------+
| simplewiki                      | database | done    |         1 |
| simplewiki-20121106-md5sums.txt | checksum | done    |         1 |
| simplewiki-images               | images   | pending |         1 |
+---------------------------------+----------+---------+-----------+
```

FSM function: The fsm-images-chown function sets ownership of all directories and files under /var/lib/mediawiki/images/wikipedia/simple/.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+---------------------------------+----------+-------+-----------+
| name                            | type     | state | semaphore |
+---------------------------------+----------+-------+-----------+
| simplewiki                      | database | done  |         1 |
| simplewiki-20121106-md5sums.txt | checksum | done  |         1 |
| simplewiki-images               | images   | done  |         1 |
+---------------------------------+----------+-------+-----------+
```

5.1.12.7 fsm-images-rebuild (Obsolete)

Obsolete: WP-MIRROR 0.6 saw the introduction of fsm-file-load-insert which directly loads the *xx*wiki.image table with performance and reliability advantages over fsm-images-rebuild.

Purpose: To have MediaWiki rebuild the *xx*wiki.image database table (where *xx* stands for a language code).

Motivation: When images are downloaded, the *xx*wiki.image database table will not be populated automatically. A maintenance script,

```
root-shell# /usr/bin/php /usr/share/mediawiki/maintenance/rebuildImages_farm.php \
simple.wpmirror.site
```

must be run for that purpose.

Design note. There are two choices about when to apply fsm-images-rebuild:

- If the *xx*wiki.image database table is populated *before* articles are imported, then resized images will be created during importation and stored under the /var/lib/mediawiki/images/thumb/ directory.
- Otherwise, images are resized, and thumbs stored, at the time a browser subsequently accesses the article (which delays the rendering process and tries the patience of the user).

5.1.12.8 fsm-images-validate (Obsolete)

Obsolete: WP-MIRROR 0.6 saw the introduction of fsm-file-wget which as such reliability advantages over fsm-file-shell as to obviate the need for fsm-images-validate. WP-MIRROR 0.6 also saw the removal of wpmirror.image which had been used by fsm-images-validate to store checksums of image filenames.

Purpose: Validate the integrity of each downloaded image file.

Motivation: The ichunks download images by invoking cURL. However, cURL sometimes downloads only a partial file. Therefore, all the downloaded image files are validated, and the corrupt ones sequestered to a bad-images directory for later inspection.

Design note. Validation is done by invoking

```
root-shell# gm identify -verbose path | grep identify
```

to look for error messages.

This is a CPU intensive process that one is loathe to repeat. WP-MIRROR, therefore, keeps a record of the validated image files. This record is kept in the wpmirror.image database table, so that all concurrent instances of WP-MIRROR can have access.

As a precaution wpmirror.image does not store the file name (which may have control characters, or be an SQL injection attack), rather it stores the md5sum of the file name. This hash is computed by invoking

```
shell$ env printf %s file | openssl dgst -md5
```

Note also that the first two hexadecimal digits of the md5sum are used by MediaWiki to create the directory tree under which the image files are stored. For example, hashing the image file named Arc_en_ciel.png yeilds

```
shell$ env printf %s Arc_en_ciel.png | openssl dgst -md5
(stdin)= 00135a44372c142bd509367a9f166733
```

and therefore, Arc_en_ciel.png would be stored under /var/lib/mediawiki/images/0/00/.

5.2 How --monitor Works

5.2.1 Start

Processing begins when you enter one of the following:

```
root-shell# wp-mirror --monitor
root-shell# wp-mirror --gui
root-shell# wp-mirror --screen
root-shell# wp-mirror --text
```

5.2.1.1 process-command-line-arguments-or-die

```
root-shell# wp-mirror --gui
[ ok ]process-command-line-arguments-or-die
...
```

WP-MIRROR first reads the command-line arguments. If there an error is found, you may expect:

```
root-shell# wp-mirror --foo
[....]process-command-line-arguments-or-die
Error: command line ...
unexpected option                : (foo)                                [fail]
For help, please try:
    $ wp-mirror --help
```

5.2.2 Initializing

```
root-shell# wp-mirror --gui
...
-----initializing-begin-----------------------
[info]set mode of operation to: MONITOR
[ ok ]assert-clisp-features-p
[ ok ]assert-utilities-p
[ ok ]assert-images-directory-or-create-p
[ ok ]assert-working-directory-or-create-p
[ ok ]assert-dbms-mysql-p
[ ok ]assert-dbms-mysql-config-debian-p
[ ok ]assert-dbms-credentials-debian-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-debian-p
[ ok ]assert-dbms-time-zone-or-load
[ ok ]assert-configuration-files-or-restore-default
[ ok ]process-configuration-files-or-die
...
```

5.2.2.1 assert-clisp-features-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-clisp-features-p
```

WP-MIRROR is written in Common Lisp (clisp 2.48). WP-MIRROR tests (asserts) that all the required libraries are loaded. For example, the CLX library is needed for running WP-MIRROR in --gui mode.

5.2.2.2 assert-utilities-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-utilities-p
```

WP-MIRROR relies on many utilities to do most of the work. WP-MIRROR asserts that they exist.

5.2.2.3 assert-images-directory-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-images-directory-or-create-p
```

WP-MIRROR creates a directory tree in which MediaWiki stores downloaded image files. This directory is /var/lib/mediawiki/images/ (default).

5.2.2.4 assert-working-directory-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-working-directory-or-create-p
```

WP-MIRROR creates a directory in which all its working files will be kept. This directory is /var/lib/mediawiki/images/wp-mirror/ (default).

5.2.2.5 assert-dbms-mysql-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-mysql-p
```

MediaWiki is able to use: MySQL, postgres, sqlite, mssql, and ibm_db2. WP-MIRROR, however, is only able to use the MySQL database management system (DBMS). If MySQL is not available, then WP-MIRROR exits.

5.2.2.6 assert-dbms-mysql-config-debian-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-mysql-config-debian-p
```

WP-MIRROR looks for Debian's database credentials, which are stored in /etc/mysql/debian.cnf. Debian's database account has root privileges.

5.2.2.7 assert-dbms-credentials-debian-or-scrape-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-credentials-debian-or-scrape-p
```

WP-MIRROR reads /etc/mysql/debian.cnf, and parses the file for Debian's MySQL user account and password. The file looks something like:

```
# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host     = localhost
user     = debian-sys-maint
password = abcdefghijklmnop
socket   = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
host     = localhost
user     = debian-sys-maint
password = abcdefghijklmnop
socket   = /var/run/mysqld/mysqld.sock
basedir  = /usr
```

5.2.2.8 assert-dbms-connect-with-credentials-debian-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-connect-with-credentials-debian-p
```

WP-MIRROR tests (asserts) if it can use the Debian database credentials to access MySQL. If it cannot, WP-MIRROR exits.

5.2.2.9 assert-dbms-time-zone-or-load

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-time-zone-or-load
```

WP-MIRROR checks if the MySQL time zone tables are populated. If they are not, the time zone data will be loaded from /usr/share/zoneinfo/, and you will see messages like:

```
root-shell# wp-mirror --mirror
...
[....]assert-dbms-time-zone-or-load
looking for time-zone             : UTC                                [fail]
Warning: Unable to load '/usr/share/zoneinfo/Asia/Riyadh87' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/Asia/Riyadh88' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/Asia/Riyadh89' as time zone. Skipping it.
...
[ ok ]assert-dbms-time-zone-or-load
```

5.2.2.10 assert-configuration-files-or-restore-default

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-configuration-files-or-restore-default
```

If all the configuration files are in place, you will see the above.

If any configuration files are missing, you will see additional messages. If all configuration files are missing, which can happen after you use the --restore-default option, you will see:

```
root-shell# wp-mirror --mirror
...
[....]assert-configuration-files-or-restore-default
[info]restoring default    : /etc/mediawiki/LocalSettings.php
[info]restoring default    : /etc/mediawiki/LocalSettings_wpmirror.php
[info]restoring default    : /etc/mysql/conf.d/wp-mirror.cnf
[info]restoring default    : /etc/wp-mirror/default.conf
[info]restoring default    : /etc/wp-mirror/local.conf
[info]restoring default    : /usr/share/mediawiki/maintenance/database_farm.sql
[info]restoring default    : /var/lib/mediawiki/favicon.ico
[info]restoring default    : /var/lib/mediawiki/wp-mirror.png
[ ok ] Stopping MySQL database server: mysqld.
[ ok ] Starting MySQL database server: mysqld . . ..
[info] Checking for tables which need an upgrade, are corrupt or were
not closed cleanly..
[ ok ]assert-configuration-files-or-restore-default
...
```

5.2.2.11 process-configuration-files-or-die

```
root-shell# wp-mirror --mirror
...
[ ok ]process-configuration-files-or-die
```

WP-MIRROR reads the local configuration file /etc/wp-mirror/local.conf. All WP-MIRROR parameters have default values, which may be seen in /etc/wp-mirror/default.conf. Parameter values set in /etc/wp-mirror/local.conf override the default values.

The parameter that most users will want to edit is,

```
(defparameter *mirror-language-code-list* '("simple"))
```

Users should make their configurations in /etc/wp-mirror/local.conf only. The /etc/wp-mirror/default.conf is for reference, and should never be edited.

5.2.3 Asserting Prerequisite Software

```
root-shell# wp-mirror --mirror
...
-----asserting-prerequisite-software-begin----
[ ok ]assert-dbms-accounts-or-create-p
[ ok ]assert-dbms-credentials-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
[ ok ]assert-dbms-connect-with-credentials-wikiuser-p
[ ok ]assert-dbms-grant-for-wikiadmin-p
[ ok ]assert-dbms-grant-for-wikiuser-p
[ ok ]assert-database-wpmirror-or-create-p
[ ok ]assert-mediawiki-dbms-credentials-p
```

These messages show the progress of asserting software prerequisites. We now describe each of them:

5.2.3.1 assert-dbms-accounts-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-accounts-or-create-p
```

WP-MIRROR asserts:

- the DBMS has an account for wikiadmin,
- the DBMS has an account for wikiuser,
- the file /etc/mediawiki/LocalSettings_account.php exists.

If the assertion fails, WP-MIRROR then:

- performs a DROP USER on any old wikiadmin and wikiuser accounts,
- generates random passwords,
- performs a CREATE USER to make new wikiadmin and wikiuser accounts, and
- writes the credentials to /etc/mediawiki/LocalSettings_account.php.

5.2.3.2 assert-dbms-credentials-or-scrape-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-credentials-or-scrape-p
```

WP-MIRROR reads /etc/mediawiki/LocalSettings_account.php and parses it to extract database credentials. The file looks something like:

```
root-shell# cat /etc/mediawiki/LocalSettings_account.php
<?php
# Automatically generated by WP-MIRROR for mediawiki scripts. DO NOT TOUCH!
$wgDBadminuser      = 'wikiadmin';
$wgDBadminpassword  = 'abcdefghijklm';
$wgDBuser           = 'wikiuser';
$wgDBpassword       = 'abcdefghijklm';
```

5.2.3.3 assert-dbms-connect-with-credentials-wikiadmin-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
```

WP-MIRROR, using the wikiadmin credentials, asserts that it is possible to connect to the MySQL server, and execute a simple query.

5.2.3.4 assert-dbms-connect-with-credentials-wikiuser-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-connect-with-credentials-wikiuser-p
```

WP-MIRROR, using the wikiuser credentials, asserts that it is possible to connect to the MySQL server, and execute a simple query.

5.2.3.5 assert-dbms-grant-for-wikiadmin-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-grant-for-wikiadmin-p
```

WP-MIRROR, using the Debian credentials, GRANTs elevated database privileges to wikiadmin. Namely,

- GRANT ALL PRIVILEGES ON each of the databases: wikidb, wpmirror, simplewiki (and any other database in a wikipedia farm), and
- GRANT PROCESS globally.

This last privilege is needed for monitoring the status of the InnoDB storage engine.

5.2.3.6 assert-dbms-grant-for-wikiuser-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-dbms-grant-for-wikiuser-p
```

WP-MIRROR, using Debian credential, GRANTs limited privileges to wikiuser. Namely, SELECT, INSERT, UPDATE, and DELETE privileges on wikidb, wpmirror, simplewiki (and any other database in a wikipedia farm).

5.2.3.7 assert-database-wpmirror-or-create-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-database-wpmirror-or-create-p
```

WP-MIRROR maintains state information that must be shared with one or more mirror and monitor processes, each of which must be Isolated (the 'I' in ACID) from each other. This state information also serves as a checkpoint, and must therefore be Durable (the 'D' in ACID) to facilitate resuming after interruption.

WP-MIRROR asserts (and if necessary creates) a database wpmirror, managed by the ACID compliant storage engine InnoDB, to hold its state information.

5.2.3.8 assert-mediawiki-dbms-credentials-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-mediawiki-dbms-credentials-p
```

Before WP-MIRROR can request MediaWiki maintenance scripts to import pages and images, all the database credentials must be in hand. WP-MIRROR asserts that they are. If they are not, WP-MIRROR exits.

5.2.4 Asserting Prerequisite Hardware

```
root-shell# wp-mirror --mirror
...
-----asserting-prerequisite-hardware-begin----
```

WP-MIRROR in monitor mode, does not assert any hardware prerequisites. This is because it does not write to disk, or occupy much memory.

5.2.5 Compile Status Report

```
root-shell# wp-mirror --mirror
...
-----monitor-mode-begin-----------------------
```

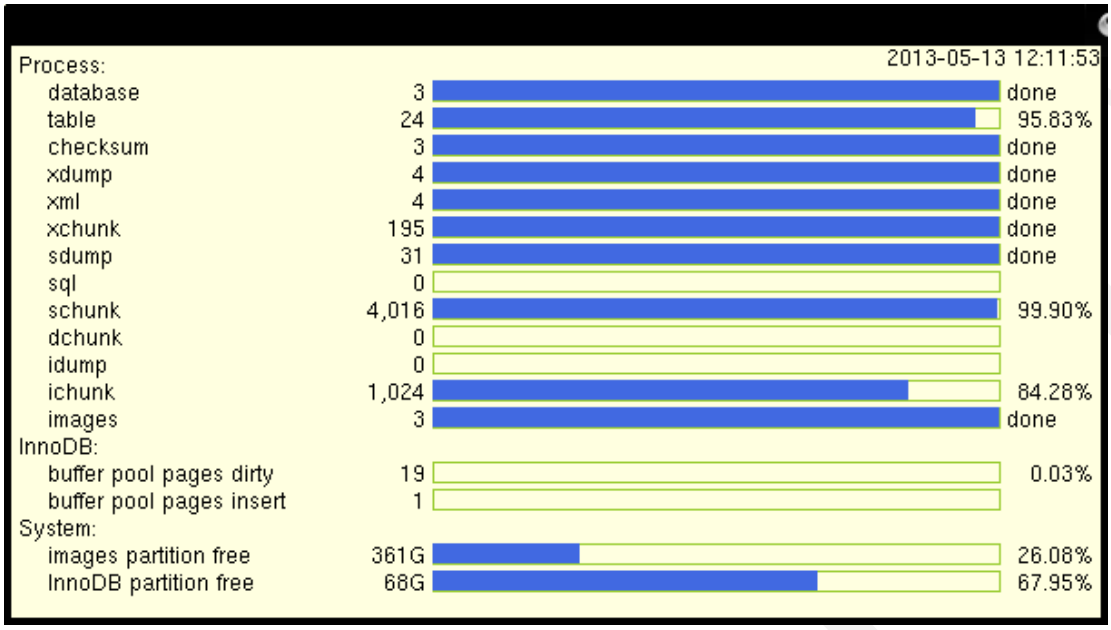
After the method of display is set, WP-MIRROR enters a loop.

Every 10 seconds (default), the monitoring process compiles a status report. This is done by running a set of SQL queries against the wpmirror database and against the InnoDB storage engine.

5.2.6 Display Progress Bars

After each status report is compiled, WP-MIRROR displays the results. Seven groups of progress bars are rendered as shown in Figure 5.3, WP-MIRROR Monitor Mode Progress Bars.

Figure 5.3: WP-MIRROR Monitor Mode Progress Bars



5.3 How --add Works

```
root-shell# wp-mirror --add xh
-----initializing-begin-----------------------
[info]set mode of operation to: ADD
[ ok ]log-start
[ ok ]assert-clisp-features-p
[ ok ]assert-utilities-p
[ ok ]assert-images-directory-or-create-p
[ ok ]assert-working-directory-or-create-p
[ ok ]assert-dbms-mysql-p
[ ok ]assert-dbms-mysql-install-db-p
[ ok ]assert-dbms-mysql-config-debian-p
[ ok ]assert-dbms-credentials-debian-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-debian-p
[ ok ]assert-dbms-time-zone-or-load
-----asserting-prerequisite-software-begin----
[ ok ]assert-dbms-accounts-or-create-p
[ ok ]assert-dbms-credentials-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
[ ok ]assert-dbms-grant-for-wikiadmin-p
[ ok ]assert-dbms-connect-with-credentials-wikiuser-p
[ ok ]assert-dbms-grant-for-wikiuser-p
[ ok ]assert-database-wpmirror-or-create-p
-----asserting-prerequisite-hardware-begin----
[ ok ]assert-internet-access-to-wikimedia-site-p
-----add-mode-begin---------------------------
[ ok ]fsm-boot
-----finalizing-begin-------------------------
[ ok ]log-stop
```

5.3.1 fsm-boot

WP-MIRROR in --add mode, simply adds the following row to the wpmirror.file database table:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+--------+----------+-------+-----------+
| name   | type     | state | semaphore |
+--------+----------+-------+-----------+
| xhwiki | database | start |         1 |
+--------+----------+-------+-----------+
```

If another WP-MIRROR process happens to be running in :first-mirror or :next-mirror mode, then it will build the new wikipedia. Adding, building, and dropping wikipedias can be done concurrently.

5.4 How --delete Works

```
root-shell# wp-mirror --delete xh
-----initializing-begin-----------------------
[info]set mode of operation to: DELETE
[ ok ]log-start
[ ok ]assert-dbms-mysql-p
[ ok ]assert-dbms-mysql-install-db-p
[ ok ]assert-dbms-mysql-config-debian-p
[ ok ]assert-dbms-credentials-debian-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-debian-p
-----asserting-prerequisite-software-begin----
[ ok ]assert-dbms-accounts-or-create-p
[ ok ]assert-dbms-credentials-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
[ ok ]assert-dbms-grant-for-wikiadmin-p
-----asserting-prerequisite-hardware-begin----
[ ok ]assert-internet-access-to-wikimedia-site-p
-----delete-mode-begin------------------------
[ ok ]delete-state-for-wiki xhwiki
[ ok ]delete-working-files-for-wiki xhwiki
[ ok ]delete-state-for-wiki xhwiktionary
[ ok ]delete-working-files-for-wiki xhwiktionary
-----finalizing-begin-------------------------
[ ok ]log-stop
```

5.4.1 delete-state-for-wiki

Purpose: Reset the Finite State Machine.

Motivation: When the user wishes to delete a wikipedia, or just start over, it is best to reset the Finite State Machine.

Method: DELETE all rows from filenamewpmirror.file for the given wiki.

5.4.2 delete-working-files-for-wiki

Purpose: Remove unneeded files from the working directory, /var/lib/mediawiki/images/wp-mirror/

Motivation: Clean up obsolete files, save disk space.

Method: Remove all files that have file names beginning with *wiki*, where *wiki* is the name of the wiki.

5.5 How --drop Works

```
root-shell# wp-mirror --drop xh
-----initializing-begin-----------------------
[info]set mode of operation to: DROP
[ ok ]log-start
[ ok ]assert-dbms-mysql-p
[ ok ]assert-dbms-mysql-install-db-p
[ ok ]assert-dbms-mysql-config-debian-p
[ ok ]assert-dbms-credentials-debian-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-debian-p
-----asserting-prerequisite-software-begin----
[ ok ]assert-dbms-accounts-or-create-p
[ ok ]assert-dbms-credentials-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
[ ok ]assert-dbms-grant-for-wikiadmin-p
-----asserting-prerequisite-hardware-begin----
[ ok ]assert-internet-access-to-wikimedia-site-p
-----drop-mode-begin--------------------------
[ ok ]delete-state-for-wiki xhwiki
[ ok ]drop-database-for-wiki-p xhwiki                          <---
[ ok ]delete-working-files-for-wiki xhwiki
[ ok ]delete-state-for-wiki xhwiktionary
[ ok ]drop-database-for-wiki-p xhwiktionary                    <---
[ ok ]delete-working-files-for-wiki xhwiktionary
-----finalizing-begin-------------------------
[ ok ]log-stop
```

5.5.1 drop-database-for-wiki

Purpose: Drop a wikipedia from the mirror.

Method: `DROP DATABASE wiki ;` where *wiki* is the name of the given wiki.

5.6 How --dump Works

```
root-shell# wp-mirror --dump xh
-----initializing-begin-----------------------
[info]set mode of operation to: DUMP
[ ok ]log-start
[ ok ]assert-dbms-mysql-p
[ ok ]assert-dbms-mysql-install-db-p
[ ok ]assert-dbms-mysql-config-debian-p
[ ok ]assert-dbms-credentials-debian-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-debian-p
-----asserting-prerequisite-software-begin----
[ ok ]assert-dbms-accounts-or-create-p
[ ok ]assert-dbms-credentials-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
[ ok ]assert-dbms-grant-for-wikiadmin-p
-----asserting-prerequisite-hardware-begin----
[ ok ]assert-internet-access-to-wikimedia-site-p
-----dump-mode-begin--------------------------
[ ok ]dump-database-for-wiki-p xhwiki
[ ok ]dump-database-for-wiki-p xhwiktionary
-----finalizing-begin-------------------------
[ ok ]log-stop
```

5.6.1 dump-database-for-wiki-p

Purpose: Backup a wikipedia.

Motivation: Prior to performing any database maintenance, one should make a backup.

Method: Execute mysqldump and write the results to *wiki*.sql in the working directory /var/lib/mediawiki/images/wp-mirror/.

5.7 How --profile Works

If no run number follows the --profile option,

```
root-shell# wp-mirror --profile
-----initializing-begin----------------------
[info]set mode of operation to: PROFILE
[ ok ]log-start
[ ok ]assert-dbms-mysql-p
[ ok ]assert-dbms-mysql-install-db-p
[ ok ]assert-dbms-mysql-config-debian-p
[ ok ]assert-dbms-credentials-debian-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-debian-p
-----asserting-prerequisite-software-begin----
[ ok ]assert-dbms-accounts-or-create-p
[ ok ]assert-dbms-credentials-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
[ ok ]assert-dbms-grant-for-wikiadmin-p
-----asserting-prerequisite-hardware-begin----
```

then WP-MIRROR provides a profile (in seconds) of the most recent five runs:

```
-----profile-mode-begin----------------------
+------------------------+----+----+-------+----+----+
| Function               | 73 | 72 |    71 | 70 | 69 |
+------------------------+----+----+-------+----+----+
| fsm-boot               |  4 |  4 |     3 |  4 |  3 |
| fsm-database-checksum  |  0 |  0 |     0 |  0 |  0 |
| fsm-database-create    |  0 |  0 |     0 |  0 |  0 |
| fsm-database-grant     |  0 |  0 |     0 |  0 |  0 |
| fsm-database-interwiki |  0 |  0 |     0 |  0 |  0 |
| fsm-file-count         |  0 |  0 |   215 |  4 | 68 |
| fsm-file-decompress    |  0 |  0 |    42 |  0 | 43 |
| fsm-file-digest        |  0 |  0 |    20 |  1 | 10 |
| fsm-file-directory     |  0 |  0 |     0 |  0 |  0 |
| fsm-file-download      |  0 |  0 |    59 |  0 | 60 |
| fsm-file-import        |  0 |  0 | 2,553 |  0 |  0 |
| fsm-file-parse         |  0 |  0 |     0 |  0 |  0 |
| fsm-file-remove        |  0 |  0 |     0 |  0 |  0 |
| fsm-file-scrape        |  0 |  0 |     5 |  0 |  5 |
| fsm-file-split         |  0 |  0 |    82 |  0 | 79 |
| fsm-file-validate      |  0 |  0 |     0 |  0 |  0 |
| fsm-no-op              |  0 |  0 |     0 |  0 |  0 |
| FSM-TOTAL              |  7 |  7 | 4,246 |  0 |  0 |
| GRAND-TOTAL            | 12 | 11 | 4,248 |  0 |  0 |
+------------------------+----+----+-------+----+----+
-----finalizing-begin------------------------
[ ok ]log-stop
```

If a run number (a positive integer) is provided,

```
root-shell# wp-mirror --profile 71
-----initializing-begin----------------------
[info]set mode of operation to: PROFILE
...
```

then WP-MIRROR provides a profile (in various units) of the given run:

```
-----profile-mode-begin----------------------
+------------------------+-----------+-----+-------+-------+------+-----+
| Function (run 71)      |      [ms] | [%] | [sec] | [min] | [hr] | [d] |
+------------------------+-----------+-----+-------+-------+------+-----+
| fsm-boot               |     2,646 |   0 |     3 |     0 |    0 |   0 |
| fsm-database-checksum  |       138 |   0 |     0 |     0 |    0 |   0 |
| fsm-database-create    |       166 |   0 |     0 |     0 |    0 |   0 |
| fsm-database-grant     |         8 |   0 |     0 |     0 |    0 |   0 |
| fsm-database-interwiki |       150 |   0 |     0 |     0 |    0 |   0 |
| fsm-file-count         |   214,668 |   5 |   215 |     4 |    0 |   0 |
| fsm-file-decompress    |    41,728 |   1 |    42 |     1 |    0 |   0 |
| fsm-file-digest        |    19,673 |   0 |    20 |     0 |    0 |   0 |
| fsm-file-directory     |       155 |   0 |     0 |     0 |    0 |   0 |
| fsm-file-download      |    58,527 |   1 |    59 |     1 |    0 |   0 |
| fsm-file-import        | 2,553,115 |  60 | 2,553 |    43 |    1 |   0 |
| fsm-file-parse         |       139 |   0 |     0 |     0 |    0 |   0 |
| fsm-file-remove        |       397 |   0 |     0 |     0 |    0 |   0 |
| fsm-file-scrape        |     4,899 |   0 |     5 |     0 |    0 |   0 |
| fsm-file-split         |    81,998 |   2 |    82 |     1 |    0 |   0 |
| fsm-file-validate      |       416 |   0 |     0 |     0 |    0 |   0 |
| fsm-no-op              |         6 |   0 |     0 |     0 |    0 |   0 |
| FSM-TOTAL              | 4,245,874 | 100 | 4,246 |    71 |    1 |   0 |
| GRAND-TOTAL            | 4,248,259 | 100 | 4,248 |    71 |    1 |   0 |
+------------------------+-----------+-----+-------+-------+------+-----+
-----finalizing-begin------------------------
[ ok ]log-stop
```

If a zero follows the --profile option, then all profile information is deleted, and run numbers begin again with one (1).

5.8 How --restore-default Works

```
root-shell# wp-mirror --restore-default
-----initializing-begin-----------------------
[info]set mode of operation to: RESTORE-DEFAULT
[ ok ]log-start
[ ok ]assert-dbms-mysql-p
[ ok ]assert-dbms-mysql-install-db-p
[ ok ]assert-dbms-mysql-config-debian-p
[ ok ]assert-dbms-credentials-debian-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-debian-p
-----asserting-prerequisite-software-begin----
------asserting-prerequisite-hardware-begin----

+-------------------------------------------------------------------------------+
| WARNING  WARNING  WARNING   WARNING   WARNING   WARNING   WARNING   WARNING |
|                                                                               |
| This option may DELETE more than you expect or want:                          |
| 1)  Delete config files      : /etc/mediawiki/LocalSettings.php               |
|                              : /etc/mediawiki/LocalSettings_wpmirror.php      |
|                              : /etc/mysql/conf.d/wp-mirror.cnf                |
|                              : /etc/wpmirror/local.conf                       |
|                              : /etc/wpmirror/default.conf                     |
|     Delete database template : /usr/share/mediawiki/mai../database_farm.sql  |
| 2)  Delete image files       : /var/lib/mediawiki/images/bad-images/          |
|                              : /var/lib/mediawiki/images/math/                |
|                              : /var/lib/mediawiki/images/thumb/               |
| 3)  Delete working files     : /var/lib/mediawiki/images/wp-mirror/           |
| 4)  Drop databases           : wikidb, *wiki, wpmirror                        |
|     Drop database users      : wikiadmin, wikiuser                            |
| The original default configuration is restored (mirror of 'simple' wiki).     |
|                                                                               |
| WARNING  WARNING  WARNING   WARNING   WARNING   WARNING   WARNING   WARNING |
+-------------------------------------------------------------------------------+

Do you wish to continue (yes/no)
```

If after the warning you enter 'no', then you will see the following:

```
-----finalizing-begin-------------------------
[ ok ]log-stop
```

If after the warning you enter 'yes', then you will see the following:

```
-----restore-default-mode-begin---------------
[ ok ]drop-database-p simplewiki
[ ok ]assert-dbms-drop-accounts-p
[ ok ]delete-directory-working
[info]deleting config file : /etc/mediawiki/LocalSettings.php
[info]deleting config file : /etc/mediawiki/LocalSettings_account.php
[info]deleting config file : /etc/mediawiki/LocalSettings_wpmirror.php
[info]deleting config file : /etc/mysql/conf.d/wp-mirror.cnf
[info]deleting config file : /etc/wp-mirror/default.conf
[info]deleting config file : /etc/wp-mirror/local.conf
[info]deleting config file : /usr/share/mediawiki/maintenance/database_farm.sql
[info]deleting config file : /var/lib/mediawiki/favicon.ico
[info]deleting config file : /var/lib/mediawiki/wp-mirror.png
[ ok ] Stopping MySQL database server: mysqld.
[ ok ] Starting MySQL database server: mysqld . ..
[info] Checking for tables which need an upgrade, are corrupt or were
not closed cleanly..
[info]done
-----finalizing-begin-------------------------
[ ok ]log-stop
```

5.9 How --update Works

```
root-shell# wp-mirror --update xh
-----initializing-begin-----------------------
[info]set mode of operation to: UPDATE
[ ok ]log-start
[ ok ]assert-dbms-mysql-p
[ ok ]assert-dbms-mysql-install-db-p
[ ok ]assert-dbms-mysql-config-debian-p
[ ok ]assert-dbms-credentials-debian-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-debian-p
-----asserting-prerequisite-software-begin----
[ ok ]assert-dbms-accounts-or-create-p
[ ok ]assert-dbms-credentials-or-scrape-p
[ ok ]assert-dbms-connect-with-credentials-wikiadmin-p
[ ok ]assert-dbms-grant-for-wikiadmin-p
-----asserting-prerequisite-hardware-begin----
[ ok ]assert-internet-access-to-wikimedia-site-p
-----update-mode-begin------------------------
[....]update-database-for-wiki-p xhwiki
mediawiki 1.19.2-2 Updater

Going to run database updates for xhwiki
Depending on the size of your database this may take a while!
...
[ ok ]update-database-for-wiki-p xhwiki
[....]update-database-for-wiki-p xhwiktionary
...
[ ok ]update-database-for-wiki-p xhwiktionary
[ ok ]log-start
-----finalizing-begin-------------------------
[ ok ]log-stop
```

5.10 How Scheduled Jobs Work

5.10.1 cron

During installation, WP-MIRROR sets up a weekly `cron` job.

The idea is to keep your mirror farm up to date. However, as a precaution, the file `/etc/cron.d/wp-mirror` first looks for the PID file `/var/run/wp-mirror.pid`. If the PID file is found, the cron job exits immediately. This is because presence of the PID file indicates that another instance of WP-MIRROR may be running. When mirroring a large wikipedia, it can happen that last week's instance is still running.

5.10.2 logrotate

During installation, WP-MIRROR sets up a daily `logrotate` job.

The idea is to avoid stuffing your `/var/` partition. WP-MIRROR instances running in mirror mode write copious logs to `/var/log/wp-mirror.log`. This log file is rotated daily by `/etc/logrotate.d/wp-mirror`. Rotated log files are compressed using `gzip`, kept for seven days, and then discarded.

If you notice that the log file is not being rotated as expected, you may force matters with the command:

```
root-shell# /usr/sbin/logrotate --force /etc/logrotate.d/wp-mirror
```

The log files are *not* world readable. This is because database user passwords are logged whenever an user invokes `wp-mirror --debug`.

```
root-shell# ls -l /var/log/wp-mirror.log*
-rw-r----- 1 root adm        0 Nov  8 07:43 /var/log/wp-mirror.log
-rw-r----- 1 root adm  1809060 Nov  7 15:32 /var/log/wp-mirror.log.1.gz
...
```

Appendix A

Change History

This appendix lists the changes from version to version in the WP-MIRROR source code.

Dates in section headings refer to the release date for that version.

A.1 Changes in Release 0.x

An overview of features added in WP-MIRROR can be found in §1.5, What Is New in WP-MIRROR. For a complete list of changes, please refer to the changelog sections for individual releases.

A.1.1 Changes in WP-MIRROR 0.6 (2013-12-10)

A.1.1.1 Functionality Added or Changed

- **Command-line option added**: The `--profile` option displays how much time the Finite State Machine functions are consuming. It may be run concurrently with adding, building, and dropping wikipedias from the mirror. This is used for performance analysis. Profiling data is stored in the `wpmirror.time` database table.
- **Command-line option enhanced**: The `--add`, `--drop`, `--delete`, `--dump`, and `--update` options now have enhanced functionality. As before, these options take just one argument. But now there are more choices for that argument:

```
wp-mirror --add    {wiki|language-code|project|all}
wp-mirror --delete {wiki|language-code|project|all|template}
wp-mirror --drop   {wiki|language-code|project|all|template}
wp-mirror --dump   {wiki|language-code|project|all|template}
wp-mirror --update {wiki|language-code|project|all|template}
```

  where *wiki* stands for a specific wiki (e.g. zuwiktionary), *language-code* stands for a language-code (e.g. zu), and *project* stands for the name of a project (e.g. wikibooks, wiktionary, etc.). For example:

```
root-shell# wp-mirror --add zuwiktionary   <-- add one wiki to the mirror
root-shell# wp-mirror --add zu             <-- add all isiZulu wikis
root-shell# wp-mirror --add wiktionary     <-- add all wiktionaries (100+ wikis)
root-shell# wp-mirror --add all            <-- add all wikis (need 30T storage)
```

- **Database schema added**: The `wpmirror.priority` table now contains the `*type-state-priority*` information. Used for sequencing tasks with greatly reduced FSM overhead.
- **Database schema added**: The `wpmirror.time` table contains FSM profile information.
- **Dependency added**: The `mediawiki-mwxml2sql` utility is used by `fsm-file-xml2sql` to convert `xchunk`s into sets of `schunk`s. `mediawiki-mwxml2sql` is now available as a DEB package (it has been packaged by the author of WP-MIRROR).
- **Dependency added**: The `rsync` utility now can be used to download dump files from sites that mirror the Wikimedia Foundation dumps.

- **Dependency added**: The tar utility is used to extract image dump tarballs.
- **FSM**: New file types have been added to the Finite State Machine: sdump, sql, schunk, dchunk, and idump. The first four are used to download and import SQL dump files. The last is used to download and extract image dump tarballs.
- **FSM**: The new fsm-file-split-sdump and fsm-file-load-insert functions improve the loading of records into MediaWiki database tables.

  WP-MIRROR 0.6 now downloads sdump files (one for each database table), which are decompressed and split into schunks, which in turn are loaded directly into database tables.

  The new functions obviate fsm-file-import and fsm-images-rebuild. fsm-file-import calls importDump.php which is comparitively slow, and fsm-images-rebuild calls rebuildImages.php which seems to have a memory leak.

  Impressive performance gains (xx% less time) are obtained. See §G.10, Experiments with mwxml2sql.
- **FSM**: The new fsm-file-xml2sql facilitates the loading of records into MediaWiki database tables. This function obviates the comparitively slow fsm-file-import which invokes importDump.php.

  fsm-file-xml2sql uses a new utility, mwxml2sql, to convert each xchunk into a set of three schunks (one each for the page, revision, and text database tables).

  The utility mwxml2sql was announced on 2013-Feb-21 by Ariel T. Glenn of the WMF, the code for which can be found by chasing a link found at https://meta.wikimedia.org/wiki/Data_dumps/Tools_for_importing.

  Impressive performance gains (xx% less time) are obtained. See §G.10, Experiments with mwxml2sql.
- **FSM**: The new functions fsm-file-extract, fsm-file-list-missing and fsm-file-wget completely overhaul the way image files are downloaded.

  Impressive performance gains (xx% less time) are obtained.
- **Image dumps**: Image dump tarballs are provided by the mirror site http://ftpmirror.your.org/pub/wikimedia/imagedumps/tarballs/fulls/.

  WP-MIRROR 0.6 can now download and extract these idumps. This feature greatly reduces the work performed by fsm-file-scrape, fsm-file-shell, and fsm-image-validate.

  Impressive performance gains (about 67% less time) are obtained.
- **Interwiki**: Previously, at the bottom of each article there was a mass of 'red links' (interlanguage links to articles in wikipedias that are not part of the mirror). These links have now been moved to the Navigation Bar, even though most of them are unusable.
- **Performance vs. durability**: The degree of Durability (the 'D' in ACID) of COMMITted transactions is now configurable. To achieve the greatest possible Durability, database transactions must be written to disk immediately after they are COMMITted. However, modest performance gains (22% less time) can be achieved by flushing transactions to disk once per second (default). This is done by setting the configuration file to read:

```
root-shell# cat /etc/mysql/conf.d/wp-mirror.cnf | grep trx
innodb_flush_log_at_trx_commit = 2 # default 1
```

  This means, in the case of a system failure, up to one second of transactions may be lost. This is not too tragic, because WP-MIRROR resumes from the last checkpoint, and repeats any lost transactions.

  WP-MIRROR 0.6 also introduces two new configuration parameters:

  - *system-hdd-write-cache* which disables/enables HDD write caching (default 1, enabled); and
  - *system-hdd-write-cache-flush* which enables flushing the HDD write cache immediately after each checkpoint (default nil).

  To achieve the greatest possible Durability (the 'D' in ACID) for its database transactions, set the InnoDB variable innodb_flush_log_at_trx_commit to 1, and set *system-hdd-write-cache* to nil.

  Modest performance gains (22% less time) are obtained using the defaults. See §G.3, fsm-file-import and Durability.

- **Performance with persistent connections**: fsm-file-shell is now obsolete. Image files are now downloaded with fsm-file-wget which invokes wget. wget as two important advantages:
  - wget (like cURL) supports "persistent connections" as defined in RFC2616, HTTP/1.1. This permits downloading several files using a single TCP connection, thereby reducing latency for all but the first file. Significant performance gains (64% less time) are obtained (see §G.4, fsm-file-shell and HTTP/1.1 Persistent Connections).
  - wget (unlike cURL) has an automatic retry feature. If a download does not complete, wget waits a while, then continues downloading the file from the offset where the interruption occurred.
- **Projects**: The Wikimedia Foundation has several projects besides the wikipedia. Other projects include: wikibooks, wikimedia, wikinews, wikiquote, wikisource, wikiversity, wikivoyage, and wiktionary. Most of these projects have language sub-domains (i.e. one wiki for each language).

  WP-MIRROR 0.6 can now mirror any set of wikis, not just ones from the wikipedia project. The default configuration now mirrors both the simple wikipedi and the simple wiktionary.
- **Reliability**: The downloading of dump files and image files from http sites is now far more reliable. This was accomplished by using wget, which has an automatic retry feature, instead of cURL, which often leaves partial files.
- **Sites**: In 2012, three organizations agreed to mirror the Wikimedia Foundation's dump files, and make them available using the http, ftp, and rsync protocols. WP-MIRROR can now be configured to download from any of them, using any of the protocols.

A.1.1.2 Bugs Fixed

- fsm-database-create: Sometimes failed to create database tables from database_farm.sql.
- **HDD write cache**: Configuration failed if database was installed on a Virtual Machine.
- **Partial download**: Downloaded files sometimes incomplete.
- **Restore-default**: Logotype did not restore.

A.1.1.3 Functionality Removed

- **FSM**: Use of fsm-file-import is removed from WP-MIRROR 0.6. fsm-file-import calls importDump.php which is comparitively slow. fsm-file-xml2sql, which calls the new (and much faster) mwxml2sql, obviates fsm-file-import.
- **FSM**: Use of fsm-file-scrape is removed from WP-MIRROR 0.6. Image file names are now taken from the *xx*wiki.imagelinks database table, which is loaded directly from an sdump.
- **FSM**: Use of fsm-file-shell is removed from WP-MIRROR 0.6. Image files are now downloaded using fsm-file-wget which performance and reliability advantages.
- **FSM**: Use of fsm-images-count is removed from WP-MIRROR 0.6. The :monitor mode was rewritten and no longer displays this information.
- **FSM**: Use of fsm-images-chown is removed from WP-MIRROR 0.6. Thumbs are now generated during browsing and are therefore created with the www-data:www-data ownership.
- **FSM**: Use of fsm-images-rebuild is removed from WP-MIRROR 0.6. fsm-images-rebuild calls rebuildImages.php which is slow and seems to have a memory leak. The image database table is now loaded directly from the corresponding schunks, which obviates fsm-images-rebuild.
- **FSM**: Use of fsm-images-validate is removed from WP-MIRROR 0.6. Improved download reliability obviates fsm-images-validate. As a consequence:
  - Use of the wpmirror.image database table is also removed from WP-MIRROR 0.6. fsm-images-validate stores its results in wpmirror.image.

  - Use of the /var/lib/mediawiki/images/bad-images/ directory is removed from WP-MIRROR 0.6. fsm-images-validate sequesters corrupt image files under that directory.

- **Wikix**: Use of wikix is removed from WP-MIRROR 0.6. The built-in alternative to wikix is now called fsm-file-scrape.

  Beginning with WP-MIRROR 0.4, wikix was deprecated, because it is not POSIX compliant. That is, wikix contains "bashisms" that do not work with the Debian Almquist SHell dash. dash is a POSIX-compliant implementation of /bin/sh, which has become the default /bin/sh for Debian.

A.1.2 Changes in WP-MIRROR 0.5 (2012-12-14)

A.1.2.1 Functionality Added or Changed

- **Command-line option added**: The --add *language-code* option allows the user to add a wikipedia to the mirror (in lieu of editing the *mirror-language-code-list* parameter in the configuration file /etc/wp-mirror/local.conf).
- **Command-line option added**: The --dump *language-code* option allows the user to dump the database of a wikipedia to the file /var/lib/mediawiki/images/wp-mirror/*xx*wiki.sql (where *xx* stands for the language-code). If the language-code is template, then the empty database wikidb is dumped to database_farm.sql.
- **Command-line option added**: The --info option lets the user see the value of every parameter can be configured in /etc/wp-mirror/local.conf.
- **Command-line option added**: The --update *language-code* option allows the user to update the database of a wikipedia to the latest MediaWiki database schema.
- **Concurrency**: WP-MIRROR 0.5 now permits concurrent adding, building, and dropping of wikipedias.
- **Concurrency**: The --gui option displays one window for each wikipedia in the mirror. WP-MIRROR 0.5 now permits the number of windows to increase or decrease dynamically as the user -adds and --drops wikipedias.
- **FSM**: A new file type has been added to the Finite State Machine: database.
- **FSM**: The functions that GRANT privileges, CREATE DATABASE, and establish interwiki (interlanguage) links; have been moved into the Finite State Machine. They are now called: fsm-database-grant, fsm-database-create, fsm-database-interwiki, respectively.
- **Interwiki**: Interlanguage links allow the user to easily switch from an article in one language to the corresponding article in another language. This is useful for the user who mirrors two or more wikipedias (e.g. the simple wikipedia and a native language wikipedia).
- **Logotype**: WP-MIRROR now has a logotype (see Figure A.1, WP-MIRROR Logotype). The logotype was composed using the GIMP.

Figure A.1: WP-MIRROR Logotype

Many image files of different resolution and format are now included:

  - /var/lib/mediawiki/favicon.ico is 16x16 pixel, and appears in the web browser Address Bar (the text box where you type the URL);
  - /var/lib/mediawiki/wp-mirror.png is 135x135 pixel to be compatible with the legacy skin Monobook, and appears as the logotype on http://wpmirror.site/;
  - /usr/share/icons/hicolor/*resolution*/apps/wp-mirror.png, where *resolution* is 16x16, 22x22, 24x24, 32x32, 48x48, 64x64, 128x128, and 256x256, and are available to the window manager;
  - /usr/share/pixmaps/wp-mirror.xpm is 32x32 pixel, and used by the Debian menu system.

- **Virtual host**: WP-MIRROR now sets up the virtual host http://wpmirror.site/ rather than http://mediawiki.site/. Consistency of naming was the motive. That is: cron job, documentation, logfile, logotype, packaging, software, and virtual host should all have very similar names.

A.1.2.2 Bugs Fixed

- **Dependency fixed**: The inkscape dependency was overspecified causing difficulty with installation on Ubuntu 12.10 Quantal.
- **False negative**: Some functions responsible for lengthy operations (e.g. drop a large database or delete a large directory) returned fail even though they succeeded. This was due to the functions sometimes returning before the underlying database or shell completed its task. These functions are now guarded by a sleep-while or sleep-until loop that periodically checks to see if the operation is complete before continuing.
- **Interwiki**: Interwiki links (interlanguage links actually) appear at the bottom of many pages. These appear as 'red links' (meaning 'page does not exist').

  Yet within a wikipedia farm, an article on say 'Nelson Mandela' appears in many languages. Therefore, the interwiki links from the article in one language to the corresponding articles in other languages (within the farm) should all work. Moreover, those links should be located in the Navigation Sidebar (usually on the left side under the 'Languages' menu).

  This turns out to be three issues:

  - **magic connector**: We want each interwiki link to connect to the corresponding article in a sister wikipedia (within the farm). Solution is to make sure that: 1) $wgInterwikiMagic is set to true (default), 2) $wgHideInterlanguageLinks is set to false (default), and 3) for each wikipedia, the language-code and URL of every other wikipedia, is entered into its interwiki database table. See http://www.mediawiki.org/wiki/Interwiki#Interwiki_links_to_other_languages.

```
mysql> SHOW CREATE TABLE xhwiki.interwiki\G
*************************** 1. row ***************************
       Table: interwiki
Create Table: CREATE TABLE `interwiki` (
  `iw_prefix` varbinary(32) NOT NULL,
  `iw_url` blob NOT NULL,
  `iw_api` blob NOT NULL,
  `iw_wikiid` varbinary(64) NOT NULL,
  `iw_local` tinyint(1) NOT NULL,
  `iw_trans` tinyint(4) NOT NULL DEFAULT '0',
  UNIQUE KEY `iw_prefix` (`iw_prefix`)
) ENGINE=InnoDB DEFAULT CHARSET=binary
1 row in set (0.00 sec)
```

  For example, if we mirror the xh and zu wikipedias, then the interwiki table of each database would need the URL of the other wikipedia:

```
mysql> INSERT INTO xhwiki.interwiki (iw_prefix, iw_url, iw_local, iw_trans,
    ->  iw_api, iw_wikiid)
    -> VALUES ('zu', 'http://zu.wpmirror.site/index.php/$1', 1, 0,
    ->  'http://zu.wpmirror.site/api.php', 'zuwiki');
mysql> INSERT INTO zuwiki.interwiki (iw_prefix, iw_url, iw_local, iw_trans,
    ->  iw_api, iw_wikiid)
    -> VALUES ('xh', 'http://xh.wpmirror.site/index.php/$1', 1, 0,
    ->  'http://xh.wpmirror.site/api.php', 'xhwiki');
```

  - **sidebar**: We want language links to appear in the sidebar, rather than in-text. Solution: make sure that: 1) `$wgHideInterlanguageLinks` is set to false (default), and 2) do not use the `polyglot` extension. See http://www.mediawiki.org/wiki/Extension:Polyglot.
  - **red links**: We want to suppress the remaining 'red links' (interwiki links to articles in languages that are not in our farm). This has not been solved in version 0.5.

- **Md5sum**: A very rare bug was caught. The fatal error:

```
*** -
      overflow during multiplication of large numbers
```

  was traced to the `md5sum` hash:

```
593726e290081349480708313641736 4
```

  The hash was correctly stored in the `wpmirror.file` database table as a `VARCHAR(32)`, but then read back as a number in exponential notation (notice the lone e).

  The ultimate cause is this: WP-MIRROR relies upon the `read` function in `common lisp` to distinguish numbers from strings, rather than checking the database schema.
- **Monitor**: The monitor gave a pair of 'database not found' error messages every 10 seconds if a wikipedia was dropped while the monitor was running.
- **Monitor**: The 'Image: importing' bar was blank.
- **Wikix**: The built-in alternative to `wikix` missed about one third of the image file names. What it did right was to scrape image file names from links:

```
... [[File:foo.png| ... ]] ...
... [[Image:bar.png| ... ]] ...
... [[Media:baz.png| ... ]] ...
```

  What it neglected to do was to scrape image file names from other sources, such as, `gallery`, `infobox`, `multiple image`, and `wide image` templates. See http://en.wikipedia.org/wiki/Wikipedia:Picture_tutorial.

A.1.3 Changes in WP-MIRROR 0.4 (2012-11-12)

A.1.3.1 Functionality Added or Changed

- **Command-line option added**: The `--restore-default` option offers users who mess up the configuration files, a way to start over. This option option drops all WP-MIRROR related databases, deletes all WP-MIRROR files (except images), and restores all WP-MIRROR related configuration files to the default (i.e. build a mirror of the simple wikipedia). This option is used frequently during coding and debugging.
- **Configuration automated**: For the dependency `MySQL`, previously, the configuration was done manually for each of two cases: laptop and desktop. WP-MIRROR now automatically configures `MySQL` for the laptop case which is appropriate for building the default mirror, the simple wikipedia.
- **Configuration automated**: For the dependency `MediaWiki`, previously, the configuration was done manually through a web interface. WP-MIRROR now automatically configures `MediaWiki`.
- **Dependency added**: Lisp libraries are now managed by `common-lisp-controller`.

- **Dependency added**: The CLX module which implements XLIB is now packaged separately from clisp. The new package is called clisp-module-clx.
- **Dependency added**: inkscape is now preferred for converting image files in SVG format into PNG format. The rsvg package is still listed as a dependency, but is no longer used as the default configuration.
- **Dependency added**: The mediawiki-extensions-math package has been added. The code that use LaTeX to format math equations into PNG image files, has been removed from MediaWiki 1.18 core, and is now packaged separately. The mediawiki-extensions-math package implements this functionality, as described in http://www.mediawiki.org/wiki/Extension:Math and in http://www.mediawiki.org/wiki/Manual:Enable_TeX.

  Note also that the *xx*wiki.math database has also been removed from the core, but can be added by running:

```
root-shell# /usr/bin/php /usr/share/mediawiki/maintenance/update_farm.php \
simple.wpmirror.site
```

- **Dependency added**: The tzdata package provides time-zone info that is now loaded into MySQL time zone tables.
- **Dependency removed**: The cl-asdf package is no longer listed as a dependency, because it is a dependency of the common-lisp-controller package.
- **Dependency removed**: The php5-suhosin package has been dropped from Debian GNU/Linux 7.0 (wheezy).
- **Document**: The WP-MIRROR Reference Manual (this document) is now available. It contains bookmarks, internal links, and URL links to make browsing easy. It is written in LaTeX, with document class memoir, and uses the hyperref package. This document is published in PDF format. Other formats such as HTML are being considered for a future release.
- **Document**: Previously, the README document was a long text file that could not easily be browsed. It has been truncated, and is now a stub that refers to the WP-MIRROR Reference Manual (this document).
- **Doc-base**: The WP-MIRROR Reference Manual (this document) is now automatically registered with doc-base. Users can now find this document on-line using utilities such as doc-central, dhelp and dwww.
- **Hardware**: WP-MIRROR now warns if disk space is likely to be inadequate for the default, a mirror of the simple wikipedia. Currently, this threshold is set at 60G.
- **Menufile**: WP-MIRROR is now automatically registered with menufile, so that is will appear in the Debian menu system.
- **Pbuilder**: The WP-MIRROR DEB package is built by pbuilder, which is now configured for Debian GNU/Linux 7.0 (wheezy).
- **Patch removed**: Import.php.patched was needed for MediaWiki 1.15. Now WP-MIRROR uses MediaWiki 1.19, and the patch is no longer needed.
- **Security**: WP-MIRROR now issues a warning if MySQL is insecure (e.g. if the root account has no password).
- **User interface**: Previously, the user running WP-MIRROR in mirror mode, saw copious messages scroll up the screen. Now these messages are far less verbose, and use color coding to indicate either an exit condition (fail, ok), or the importance of the message to the user (info, warn). The log files are still verbose and properly so.

A.1.3.2 Bugs Fixed

- **Hardware**: HDD write-cache disabling requires identifying the underlying hard disk. This failed for the simple case, such as when the table space /var/lib/mysql/ibdata0 was a file in a file system on top of a partition like /dev/sda6.
- **Ichunks**: Incorrect URLs were found in ichunks. These URLs contained a nil instead of a language code. For example,

```
IMAGEPATH=http://upload.wikimedia.org/wikipedia/nil/
COMMONSPATH=http://upload.wikimedia.org/wikipedia/commons/
```

should read

```
IMAGEPATH=http://upload.wikimedia.org/wikipedia/simple/
COMMONSPATH=http://upload.wikimedia.org/wikipedia/commons/
```

- **POSIX**: The C language program wikix generates shell scripts that download images files. These scripts contain 'bashisms' that do not work with the Debian Almquist SHell dash. dash is a POSIX compliant implementation of /bin/sh, which has become the default /bin/sh for Debian. Use of wikix is now deprecated.
- **POSIX**: The built-in alternative to wikix generated shell scripts, ichunks, that were not POSIX compliant. For example,

```
if [-a  $IMAGE./foo]; then
```

should read

```
if [-e  $IMAGE./foo]; then
```

- **POSIX**: Several shell commands, forked by WP-MIRROR for a variety of tasks, were not POSIX compliant. For example,

```
echo -n "foo" | openssl md5
```

should read

```
env printf %s "foo" | openssl dgst -md5
```

- **Images**: The initial estimates for the number image files was far too low.
- **Images**: When *mirror-image-download-p* was set to nil (i.e. when the user wanted a mirror without images), the xchunks were not were not processed.
- **Logrotate**: Log files /var/log/wp-mirror.log* were world readable. This is a problem because the log files contain database user accounts and passwords whenever the user runs WP-MIRROR with the --debug option.

A.1.4 Changes in WP-MIRROR 0.3 (2012-03-04)

A.1.4.1 Functionality Added or Changed

- **Document**: The README has been updated with dependency information. Additionally, there are now instructions for installing WP-MIRROR: from a DEB package consistent with Debian GNU/Linux 6.0 (squeeze); and from a traditional tarball.
- **FSM**: WP-MIRROR undertakes a great number of tasks. While some can be run concurrently, some must be done in a particular order. Previously, the sequencing was handled by hundreds of lines of 'spaghetti code'. This has been replaced by very few lines of code that perform table look-ups (against the *type-state-priority* table). Basically, it is a Finite State Machine (FSM). This greatly eases to task of coding and debugging.

A.1.4.2 Bugs Fixed

- **Copyright**: The debian/copyright file has been rewritten to be machine-readable to satisfy Debian's DEP5 requirement.
- **Lintian**: The configuration file /root/.clisprc has been removed to satisfy a lintian error message. The necessary configuration has been moved into the WP-MIRROR code.
- **Miscellaneous**: Several minor bugs were fixed.
- **Pbuilder**: The Makefile for building, installing, and deinstalling WP-MIRROR has been rewritten for compatibility with pbuilder (e.g. commands such as mv and cp were replaced with install).
- **Pbuilder**: The WP-MIRROR DEB package is built by pbuilder, which is now configured for Debian GNU/Linux 6.0 pbuilder is used to build Debian packages from source in a fakeroot 'clean-room'. A successful build with pbuilder is a prerequisite for acceptance into a Debian distribution. (squeeze).

A.1.5 Changes in WP-MIRROR 0.2 (2011-12-25)

A.1.5.1 Functionality Added or Changed

- **Mirror**: Previously, WP-MIRROR could only mirror a single wikipedia. Now it can mirror a set of wikipedias (e.g. meta, simple, and zh).
- **Wikix**: Previously, one of the important tasks of WP-MIRROR was performed by `wikix`. According to the Wikimedia Foundation:

  > `Wikix` is a `C` based program written by Jeffrey Vernon Merkey that will read any `XML` dump provided by the foundation, extract all image names from the `XML` dump which it may reference, then generate a series of `BASH` or Bourne Unix style scripts which can be invoked to download all images from Wikimedia Commons and Wikipedia.
  >
  > `http://meta.wikimedia.org/wiki/Wikix`, accessed 2012-12-19

  The code for `wikix` is posted on the above mentioned web page. It can also be downloaded as a tarball from `http://wikix.ngen-cast.de/wikix.tar.gz`.

  Now, WP-MIRROR offers a built-in alternative to `wikix`. This alternative:

  - **PRO**: generates smaller shell scripts that capture more images files, and throw fewer `HTTP` 400 and 404 errors than `wikix`; but
  - **CON**: takes longer to run than `wikix`, and fails to download some image files that `wikix` does.

Appendix B

Configuration File Parameter Reference

Table B.1: WP-MIRROR Configuration File Parameter Reference

Name	Default	Range	Intr	Rem
mediawiki-version	1.19		0.6	
mirror-alter-table-timeout-sec-max	15000		0.6	
mirror-commonswiki	commonswiki		0.6	
mirror-create-table-timeout-sec-max	100		0.6	
mirror-dchunk-load-timeout-sec-max	3000		0.6	
mirror-dchunk-page-count	10		0.6	
mirror-download-connection-time-max	10000		0.6	
mirror-image-download-p	t	nil, t	0.1	
mirror-innodb-fast-index-creation	t	nil, t	0.6	
mirror-innodb-table-key-block-size-list	'(("categorylinks" . 4) ("externallinks" . 4) ("image" . 4) ("imagelinks" . 4) ("langlinks" . 4) ("pagelinks" . 4) ("templatelinks" . 4) ("text" . 4)))		0.6	
mirror-language-code-list	'("simple")		0.6	
mirror-objectcache-delete-limit	1000		0.1	
mirror-objectcache-threshold	10000		0.1	
mirror-profiles-max	5		0.6	
mirror-project-list	("wikipedia" "wiktionary")		0.6	
mirror-schunk-page-count	10		0.6	
mirror-split-sql	schunk	dchunk, schunk	0.6	

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
mirror-virtual-host-name	wpmirror.site		0.6	
mirror-xchunk-page-count	1000		0.6	
monitor-gui-font-preference-list	'("-*-helvetica-medium-r-*-*-12-*-*-*-*-*-*-*" "-*-lucida-medium-r-*-*-12-*-*-*-*-*-*" "-*-lucida-medium-r-*-*-14-*-*-*-*-*-*")		0.1	
monitor-mode	:auto	:auto, :gui, :screen, :text	0.1	
monitor-poll-sec	10		0.1	
monitor-poll-sec-min	10		0.1	
monitor-poll-sec-max	1000		0.1	
system-cpu	1	1-3	0.1	
system-cpu-min	1		0.4	
system-cpu-max	3		0.4	
system-hdd-write-cache	1	0, 1	0.6	
system-hdd-write-cache-flush	nil	nil, t	0.6	
system-partition-free-images-min	(* 5 *gigabyte*))		0.4	
system-partition-free-images-start-min	(* 100 *gigabyte*))		0.4	
system-partition-free-images-warn	(* 50 *gigabyte*))		0.4	
system-partition-free-innodb-min	(* 5 *gigabyte*))		0.4	
system-physical-memory-min	(* 4 *gigabyte*)		0.1	
whereis-apache2ctl	/usr/sbin/apache2ctl		0.2	
whereis-bachrc	/etc/bash.bashrc		0.1	
whereis-bunzip2	/bin/bunzip2		0.1	
whereis-cat	/bin/cat		0.4	
whereis-cp	/bin/cp		0.1	
whereis-chown	/bin/chown		0.1	
whereis-chmod	/bin/chmod		0.1	
whereis-cron	/etc/cron.d/wp-mirror		0.1	
whereis-curl	/usr/bin/curl		0.1	
whereis-curlrc	/root/.curlrc		0.1	
whereis-directory-apache-sites-available	/etc/apache2/sites-available/		0.2	
whereis-directory-apache-sites-enabled	/etc/apache2/sites-enabled/		0.2	
whereis-directory-mediawiki	/var/lib/mediawiki/		0.5	
whereis-directory-mediawiki-config	/etc/mediawiki/		0.4	

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
whereis-directory-mediawiki-extensions-math	/var/lib/mediawiki/extensions/Math/math/		0.4	
whereis-directory-mediawiki-images	/var/lib/mediawiki/images/		0.6	
whereis-directory-mediawiki-maintenance	/usr/share/mediawiki/maintenance/		0.1	
whereis-directory-mysql-config	/etc/mysql/		0.4	
whereis-directory-mysql-config-conf.d	/etc/mysql/conf.d/		0.4	
whereis-directory-mysql-datadir	/var/lib/mysql/		0.4	
whereis-directory-tmp	/tmp/		0.2	
whereis-directory-wpmirror-config	/etc/wp-mirror/		0.4	
whereis-directory-wpmirror-restore	/usr/share/doc/wp-mirror/restore/		0.6	
whereis-directory-wpmirror-working	/var/lib/mediawiki/images/wp-mirror/		0.6	
whereis-echo	/bin/echo		0.6	
whereis-env	/usr/bin/env		0.4	
whereis-file-dev-null	/dev/null		0.2	
whereis-file-etc-hosts	/etc/hosts		0.2	
whereis-file-mediawiki-config-localsettings	LocalSettings.php		0.4	
whereis-file-mediawiki-config-localsettings-account	LocalSettings_account.php		0.4	
whereis-file-mediawiki-config-localsettings-wpmirror	LocalSettings_wpmirror.php		0.4	
whereis-file-mediawiki-farm-database	database_farm.sql		0.4	
whereis-file-mediawiki-farm-update	update_farm.php		0.2	
whereis-file-mediawiki-favicon	favicon.ico		0.5	
whereis-file-mediawiki-logo	wp-mirror.png		0.5	
whereis-file-mediawiki-update	update.php		0.2	
whereis-file-mysql-config-debian	debian.cnf		0.4	
whereis-file-mysql-config-wpmirror	wp-mirror.cnf		0.4	
whereis-file-mysql-log-file	ib_logfile*		0.4	
whereis-file-tmp-http	/tmp/http		0.4	

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
whereis-file-virtual-host	wpmirror.site.conf		0.2	
whereis-file-wpmirror-config-default	default.conf		0.4	
whereis-file-wpmirror-config-local	local.conf		0.4	
whereis-file-wpmirror-log	/var/log/wp-mirror.log		0.6	
whereis-file-wpmirror-pid	/var/run/wp-mirror.pid		0.6	
whereis-gm	/usr/bin/gm		0.1	
whereis-grep	/bin/grep		0.2	
whereis-gunzip	/bin/gunzip		0.4	
whereis-hdparm	/sbin/hdparm		0.1	
whereis-inkscape	/usr/bin/inkscape		0.4	
whereis-invoke-rc.d	/usr/sbin/invoke-rc.d		0.4	
whereis-ls	/bin/ls		0.1	
whereis-md5sum	/usr/bin/md5sum		0.1	
whereis-mv	/bin/mv		0.1	
whereis-mwxml2sql	/usr/bin/mwxml2sql		0.6	
whereis-mysql	/usr/bin/mysql		0.1	
whereis-mysqldump	/usr/bin/mysqldump		0.2	
whereis-mysql-tzinfo-to-sql	/usr/bin/mysql_tzinfo_to_sql		0.4	
whereis-openssl	/usr/bin/openssl		0.2	
whereis-php	/usr/bin/php		0.1	
whereis-replace	/usr/bin/replace		0.6	
whereis-rm	/bin/rm		0.1	
whereis-rsvg	/usr/bin/rsvg		0.1	
whereis-split	/usr/bin/split		0.6	
whereis-tar	/usr/bin/texvc		0.6	
whereis-texvc	/usr/bin/texvc		0.4	
whereis-wc	/usr/bin/wc		0.1	
whereis-wget	/usr/bin/wget		0.1	
whereis-wgetrc	/etc/wgetrc		0.1	
whereis-x	/usr/bin/X		0.1	
whereis-zgrep	/bin/zgrep		0.6	
whereis-zoneinfo	/usr/share/zoneinfo		0.4	
wikimedia-path-checksums-template	xxwiki/latest/xxwiki-latest-md5sums.txt		0.6	

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
wikimedia-path-dump-template-list	'("xxwiki/yyyymmdd/xxwiki-yyyymmdd-categorylinks.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-category.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-externallinks.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-imagelinks.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-image.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-interwiki.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-iwlinks.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-langlinks.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-oldimage.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-pagelinks.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-page_props.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-page_restrictions.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-protected_titles.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-redirect.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-templatelinks.sql.gz" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-pages-articles.xml.bz2" "xxwiki/yyyymmdd/xxwiki-yyyymmdd-stub-articles.xml.gz"))		0.6	
wikimedia-path-idump-template	"fulls/yyyymmdd/xxwiki-yyyymmdd-zz-media-tbl.tar"		0.6	
wikimedia-language-code-list			0.6	
wikimedia-large-wiki-list	'("en" "de" "nl" "fr" "it" "ru" "es" "sv" "pl" "ja")		0.1	
wikimedia-site-idump	http://ftpmirror.your.org/pub/wikimedia/imagedumps/tarballs/		0.6	
wikimedia-site-image	http://ftpmirror.your.org/pub/wikimedia/images/		0.6	
wikimedia-site-xdump	http://ftpmirror.your.org/pub/wikimedia/dumps/		0.6	
wikimedia-site-xdump-dirlist	"rsync-dirlist-last-1-good.txt"		0.6	
wpmirror-config-delete-list			0.4	
wpmirror-config-restore-list			0.4	

Table B.2: WP-MIRROR Configuration File Parameter Reference (Obsolete)

Name	Default	Range	Intr	Rem
ichunk-curl-retry	0	0, 1	0.1	0.6
mirrors	1	1-3	0.1	0.6
mirror-chunk-diff-threshold	20		0.1	0.2
mirror-ichunk-connection-time-max	3000		0.6	0.6
mirror-ichunk-persistent-connection	1000		0.6	0.6
mirror-ichunk-post-connection-sleep	1		0.6	0.6
mirror-image-rebuild-threshold	20000		0.1	0.2
mirror-image-then-page-p	t	nil, t	0.1	0.2

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
mirror-image-use-wikix-p	nil	nil, t	0.1	0.6
mirror-image-validate-p	t	nil, t	0.1	0.6
mirror-importdump-timeout-sec-max	nil	nil, 500-5000	0.4	0.6
mirror-languages	'("simple")		0.2	0.6
mirror-rebuildimages-threshold	20000		0.2	0.6
mirror-rebuildimages-timeout-sec-inc	200		0.2	0.6
mirror-rebuildimages-timeout-sec-min	2000		0.2	0.6
mirror-rebuildimages-timeout-sec-max	20000		0.2	0.6
monitor-gui-one-window-per-wiki	t	nil, t	0.6	0.6
monitor-gui-one-window-per-language	t	nil, t	0.1	0.6
system-disk-space-min	(* 250 *gigabyte*)		0.1	0.4
virtual-host-name	wpmirror.site		0.4	0.6
whereis-directory-configuration	/etc/wp-mirror/		0.1	0.4
whereis-directory-images	/var/lib/mediawiki/images/		0.1	0.6
whereis-directory-images-bad	/var/lib/mediawiki/images/bad-images/		0.1	0.6
whereis-directory-images-math	/var/lib/mediawiki/images/math/		0.4	0.6
whereis-directory-images-thumb	/var/lib/mediawiki/images/thumb/		0.4	0.6
whereis-directory-images-tmp	/var/lib/mediawiki/images/tmp/		0.4	0.6
whereis-directory-mediawiki	/etc/mediawiki/		0.2	0.4
whereis-directory-mediawiki-extensions	/etc/mediawiki-extensions/		0.4	0.4
whereis-directory-mediawiki-extensions-available	/etc/mediawiki-extensions/extensions-available/		0.4	0.4
whereis-directory-mediawiki-extensions-enabled	/etc/mediawiki-extensions/extensions-enabled/		0.4	0.4
whereis-directory-mediawiki-includes	/usr/share/mediawiki/includes/		0.2	0.4
whereis-directory-mysql	/var/lib/mysql/		0.1	0.4
whereis-directory-mysql	/etc/mysql/		0.4	0.4

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
whereis-directory-working	/var/lib/mediawiki/images/wp-mirror/		0.1	0.6
whereis-echo	/bin/echo/		0.2	0.4
whereis-file-checksums-template	xxwiki-latest-md5sums.txt		0.2	0.6
whereis-file-config-default	default.conf		0.1	0.4
whereis-file-config-local	local.conf		0.1	0.4
whereis-file-dump-template	xxwiki-yyyymmdd-pages-articles.xml.bz2		0.2	0.6
whereis-file-log	/var/log/wp-mirror.log		0.1	0.6
whereis-file-mediawiki-farm-importdump	importDump_farm.php		0.2	0.6
whereis-file-mediawiki-farm-rebuildimages	rebuildImages_farm.php		0.2	0.6
whereis-file-mediawiki-importdump	importDump.php		0.2	0.6
whereis-file-mediawiki-rebuildimages	rebuildImages.php		0.2	0.6
whereis-file-mediawiki-import	Import.php		0.2	0.4
whereis-file-pid	/var/run/wp-mirror.pid		0.4	0.6
whereis-file-wikidb-template	template.sql		0.2	0.4
whereis-file-mediawiki-adminsettings	AdminSettings.php		0.2	0.4
whereis-file-mediawiki-localsettings	LocalSettings.php		0.2	0.4
whereis-file-mediawiki-localsettings-wpmirror	LocalSettings_wpmirror.php		0.2	0.4
whereis-file-mysql-custom	/etc/mysql/conf.d/custom.cnf		0.4	0.4
whereis-file-mysql-debian	/etc/mysql/debian.cnf		0.4	0.4
whereis-mediawiki-adminsettings	/etc/mediawiki/AdminSettings.php		0.1	0.2
whereis-mediawiki-importdump	/usr/share/mediawiki/maintenance/importDump.php		0.1	0.2
whereis-mediawiki-localsettings	/etc/mediawiki/LocalSettings.php		0.1	0.2
whereis-mediawiki-rebuildimages	/usr/share/mediawiki/maintenance/rebuildImages.php		0.1	0.2
whereis-mediawiki-update	/usr/share/mediawiki/maintenance/update.php		0.1	0.2
whereis-mysql-custom	/etc/mysql/conf.d/custom.cnf		0.1	0.4
whereis-php5-suhosin	/etc/php5/conf.d/suhosin.ini		0.1	0.4
whereis-pidfile	/var/run/wp-mirror.pid		0.1	0.4
whereis-wikix	/usr/bin/wikix		0.1	0.6
wikimedia-checksums	xxwiki/latest/xxwiki-latest-md5sums.txt		0.1	0.2

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
wikimedia-checksums-template	xxwiki/latest/xxwiki-latest-md5sums.txt		0.2	0.6
wikimedia-dump-pattern	pages-articles.xml.bz2		0.1	0.2
wikimedia-dump-template	xxwiki/yyyymmdd/xxwiki-yyyymmdd-pages-articles.xml.bz2		0.2	0.6
wikimedia-import	http://svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/includes/Import.p		0.1	0.4
wikimedia-languages-to-mirror	'("simple")		0.1	0.2
wikimedia-site	http://dumps.wikimedia.org/		0.1	0.6
wikimedia-wikix	http://wikix.ngen-cast.de/wikix.tar.gz		0.1	0.6
wpmirror-directory-restore	/usr/share/doc/wp-mirror/restore/		0.4	0.6
xchunk-page-count	1000		0.1	0.6

Appendix C

Design Notes

C.1 Concurrency

WP-MIRROR imposes limits on concurrency:

- If `InnoDB` uses the `Antelope` storage format, then concurrency is limited to the lesser of the number of CPUs and three; and
- If `InnoDB` uses the `Barracuda` storage format, then concurrency is limited one.

C.1.1 Why not Fork 20 `ichunk` Sub-processes? (Obsolete)

Obsolete: As of WP-MIRROR 0.6, images are downloaded using `wget`, which has an automatic retry feature. So corrupt images files are now rare. Concurrency is limited on the server side. Limits to the number of connections from the same IP address, and rate limits should be expected.

Original text: Images are downloaded using `cURL`.

Experiments showed that forking three or more sub-processes results in `cURL` failing to download an unacceptable number of image files. Of course, many of the missing images can be downloaded later by rerunning each `ichunk`, but this hurts performance. There is, however, a worse problem, namely, `cURL` frequently times-out after partially downloading a file. Hence all image files must later be validated (e.g. with `gm identify -verbose`), and the corrupt ones sequestered to a `bad-images` directory for later inspection. Processing `ichunk`s one or two at a time works acceptably well.

C.1.2 Concurrency and `fsm-file-scrape` (Obsolete)

Obsolete: As of WP-MIRROR 0.6, `fsm-file-scrape` is no longer used. Rather image file names are taken from `xxwiki.imagelinks.il_to` and `commonswiki.image.img_name`.

Original text: Experiments showed that:

1. three processes running `fsm-file-scrape` concurrently, can generate enough heat to shut down the laptop, and
2. two processes running `fsm-file-scrape` concurrently, seems reasonable.

Scraping `xchunk`s for image file names is CPU intensive.

Note: In recent years, most laptops are equiped with a number of temperature sensors. These can be used for self-protection of the laptop. If one of the temperatures exceeds a certain threshold, the laptop will automatically turn off. Some thresholds can be set by the user. The author sets a threshold of 55^o for hard disk drives because: 1) they have a design limit of 60^o, and 2) hot disks do not last.

C.1.3 Concurrency and `xchunk` Sub-processes (Obsolete)

Obsolete: As of WP-MIRROR 0.6, `importDump.php` is no longer used. Nor are thumbs make during the importation process.

Original text: Experiments showed that:

1. During building the mirror for the first time, forking even two sub-processes results in significantly degraded InnoDB performance. The main reason seems to be that PHP (running importDump.php), and gm (making thumbs) burden the CPUs heavily.
2. During mirror updates, forking two sub-processes can improve performance. The main reasons being that there is much less gm activity (most thumbs already made), which leaves a 'locality of reference' issue, mostly involving the pagelinks table, which burdens the disks heavily. Competition between multiple sub-processes for a finite number of InnoDB buffer pool pages in DRAM results in more 'cache misses' and more disk accesses. If there is ample DRAM, then the number of CPUs becomes the limiting factor. There seems to be no advantage to forking more xchunks than one has CPUs. Indeed one or two seems reasonable.
3. If InnoDB uses the Barracuda file format (data compressed using zlib) then xchunks should be processed one-by-one. Barracuda does not handle concurrency well. Even two concurrent xchunks will cause a great number of deadlocks. The author recommends: for laptops, use Barracuda to save space; but for desktops, use Antelope on a raw partition to maximize speed.

C.1.4 Concurrent Processing of ichunks in Parallel with xchunks (Obsolete)

Experiments showed that:

- When the xchunk corresponds to the ichunk (that is, the chunks pertain to the same articles), then this results in InnoDB resource deadlocks, failure to create the records involved in the deadlock, and degraded InnoDB performance while waiting for deadlocks to time-out.
- When care is exercised to ensure that corresponding chunks are not processed concurrently, then significant performance gains are achieved.

Update 2013: As of WP-MIRROR 0.6, the databases are loaded before the images are downloaded.

C.1.5 Design Decision

WP-MIRROR was designed so that one could have multiple mirror and monitor processes communicating state via the wpmirror database. For each working file (checksum, xdump, idump, xml, xchunk, ichunk) there is a record in the database showing its name, size, type, and state—and a semaphore.

InnoDB is an ACID compliant transactional storage engine, and therefore already has the locking mechanism needed to keep transactions Isolated (the 'I' in ACID). The locking can be used to maintain semaphores (one for each file) that can keep concurrent mirror processes from handling the same file at the same time.

For desktop, experiments using four processes: three mirror (two downloading images, one importing articles) and one monitor, have worked well by providing a balanced burden upon CPUs, disks, and Internet.

For laptop, experiments using two processes: one mirror and one monitor, have worked well.

C.2 Durability (the 'D' in ACID)

C.2.1 HDD Write Caching

Source: http://monolight.cc/2011/06/barriers-caches-filesystems/

WP-MIRROR 0.5 and earlier versions, automatically disabled the HDD write cache for any disks underlying InnoDB. This was to assure the Durability of each database transaction COMMITted.

If HDD write caching is enabled, then any writes to disk actually go first to an electronic cache, and only later are written to disk. Caching has a few consequences:

- **Write Performance**: Writing to the physical disk, which involves the mechanical movement of read-write heads and disk rotation, can be ordered using something like the 'elevator algorithm'.
- **Read Performance**: Reading from the physical disk has a latency (5ms) that is orders of magnitude greater than that of reading from electronic cache (90μs).
- **Durability**: Unfortunately, in the event of a system failure (e.g. power fails) items in electronic cache are lost. This can result in database corruption (lost transactions that were supposedly COMMITted) as well as lost files.

It is possible to strike a compromise between performance and durability. For example, all modern file systems use journalling to guard against file system corruption.

WP-MIRROR 0.6 by default disables the HDD write cache. The user may configure HDD write caching using two parameters: *system-hdd-write-cache* and *system-hdd-write-cache-flush*. For example, consider:

```
root-shell# cat /etc/wp-mirror/local.conf | grep hdd
(defparameter *system-hdd-write-cache*         1)
(defparameter *system-hdd-write-cache-flush*   t)
```

This setting has two effects: The first parameter enables HDD write caching, while the second flushes the HDD write cache immediately after each checkpoint. This assures the Durability of all transactions prior to the last checkpoint. Transactions after the last checkpoint may be lost, but this is not tragic because WP-MIRROR resumes from the last checkpoint and therefore repeats those transactions.

C.2.2 Experiments

Time trials is needed to quantify the performance gains (if any).

C.2.2.1 Experimental Method

We set up a desktop PC with Debian GNU/Linux 7.0 and WP-MIRROR 0.6 installed.

Platform	Storage
Desktop	InnoDB Barracuda format, on ReiserFS over LVM2 over LUKS over RAID1 (mirror) over whole disks Seagate, 1.5T, Model: ST31500341AS, Transport: Serial

Disk information can be obtained by executing:

```
root-shell# hdparm -I /dev/sde
/dev/sde:
ATA device, with non-removable media
        Model Number:       ST31500341AS
        Serial Number:      9VS2HM9Q
        Firmware Revision:  CC1H
        Transport:          Serial
...
```

First, build a mirror by executing:

```
root-shell# emacs /etc/wp-mirror/local.conf
(defparameter *mirror-language-code-list*        '(xh zu))
root-shell# wp-mirror --mirror
```

Building the mirror involves downloading all the images, validating them, and producing thumbs. Validation and resizing burden the disks with far more reads than writes. Therefore we shall complete this image processing before the time trials, in order to better isolate the effect of HDD write caching upon database performance.

Second, for each run: 1) drop the database (keeping all image files intact), 2) edit the configuration file, 3) SET MySQL variables, and 4) time the mirror building.

C.2.2.2 Experiment 1a: Baseline

Process: When a transaction COMMITs:

- transaction is written to log buffer,
- log buffer is written immediately to log file,
- log file is written immediately to physical disk.

Durability: Greatest possible durability.
Configuration:

```
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# emacs /etc/wp-mirror/local.conf
(defparameter *mirror-language-code-list* '(xh zu))
(defparameter *system-hdd-write-cache*           0)          <-- 0 or 1
(defparameter *system-hdd-write-cache-flush*   nil)          <-- nil or t
root-shell# mysql --host=localhost --user=root --password
mysql> SET GLOBAL innodb_flush_log_at_trx_commit=1;          -- 0, 1, or 2
mysql> SHOW VARIABLES LIKE 'innodb_flush_log_at_trx_commit';
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| innodb_flush_log_at_trx_commit | 1     |
+--------------------------------+-------+
1 row in set (0.00 sec)
mysql> quit;
root-shell# time wp-mirror --mirror
real    51m21.812s
user    20m9.332s
sys     2m23.192s
```

Of the three times (real, user, and sys) only the first is interesting to the user. It is sometimes referred to as 'wall clock' time. The other two (user and sys) are not accurate because: they refer only to resources consumed by WP-MIRROR, and not to resources consumed by the many sub-processes that WP-MIRROR forks to perform most of the work.

Experiments 1a and 1b were repeated to estimate how much 'scatter' there is in the times. The elapsed 'real' times vary by about 1m, probably due to other loads on the system. This means that the performance *loss* incurred by enabling HDD write caching is statistically significant.

C.2.2.3 Experiment 1b: HDD Write Cache Enabled

Process: When a transaction COMMITs:

- transaction is written to log buffer,
- log buffer is written immediately to log file,
- log file is flushed immediately to HDD write cache,
- HDD write cache is written *later* to physical disk.

Durability: System failure can cause loss of any transaction held in HDD write cache that is not yet written to the physical disk.

Configuration:

```
(defparameter *system-hdd-write-cache*         1)
(defparameter *system-hdd-write-cache-flush*   t)
mysql> SET GLOBAL innodb_flush_log_at_trx_commit=1;
```

C.2.2.4 Experiment 1c: `Log file` Flushed Once Per Second

Process: When a transaction `COMMIT`s:

- transaction is written to `log buffer`,
- `log buffer` is written immediately to `log file`,
- `log file` is written *once per second* to HDD write cache,
- HDD write cache is *later* written to physical disk.

Durability: System failure can cause loss of up to one second of transactions.

Configuration:

```
(defparameter *system-hdd-write-cache*         1)
(defparameter *system-hdd-write-cache-flush*   t)
mysql> SET GLOBAL innodb_flush_log_at_trx_commit=2;
```

C.2.2.5 Experiment 1d: `Log Buffer` Flushed Once Per Second

Process: When a transaction `COMMIT`s:

- transaction is written to `log buffer`,
- `log buffer` is written *once per second* to `log file`,
- `log file` is flushed immediately to HDD write cache,
- HDD write cache is written *later* to physical disk.

Durability: Any `mysqld` crash, and any system failure, can cause loss of up to one second of transactions.

Configuration:

```
(defparameter *system-hdd-write-cache*         1)
(defparameter *system-hdd-write-cache-flush*   t)
mysql> SET GLOBAL innodb_flush_log_at_trx_commit=0;
```

C.2.2.6 Experiment 1e: HDD Write Cache Disabled, `Log File` Flushed Once Per Second

Process: When a transaction `COMMIT`s:

- transaction is written to `log buffer`,
- `log buffer` is written immediately to `log file`,
- `log file` is written *once per second* to physical disk.

Durability: Any `mysqld` crash, and any system failure, can cause loss of up to one second of transactions.

Configuration:

```
(defparameter *system-hdd-write-cache*         0)
(defparameter *system-hdd-write-cache-flush* nil)
mysql> SET GLOBAL innodb_flush_log_at_trx_commit=2;
```

C.2.2.7 Experimental Results

The time trials, rounded to nearest minute, yield the following:

	Experiments				
	1a	**1b**	**1c**	**1d**	**1e**
Platform	desktop				
mirror-language-code-list	(xh zu)				
system-hdd-write-cache	0	1	1	1	0
system-hdd-write-cache-flush	nil	t	t	t	nil
innodb_flush_log_at_trx_commit	1	1	2	0	2
real:	51m	57m	44m	44m	43m
user:	20m	20m	21m	21m	20m
sys:	2m	3m	2m	3m	2m
real time as percentage of baseline	100%	111%	86%	86%	84%

C.2.2.8 Conclusions

For the storage configuration used in these experiments:

- **HDD Write Caching**: A performance *loss* (11% more time overall) was incurred by *enabling* HDD write caching,
- **Log Flushing**: Modest performance gain (16% less time overall) was achieved by writing log files to disk *once per second*.

Update: WP-MIRROR 0.6 can perform profiling. Profiling data is displayed using the command-line option --profile. These experiments were repeated with profiling. See §G.3, fsm-file-import and Durability.

C.3 Forking Subprocesses

WP-MIRROR gets most of its work done by forking sub-processes. This is done for robustness. If a sub-process fails, the damage is contained within that sub-process, and WP-MIRROR continues to run.

When we print out the kernel ring buffer, we discover that image processing is error prone:

```
root-shell# dmesg | grep segfault
[311284.542732] gm[18286]: segfault at 30000001f ip 00007ffc3a7d63eb sp 00007fffba65c5d0
 error 4 in libGraphicsMagick.so.3.9.0[7ffc3a6bd000+29f000]
[312876.143152] gm[21701]: segfault at 30000001f ip 00007f5029d853eb sp 00007fff4678cef0
 error 4 in libGraphicsMagick.so.3.9.0[7f5029c6c000+29f000]
[631890.556242] convert[28284]: segfault at 7f5a2834e000 ip 00007f5a26c57f31 sp 00007fff1434b750
 error 6 in libGraphicsMagick.so.3.9.0[7f5a26b46000+29f000]
[638061.613181] convert[11446]: segfault at 7f8461790000 ip 00007f8460843f31 sp 00007fff84543de0
 error 6 in libGraphicsMagick.so.3.9.0[7f8460732000+29f000]
[678738.583009] convert[24596]: segfault at 7f450210e020 ip 00007f4500175f31 sp 00007fffd0725f90
 error 6 in libGraphicsMagick.so.3.9.0[7f4500064000+29f000]
```

These 'segfaults' however are contained within their respective sub-processes, and do not disturb WP-MIRROR.

C.4 Icons

C.4.1 Graphic Arts

Icons fall into two categories: raster and vector.

C.4.1.1 Raster Images

What: Raster images, such as the PNG format, are created with a fixed resolution (e.g. 32x32 pixels).

Pros: Good for working with photographic images.

Cons: When magnified, lines and edges can look ragged.

Tools: Open source tools for raster images include the gimp.

C.4.1.2 Vector Images

What: Vector images, such as the SVG format, are defined as a set of geometrical objects (lines, polygons) with edges and interiors that may be colored or textured.

Pros: Good for simple geometric patterns, such as national flags, coats of arms, company logos, and the like. When magnified they should preserve their appearance.

Tools: Open source tools for vector images include inkscape.

C.4.2 Logotype for use by MediaWiki

Documentation: http://www.mediawiki.org/wiki/Manual:$wgLogo

MediaWiki offers a number of "skins". A skin provides a consistent look and feel to all of its web pages, and consists of: a cascading style sheet (css file), some image files, and a few bits of code (js or php files).

As of MediaWiki 1.19, the default skin is vector. Several other skins are also available (chick, cologneblue, modern, monobook, nostalgia, and simple). These are all found under /var/lib/mediawiki/skins. The "legacy" skin is monobook.

These skins should display a logo. The default is /var/lib/mediawiki/skins/common/images/wiki.png (see margin). This default should be replaced with one of our own. However, according to http://www.mediawiki.org/wiki/Manual:FAQ:

> The maximum logo size in Vector is 160x160px, while MonoBook's is 155x155px, and Standard / Classic's is 135x135px. A good logo size is 135x135px or one of 150x150px or 155x155px if you don't care about legacy skins. Note that in Vector and MonoBook a logo that is too large will be cut off, while in Standard / Classic a logo that is too large will simply extend a tiny bit into the content area.
>
> FAQ, "How do I change the logo?", accessed 2012-12-19

To assure backward compatibility with legacy skins, the WP-MIRROR Logotype is 135x135 pixels. It was made using the gimp. The artistic process is not discussed here, nor are the details of how to use the gimp.

We install the logotype as /var/lib/mediawiki/wp-mirror.png, set ownership to www-data:www-data (so apache2 can use it), and configure MediaWiki to find it.

```
shell$ ls /var/lib/mediawiki/[fw]*[og]
-rw-r--r-- 1 www-data www-data   971 Dec 14 05:41 favicon.ico
-rw-r--r-- 1 www-data www-data 30677 Dec 14 05:41 wp-mirror.png
shell$ cat /etc/mediawiki/LocalSettings_wpmirror.php
...
$wgFavicon              = "$wgScriptPath/favicon.ico";
$wgLogo                 = "$wgScriptPath/wp-mirror.png";
```

C.4.3 Favicon for use by the Web Browser

Documentation: http://www.mediawiki.org/wiki/Manual:$wgFavicon.

A favicon is a 16x16 (or 32x32) pixel icon in ICO format. It is displayed in the web browser's Address Bar (the text box where you enter the URL).

We can make a favicon by miniaturizing our logotype. Interestingly, Graphics Magick is not able to write an ICO file. Therefore, we must manually use the gimp. First, execute:

```
shell$ gimp 16px-wp-mirror.png
File -> Export ...
```

Second, a new window should open with title “Export Image”. Click on

```
Select File Type (By Extension)
```

scroll down to

```
File Type                                       Extensions
Microsoft Windows Icon                          ico
```

click on “Microsoft Window Icon”, and click “Export”.

Third, a new window should open with title “Export Image as Windows Icon”. Click “Export”, and exit the gimp.

Fourth, repeat for 32px-wp-mirror.png.

Finally, execute:

```
shell$ cp 16px-wp-mirror.ico favicon.ico
shell$ ls -l *ico
-rw-r--r-- 1 kmiller kmiller 1150 Dec 15 02:22 16px-wp-mirror.ico
-rw-r--r-- 1 kmiller kmiller 4286 Nov 29 06:43 32px-wp-mirror.ico
-rw-r--r-- 1 kmiller kmiller 1150 Dec 15 02:22 favicon.ico
```

We install the favicon as /var/lib/mediawiki/favicon.ico, set ownership to www-data:www-data (so apache2 can use it), and configure MediaWiki to find it.

```
shell$ ls /var/lib/mediawiki/[fw]*[og]
-rw-r--r-- 1 www-data www-data   971 Dec 14 05:41 favicon.ico
-rw-r--r-- 1 www-data www-data 30677 Dec 14 05:41 wp-mirror.png
shell$ cat /etc/mediawiki/LocalSettings_wpmirror.php
...
$wgFavicon              = "$wgScriptPath/favicon.ico";
$wgLogo                 = "$wgScriptPath/wp-mirror.png";
```

C.4.4 Makefile

All the icons, except for the favicon, can be converted by Graphics Magick from the original PNG file.

```
shell$ cat Makefile
PROGRAM   = wp-mirror
XCF       = $(PROGRAM).xcf
PNG       = $(PROGRAM).png
ICON16    = 16px-$(PROGRAM).png
ICON22    = 22px-$(PROGRAM).png
ICON24    = 24px-$(PROGRAM).png
ICON32    = 32px-$(PROGRAM).png
ICON48    = 48px-$(PROGRAM).png
ICON64    = 64px-$(PROGRAM).png
ICON128   = 128px-$(PROGRAM).png
ICON256   = 256px-$(PROGRAM).png
ICON135   = 135px-$(PROGRAM).png
PIXMAP    = $(PROGRAM).xpm
ICO       = favicon.ico

thumb:
        gm convert -size 16x16   $(PNG) -resize 16x16   +profile "*" $(ICON16)
        gm convert -size 22x22   $(PNG) -resize 22x22   +profile "*" $(ICON22)
        gm convert -size 24x24   $(PNG) -resize 24x24   +profile "*" $(ICON24)
        gm convert -size 32x32   $(PNG) -resize 32x32   +profile "*" $(ICON32)
        gm convert -size 48x48   $(PNG) -resize 48x48   +profile "*" $(ICON48)
        gm convert -size 64x64   $(PNG) -resize 64x64   +profile "*" $(ICON64)
        gm convert -size 128x128 $(PNG) -resize 128x128 +profile "*" $(ICON128)
        gm convert -size 256x256 $(PNG) -resize 256x256 +profile "*" $(ICON256)
        gm convert -size 135x135 $(PNG) -resize 135x135 +profile "*" $(ICON135)
        gm convert -size 32x32   $(PNG) -resize 32x32   +profile "*" $(PIXMAP)
        # the ICO file must be made with GIMP (File->Export...)

clean:
      rm -f *~
```

C.4.5 Pixmap for use by the Debian Menu System

We would like to add WP-MIRROR to the Debian Menu System. Tools and documentation are available.

```
root-shell# aptitude install dh-make maint-guide menu
shell$ man menufile
shell$ lynx http://localhost/doc/maint-guide/html/index.en.html
```

According to the Debian New Maintainer's Guide

> X Window System users usually have a window manager with a menu that can be customized to launch programs. If they have installed the Debian menu package, a set of menus for every program on the system will be created for them.
>
> Here's the default menu.ex file that dh_make created.
>
> ```
> ?package(gentoo):needs=X11|text|vc|wm \
>        section=Applications/see-menu-manual \
>        title=gentoo command=/usr/bin/gentoo
> ```
>
> The first field after the colon character is needs, and it specifies what kind of interface the program needs. Change this to one of the listed alternatives, e.g. text or X11.
>
> The next is the section that the menu and submenu entry should appear in. [58]
>
> The title field is the name of the program. You can start this one in uppercase if you like. Just keep it short.

Finally, the command field is the command that runs the program.
Let's change the file name to menu and change the menu entry to this:

```
?package(gentoo): needs=X11 \
        section=Applications/Tools \
        title=Gentoo command=gentoo
```

You can also add other fields like longtitle, icon, hints etc. See dh_installmenu(1), menufile(5), update-menus(1), and The Debian Menu sub-policy for more information. Debian New Maintainer's Guide, Chapter 5.16

In our case, we filled out the menu file like so:

```
shell$ cat debian/menu
?package(wp-mirror):\
        needs="X11" \
        section="Applications/Network/Web Browsing" \
        command="/usr/bin/wp-mirror --gui" \
        title="wp-mirror" \
        icon="/usr/share/pixmaps/wp-mirror.xpm" \
        longtitle="Wp-mirror: Utility for mirroring a set of wikipedias"
```

When we execute:

```
root-shell# dpkg --install wp-mirror_0.6-1_all.deb
```

This file is then automatically installed as /usr/share/menu/wp-mirror.

The icon, which is optional, should be 32x32 pixels, in XPM format, and installed as /usr/share/pixmaps/wp-mirror.xpm. This pixmap is easily converted from the original PNG image file using Graphics Magick. See the Makefile shown in §C.4.4, Makefile, the relevant lines of which are:

```
PROGRAM    = wp-mirror
PNG        = $(PROGRAM).png
PIXMAP     = $(PROGRAM).xpm

thumb:
        gm convert -size 32x32   $(PNG) -resize 32x32   +profile "*" $(PIXMAP)
```

Experience: This works in the sense that WP-MIRROR and the icon are added to the menu system. However, when clicked, we see a small icon bounce up and down for while, and then vanish. This seems to be a permissions issue. Presently, WP-MIRROR must be run with root permissions.

C.4.6 Icons for use by the Window Manager

C.4.6.1 Icon Theme Specification

Documentation:
http://standards.freedesktop.org/icon-theme-spec/icon-theme-spec-latest.html

Window Manager: An X server (almost) always comes with a couple of specialized X clients: a window manager and a clip board. A window manager determines where and how the windows of other X clients are displayed on the screen. There are dozens of window managers available (e.g. kwin is the window manager for KDE). See http://en.wikipedia.org/wiki/Comparison_of_X_window_managers.

Window Icons: A window manager uses many icons of different resolution. For example:

- a window can be minimized and represented by an icon with some accompanying text;

- a window is usually surrounded by a frame at the top of which is a "Title Bar", which usually has a "Title Bar Icon" in the upper left corner;
- when the user presses alt-tab to cycle among the windows, an icon for each window will appear in turn;
- most window managers offer some kind of application launcher, which also make use of icons.

Themes: Window managers have "themes" (which are much like "skins" discussed above). Most of the theme information (icons, indexes, and caches) is stored under /usr/share/icons/:

```
shell$ ls /usr/share/icons/
default         oxy-chrome            oxy-norway          oxy-sea_blue
default.kde4    oxy-desert            oxy-obsidian        oxy-steel
gnome           oxy-emerald           oxy-obsidian-hc     oxy-terra
hicolor         oxygen                oxy-olympus         oxy-terra_green
locolor         oxy-green             oxy-olympus-inv     oxy-violet
mono            oxy-grey              oxy-orchid          oxy-viorange
oxy-black       oxy-honeycomb         oxy-oxygen          oxy-white
oxy-blue        oxy-hot_orange        oxy-peach           oxy-whitewater
oxy-bluecurve   oxy-lilac             oxy-purple          oxy-wonton
oxy-brown       oxy-midnight_meadow   oxy-red             oxy-yellow
oxy-cherry      oxy-navy              oxy-red-argentina   oxy-zion
```

One theme can "inherit" icons from another theme (the default, or perhaps fallback, theme is hicolor):

```
shell$ cat /usr/share/icons/oxygen/index.theme | grep Inherit
Inherits=hicolor
```

For efficiency, this information is cached (one cache per theme):

```
shell$ ls -l /usr/share/icons/oxygen/
drwxr-xr-x 11 root root       280 Sep  2 22:32 128x128
drwxr-xr-x 13 root root       336 Sep  2 22:32 16x16
drwxr-xr-x 13 root root       336 Sep  2 22:32 22x22
drwxr-xr-x  9 root root       232 Sep  2 22:32 256x256
drwxr-xr-x 12 root root       312 Sep  2 22:32 32x32
drwxr-xr-x 12 root root       312 Sep  2 22:32 48x48
drwxr-xr-x 11 root root       280 Sep  2 22:32 64x64
drwxr-xr-x  3 root root        72 Sep  2 22:32 8x8
-rw-r--r--  1 root root 146850472 Dec 14 09:06 icon-theme.cache
-rw-r--r--  1 root root      8827 Jul 27  2011 index.theme
drwxr-xr-x  6 root root       152 Sep  2 22:40 scalable
```

The Freedesktop.org publishes a number of standards for the X Window System. According to their Icon Theme Specification:

> Installing Application Icons
>
> So, you're an application author, and want to install application icons so that they work in the KDE and Gnome menus. Minimally you should install a 48x48 icon in the hicolor theme. This means installing a PNG file in $prefix/share/icons/hicolor/48x48/apps. Optionally you can install icons in different sizes. For example, installing a svg icon in $prefix/share/icons/hicolor/scalable/apps means most desktops will have one icon that works for all sizes. You might even want to install icons with a look that

matches other well known themes so your application will fit in with some specific desktop environment.

It is recommended that the icons installed in the hicolor theme look neutral, since it is a fallback theme that will be used in combination with some very different looking themes. But if you don't have any neutral icon, please install whatever icon you have in the hicolor theme so that all applications get at least some icon in all themes. Icon Theme Specification, v0.11, accessed 2012-12-19

C.4.6.2 WP-MIRROR Themes

WP-MIRROR provides a file to help the window manager locate an icon for a minimized window.

```
shell$ cat /usr/share/applications/wp-mirror.desktop
[Desktop Entry]
Categories=Network;
Comment=Mirror a set of wikipedias
Exec=/usr/bin/wp-mirror --gui
Icon=wp-mirror
Name=wp-mirror
Terminal=true
TryExec=/usr/bin/wp-mirror
Type=Application
```

C.4.6.3 WP-MIRROR Icons

WP-MIRROR provides icons for use by the window manager. They are stored as /usr/share/icons/hicolor/*resolution*/apps/wp-mirror.png, where *resolution* is 16x16, 22x22, 24x24, 32x32, 48x48, 64x64, 128x128, and 256x256.

C.4.6.4 Experience

None of the WP-MIRROR icons are displayed by the window manager (in our case kwin). Instead the default "X" icon is displayed. Rebuilding the KDE System Configuration Cache did not help:

```
shell$ man kbuildsycoca4
shell$ kbuildsycoca4 --noincremental
root-shell# kbuildsycoca4 --noincremental
root-shell# ls -l /var/tmp/kdecache-[UserName]/
...
root-shell# ls -l /var/tmp/kdecache-root/
...
```

Nor did clearing the icon caches:

```
shell$ cd ~/.kde/cache-[ComputerName]/
shell$ rm icon-cache.kcache
shell$ rm plasma_theme_*.kcache
```

Nor did restarting the X server. Still no WP-MIRROR icons displayed.

C.4.6.5 The Extended Window Manager Hints Standard

Documentation:
http://standards.freedesktop.org/wm-spec/1.3/index.html

Freedesktop.org published a standard for Extended Window Manager Hints. Each X client sends messages to the X server, which passes in turn passes them to the window manager. The

window manager takes these message under advisement. The standard describes over 30 of these messages (most of which are called properties).

Some of these properties are in regards to icons. In particular, we are interested in NET_WM_ICON, which tells the window manager which icons to use for the Title Bar Icon and the alt-tab icon.

C.4.6.6 The _NET_WM_ICON Hint

According to the Extended Window Manager Hints standard:

> _NET_WM_ICON
>
> _NET_WM_ICON CARDINAL[][2+n]/32
>
> This is an array of possible icons for the client. This specification does not stipulate what size these icons should be, but individual desktop environments or toolkits may do so. The Window Manager MAY scale any of these icons to an appropriate size.
>
> This is an array of 32bit packed CARDINAL ARGB with high byte being A, low byte being B. The first two cardinals are width, height. Data is in rows, left to right and top to bottom. Extended Window Manager Hints, v1.3, Application Window Properties, accessed 2012-12-19

So let us test this.

C.4.6.7 C++ Program to Test the _NET_WM_ICON Hint

The following example was found at http://lists.kde.org/?l=kwin&m=115435217715650&w=2.

The array buffer contains two icons: the first is 16x16 pixels, the second is 32x32 pixels. Each pixel is in ARGB format. Here the A stands for Alpha (as in alpha-blending, where 0 is transparent, and 255 is opaque), R for Red, G for Green, and B for Blue.

This array, buffer, is assigned to the property NET_WM_ICON, and the message sent to the X server. The window manager should take the hint, and use the icons for the Title Bar Icon and the alt-tab icon.

If the reader wishes to download the example posted above, then please be aware that it contains an half dozen typos. Some of the larger numbers in the buffer array contain an unwanted blank space. These blank spaces are caught by the compiler, and must then be manually removed. The code provided below is the corrected version.

```
shell$ cat native.cpp
// gcc -lX11 -lstdc++ -L/usr/X11/lib -o native native.cpp

#include <stdlib.h>
#include <X11/Xlib.h>

int main( int argc, char **argv )
{

    unsigned int buffer[] = {16, 16, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 338034905, 3657433343, 0, 184483840, \
234881279, 3053453567, 3221225727, 1879048447, 0, 0, 0, 0, 0, 0, 0, 1224737023, \
3305111807, 3875537151,0, 0, 2063597823, 1291845887, 0, 67109119, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
50266112, 3422552319, 0, 0, 3070230783, 2063597823, 2986344703, 771752191, 0, 0, 0, \
0, 0, 0, 0, 0, 0, 3422552319, 0, 0, 3372220671, 1509949695, 704643327, 3355443455, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 0, 3422552319, 0, 134152192, 3187671295, 251658495, 0, 3439329535, 0, 0, \
0, 0, 0, 0, 0, 0, 0, 3422552319, 0, 0, 2332033279, 1342177535, 167772415, 3338666239, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 0, 3422552319, 0, 0, 436207871, 3322085628, 3456106751, 1375731967, \
4278255360, 4026597120, 3758161664, 3489726208, 3204513536, 2952855296, 2684419840, \
2399207168, 2130771712, 1845559040, 1593900800, 1308688128, 1040252672, 755040000, \
486604544, 234946304, 4278255360, 4043374336, 3774938880, 3506503424, 3221290752, \
2952855296, 2667642624, 2399207168, 2130771712, 1862336256, 1627453957, 1359017481, \
1073805064, 788591627, 503379721, 218169088, 4278255360, 4043374336, 3758161664, \
3506503424, 3221290752, 2952855296, 2684419840, 2415984384, 2130771712, 1862336256, \
1577123584, 1308688128, 1040252672, 755040000, 486604544, 218169088, 4278190335, \
4026532095, 3758096639, 3489661183, 3221225727, 2952790271, 2667577599, 2415919359, \
2130706687, 1862271231, 1593835775, 1325400319, 1056964863, 771752191, 520093951, \
234881279, 4278190335, 4026532095, 3758096639, 3489661183, 3221225727, 2952790271, \
2667577599, 2415919359, 2130706687, 1862271231, 1593835775, 1325400319, 1056964863, \
771752191, 503316735, 234881279, 4278190335, 4026532095, 3758096639, 3489661183, \
3221225727, 2952790271, 2684354815, 2399142143, 2130706687, 1862271231, 1593835775, \
1325400319, 1040187647, 771752191, 520093951, 234881279, 4294901760, 4043243520, \
3774808064, 3506372608, 3221159936, 2952724480, 2684289024, 2399076352, 2147418112, \
1862205440, 1593769984, 1308557312, 1040121856, 771686400, 503250944, 234815488, \
4294901760, 4060020736, 3758030848, 3506372608, 3221159936, 2952724480, 2684289024, \
2415853568, 2130640896, 1862205440, 1593769984, 1308557312, 1040121856, 771686400, \
503250944, 234815488, 4294901760, 4043243520, 3774808064, 3489595392, 3237937152, \
2952724480, 2684289024, 2415853568, 2147418112, 1862205440, 1593769984, 1325334528, \
1056899072, 788463616, 503250944, 234815488, 32, 32, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 268369920, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 1509949695, 3120562431, 4009754879, 4194304255, 3690987775, 2130706687, \
```

```
83886335, 0, 50331903, 1694499071, 3170894079, 3992977663, 4211081471, 3657433343, \
1879048447, 16777471, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3087007999, \
2281701631, 1191182591, 1040187647, 2030043391, 4127195391, 2566914303, 0, 16777471, \
3254780159, 2181038335, 1191182591, 973078783, 2030043391,4177527039, 2130706687, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 0, 0, 0, 0, 0, 2214592767, 4093640959, 0, 0, 0, 0, 0, 0, 0, \
2298478847, 3909091583, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
0, 2214592767, 3607101695, 0, 0, 0, 0, 0, 0, 0, 1946157311, 4093640959, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 0, 0, 536871167, 1191182591, 2281701631,3019899135, 637534463, 0, 0, 0, \
100597760, 251592704, 33488896, 0, 3321889023, 2919235839, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
0, 0, 0, 0, 0, 0, 0, 0, 0, 2550137087, 4278190335, 4278190335, 3405775103, 570425599, \
0, 0, 0, 0, 0, 0, 2046820607, 4043309311, 620757247, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
33488896, 0, 0, 218104063, 1291845887, 3841982719, 3388997887, 0, 0, 0, 0, 0, \
1996488959, 4093640959, 1073742079, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
0, 0, 0, 0, 0, 0, 1761607935, 4278190335, 150995199, 0, 0, 67109119, 2550137087, \
3909091583, 889192703, 0, 0, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 0, 0, 0, 0, 0, \
2181038335, 3925868799, 0, 0, 218104063, 3070230783, 3623878911, 570425599, 0, 0, 0, \
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 805306623, 3288334591, 1795162367, \
1040187647, 1023410431, 2231369983, 4211081471, 1694499071, 0, 369099007, 3456106751, \
3825205503, 1174405375, 872415487, 872415487, 872415487, 872415487, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 4294901760, \
4293984270, 2046951677, 3422552319, 4110418175, 4177527039, 3405775103, 1409286399, \
0, 0, 1409286399, 4278190335, 4278190335, 4278190335, 4278190335, 4278190335, \
4278190335, 4278190335, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4294901760, 4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760,4294901760, 4294901760, 4294901760, \
4294901760, 4294901760, 4294901760, 4294901760, 4294901760, 0, 0, 0, 0, 0, 0, 0, 0, \
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,0, 0, \
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4278255360, 4144037632, 4009819904, \
3875602176, 3741384448, 3607166720, 3472948992, 3338731264, 3204513536, 3053518592, \
2936078080, 2801860352, 2650865408, 2516647680, 2382429952, 2264989440, 2113994496, \
```

```
1996553984, 1862336256, 1728118528, 1577123584, 1459683072, 1325465344, 1191247616, \
1040252672, 922812160, 771817216, 637599488, 503381760, 385941248, 234946304, \
100728576, 4278255360, 4144037632, 4009819904, 3875602176, 3724607232, 3607166720, \
3472948992, 3338731264, 3204513536, 3070295808, 2936078080, 2801860352, 2667642624, \
2516647680, 2399207168, 2264989440, 2130771712, 1996553984, 1845559040, 1728118528, \
1593900800, 1459683072, 1308688128, 1191247616, 1057029888, 922812160, 788594432, \
637599488, 503381760, 369164032, 234946304, 117505792, 4278255360, 4144037632, \
4009819904, 3875602176, 3741384448, 3607166720, 3472948992, 3338731264, 3204513536, \
3053518592, 2919300864, 2801860352, 2650865408, 2533424896, 2399207168, 2264989440, \
2113994496, 1996553984, 1862336256, 1728118528,1593900800, 1459683072, 1325465344, \
1191247616, 1040252672, 906034944, 771817216, 654376704, 503381760, 369164032, \
234946304, 117505792, 4278255360, 4144037632, 4009819904, 3858824960, 3741384448, \
3607166720, 3472948992, 3338731264, 3204513536, 3070295808, 2936078080, 2801860352, \
2667642624, 2533424896, 2382429952, 2264989440, 2130771712, 1979776768, 1862336256, \
1728118528, 1577123584, 1442905856, 1325465344, 1191247616, 1040252672, 922812160, \
771817216, 637599488, 503381760, 369164032, 234946304, 100728576, 4278255360, \
4144037632, 4009819904, 3875602176, 3741384448, 3607166720, 3472948992, 3338731264, \
3204513536, 3070295808, 2919300864, 2801860352, 2667642624, 2533424896, 2399207168, \
2264989440, 2113994496, 1996553984, 1862336256, 1728118528, 1593900800, 1442905856, \
1342241795, 1174470400, 1057029888, 906034944, 788594432, 654376704, 503381760, \
385941248, 251723520, 100728576, 4278190335, 4160749823, 4026532095, 3892314367, \
3741319423, 3623878911, 3472883967, 3338666239, 3221225727, 3070230783, 2952790271, \
2818572543, 2667577599, 2533359871, 2399142143, 2264924415, 2147483903, 1996488959, \
1862271231, 1728053503, 1593835775, 1459618047, 1325400319, 1191182591, 1056964863, \
922747135, 788529407, 654311679, 520093951,385876223, 251658495, 117440767, \
4278190335, 4160749823, 4026532095, 3892314367, 3741319423, 3623878911, 3489661183, \
3355443455, 3221225727, 3087007999, 2936013055, 2801795327, 2667577599, 2533359871, \
2399142143, 2281701631, 2130706687, 1996488959, 1862271231, 1728053503, \
1593835775,1459618047, 1325400319, 1191182591, 1056964863, 922747135, 788529407, \
654311679, 520093951, 385876223, 234881279, 100663551, 4278190335, 4160749823, \
4026532095, 3892314367, 3758096639, 3623878911, 3489661183, 3355443455, 3221225727, \
3087007999, 2936013055, 2801795327, 2667577599, 2550137087, 2415919359, 2264924415, \
2130706687, 1996488959, 1862271231, 1728053503, 1593835775, 1459618047, 1325400319, \
1191182591, 1056964863, 922747135, 788529407, 654311679, 503316735, 369099007, \
251658495, 100663551, 4278190335, 4160749823, 4026532095, 3892314367, 3758096639, \
3623878911, 3489661183, 3355443455, 3204448511, 3087007999, 2936013055, 2818572543, \
2667577599, 2533359871, 2399142143, 2264924415, 2130706687, 1996488959, 1879048447, \
1728053503, 1593835775, 1459618047, 1325400319, 1191182591, 1056964863, 922747135, \
788529407, 654311679, 520093951, 385876223, 251658495, 117440767, 4278190335, \
4160749823, 4026532095, 3892314367, 3758096639, 3623878911, 3489661183, 3355443455, \
```

```
3221225727, 3087007999, 2952790271, 2818572543, 2667577599, 2533359871, 2399142143, \
2264924415, 2147483903, 2013266175, 1862271231, 1744830719, 1610612991, 1476395263, \
1342177535, 1191182591, 1056964863, 922747135, 788529407, 654311679, 520093951, \
385876223, 251658495, 100663551, 4294901760, 4160684032, 4026466304, 3909025792, \
3774808064, 3623813120, 3489595392, 3355377664, 3237937152, 3103719424, 2952724480, \
2818506752, 2684289024, 2550071296, 2415853568, 2281635840, 2147418112, 2013200384, \
1878982656, 1744764928, 1593769984, 1476329472,1325334528, 1207894016, 1056899072, \
939458560, 788463616, 654245888, 520028160, 385810432, 251592704, 117374976, \
4294901760, 4177461248, 4043243520, 3909025792, 3774808064, 3640590336, 3506372608, \
3355377664, 3221159936, 3086942208, 2952724480, 2818506752, 2701066240, 2550071296, \
2415853568, 2281635840, 2147418112, 2013200384, 1878982656, 1727987712, 1610547200, \
1476329472, 1325334528, 1191116800, 1073676288, 922681344, 788463616, 654245888, \
520028160, 385810432, 251592704, 100597760, 4294901760, 4177461248, 4043243520, \
3909025792, 3774808064, 3640590336, 3489595392, 3372154880, 3237937152, 3103719424, \
2952724480, 2818506752, 2700935170, 2550071296, 2415853568, 2281635840, 2147418112, \
2013200384, 1878982656, 1744764928, 1610547200, 1459552256, 1342111744, 1191116800, \
1056899072, 922681344, 788463616, 671023104, 520028160, 385810432, 251592704, \
100597760, 4294901760, 4177461248, 4043243520, 3909025792, 3774808064, 3640590336, \
3489595392, 3372154880, 3237937152, 3086942208, 2969501696, 2818506752, 2684289024, \
2550071296, 2432630784, 2281635840, 2147418112, 2013200384, 1862205440, 1744764928, \
1610547200, 1476329472, 1342111744, 1191116800, 1056899072, 922681344, 788463616, \
654245888, 520028160, 385810432, 251592704, 117374976, 4294901760, 4177461248, \
4043243520, 3909025792, 3774808064, 3623813120, 3506372608, 3372154880, 3237937152, \
3103719424, 2952724480, 2835283968, 2684289024, 2550071296, 2432630784, 2281635840, \
2147418112, 2046492676, 1862205440, 1744764928, 1610547200, 1476329472, \
1342111744,1207894016, 1056899072, 939458560, 788463616, 654245888, 536281096, \
385810432, 251592704, 134152192};

    Display *d = XOpenDisplay(0);
    int s = DefaultScreen(d);
    Atom net_wm_icon = XInternAtom(d, "_NET_WM_ICON", False);
    Atom cardinal = XInternAtom(d, "CARDINAL", False);
    Window w;
    XEvent e;
    w = XCreateWindow(d, RootWindow(d, s), 0, 0, 200, 200, 0,
                      CopyFromParent, InputOutput, CopyFromParent, 0, 0);

    int length = 2 + 16 * 16 + 2 + 32 * 32;
    XChangeProperty(d, w, net_wm_icon, cardinal, 32,
                    PropModeReplace, (const unsigned char*) buffer, length);


    XMapWindow(d, w);
    while(1) XNextEvent(d, &e);
}
```

C.4.6.8 Experience with _NET_WM_ICON Hint

First, we need the libraries:

```
root-shell# aptitude install libstdc++6 libX11-6
```

Second, a Makefile can save much typing:

```
shell$ Makefile
compile:
        gcc -lX11 -lstdc++ -L/usr/X11/lib -o native native.cpp
clean:
        rm -f *~
```

Finally, we run:

```
shell$ make
gcc -lX11 -lstdc++ -L/usr/X11/lib -o native native.cpp
shell$ ./native
```

The X server does render this window.

- The drawable area is transparent, of course, as the X client draws nothing.
- The window frame has the usual border, title bar, and buttons (the window manager requests the X server to draw these).
- However the Title Bar Icon is still the default "X" icon.
- This is also true of the alt-tab icon.

It is unclear why the window manager, kwin, is not using the _NET_WM_ICON hint.

C.5 Mathematical Equations

Many wikipedia articles present mathematical equations. This is especially true of articles about math, science, and engineering. Often these equations are delimited by the tags:

```
<math>y = mx + b</math>
```

where the equation between the tags is formatted using TEX.

C.5.1 Example: the Article on Algebra

The article on Algebra contains a few simple equations formatted this way. This can be seen as follows:

First, find the page.page_id:

```
mysql> USE simplewiki;
mysql> SHOW CREATE TABLE page\G
*************************** 1. row ***************************
       Table: page
Create Table: CREATE TABLE `page` (
  `page_id` int(10) unsigned NOT NULL AUTO_INCREMENT,         <--- 25
  `page_namespace` int(11) NOT NULL,
  `page_title` varbinary(255) NOT NULL,                       <--- Algebra
  `page_restrictions` tinyblob NOT NULL,
  `page_counter` bigint(20) unsigned NOT NULL DEFAULT '0',
  `page_is_redirect` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `page_is_new` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `page_random` double unsigned NOT NULL,
  `page_touched` binary(14) NOT NULL DEFAULT '\0\0\0\0\0\0\0\0\0\0\0\0\0\0',
  `page_latest` int(10) unsigned NOT NULL,
  `page_len` int(10) unsigned NOT NULL,
  PRIMARY KEY (`page_id`),
  UNIQUE KEY `name_title` (`page_namespace`,`page_title`),
  KEY `page_random` (`page_random`),
  KEY `page_len` (`page_len`),
  KEY `page_redirect_namespace_len` (`page_is_redirect`,`page_namespace`,`page_len`)
) ENGINE=InnoDB AUTO_INCREMENT=945 DEFAULT CHARSET=binary
1 row in set (0.00 sec)
```

```
mysql> SELECT page_id,page_title FROM page WHERE page_title LIKE 'Algebra';
+---------+------------+
| page_id | page_title |
+---------+------------+
|      25 | Algebra    |
+---------+------------+
1 row in set (0.00 sec)
```

Second, find the `revision.rev_id`:

```
mysql> SHOW CREATE TABLE revision\G
*************************** 1. row ***************************
       Table: revision
Create Table: CREATE TABLE `revision` (
  `rev_id` int(10) unsigned NOT NULL AUTO_INCREMENT,          <--- 25
  `rev_page` int(10) unsigned NOT NULL,                       <--- 25
  `rev_text_id` int(10) unsigned NOT NULL,                    <--- 25
  `rev_comment` tinyblob NOT NULL,
  `rev_user` int(10) unsigned NOT NULL DEFAULT '0',
  `rev_user_text` varbinary(255) NOT NULL DEFAULT '',
  `rev_timestamp` binary(14) NOT NULL DEFAULT '\0\0\0\0\0\0\0\0\0\0\0\0\0\0',
  `rev_minor_edit` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `rev_deleted` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `rev_len` int(10) unsigned DEFAULT NULL,
  `rev_parent_id` int(10) unsigned DEFAULT NULL,
  `rev_sha1` varbinary(32) NOT NULL DEFAULT '',
  PRIMARY KEY (`rev_id`),
  UNIQUE KEY `rev_page_id` (`rev_page`,`rev_id`),
  KEY `rev_timestamp` (`rev_timestamp`),
  KEY `page_timestamp` (`rev_page`,`rev_timestamp`),
  KEY `user_timestamp` (`rev_user`,`rev_timestamp`),
  KEY `usertext_timestamp` (`rev_user_text`,`rev_timestamp`)
) ENGINE=InnoDB AUTO_INCREMENT=945 DEFAULT CHARSET=binary MAX_ROWS=10000000 AVG_ROW_LENGTH=1024
1 row in set (0.00 sec)
```

```
mysql> SELECT rev_id,rev_page,rev_text_id FROM revision WHERE rev_page=25;
+--------+----------+-------------+
| rev_id | rev_page | rev_text_id |
+--------+----------+-------------+
|     25 |       25 |          25 |
+--------+----------+-------------+
1 row in set (0.01 sec)
```

Third, find the `text.old_id`:

```
mysql> SHOW CREATE TABLE text\G
*************************** 1. row ***************************
       Table: text
Create Table: CREATE TABLE `text` (
  `old_id` int(10) unsigned NOT NULL AUTO_INCREMENT,          <--- 25
  `old_text` mediumblob NOT NULL,                             <--- <math>...</math>
  `old_flags` tinyblob NOT NULL,
  PRIMARY KEY (`old_id`)
) ENGINE=InnoDB AUTO_INCREMENT=945 DEFAULT CHARSET=binary MAX_ROWS=10000000 AVG_ROW_LENGTH=10240
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM text WHERE old_id=25\G
*************************** 1. row ***************************
   old_id: 25
 old_text: complex|date=February 2010
...
== Graphing algebra ==
Algebra also introduces graphing, or drawing a picture that shows all the
values of the variables that make the equation true. Usually this is easy to do
when there are only one or two variables. The graph is often a line, and if the
line does not bend or go straight up-and-down it can be described by the basic
formula <math>y = mx + b</math> where ''b'' is the [[y-intercept]] of the graph
and ''m'' is the [[slope]]. This formula applies to the coordinates of the
graph or <math>(x, y)</math>.
...
```

Fourth, Math.php invokes texvc to generate the:

- md5sum (with a leading 'C' character),
- HTML string,
- MathML string, and
- PNG image file.

```
root-shell# texvc '/var/lib/mediawiki/images/tmp/' '/var/lib/mediawiki/images/tmp/' \
 'y = mx + b' 'UTF-8'
Cee668a8b96a7fc367f89135101df6c90
<i>y</i> = <i>m</i><i>x</i> + <i>b</i>
<mi>y</mi><mo>=</mo><mi>m</mi><mi>x</mi><mo>+</mo><mi>b</mi>
root-shell# ls -l /var/lib/mediawiki/images/tmp/
ee668a8b96a7fc367f89135101df6c90.png
root-shell# mv /var/lib/mediawiki/images/tmp/ee668a8b96a7fc367f89135101df6c90.png \
/var/lib/mediawiki/images/math/e/e/6/.
```

It seems that the md5sum is computed after removing white space:

```
env printf %s "y = mx + b" | openssl dgst -md5         <--- wrong
(stdin)= 6d74d8f1c6d1196d2e75893f266ae552
env printf %s "y=mx+b" | openssl dgst -md5             <--- right
(stdin)= ee668a8b96a7fc367f89135101df6c90
```

Fifth, the first three characters in the md5sum are used to create a directory path where the PNG image file will be stored. In this case, the PNG image file is stored under /var/lib/mediawiki/images/math/e/e/6/.

Sixth, the metadata is stored in the simplewiki.math database table:

```
mysql> SHOW CREATE TABLE math\G
*************************** 1. row ***************************
       Table: math
Create Table: CREATE TABLE `math` (
  `math_inputhash` varbinary(16) NOT NULL,
  `math_outputhash` varbinary(16) NOT NULL,
  `math_html_conservativeness` tinyint(4) NOT NULL,
  `math_html` blob,
  `math_mathml` blob,
  UNIQUE KEY `math_inputhash` (`math_inputhash`)
) ENGINE=InnoDB DEFAULT CHARSET=binary
1 row in set (0.00 sec)
```

```
mysql> SELECT HEX(math_inputhash),hex(math_outputhash),math_html_conservativeness,
 math_html,math_mathml FROM simplewiki.math WHERE HEX(math_inputhash) LIKE 'ee66%'\G
*************************** 1. row ***************************
       hex(math_inputhash): EE668A8B96A7FC367F89135101DF6C90
      hex(math_outputhash): EE668A8B96A7FC367F89135101DF6C90
math_html_conservativeness: 2
                 math_html: <i>y</i> = <i>m</i><i>x</i> + <i>b</i>
               math_mathml: <mi>y</mi><mo>=</mo><mi>m</mi><mi>x</mi><mo>+</mo><mi>b</mi>
1 row in set (0.00 sec)
```

Finally, when the page is rendered by a browser, math tags:

```
<math>y = mx + b<math>
```

are replaced with image tags:

```
<img class="tex" alt="y = mx + b" src="/images/math/e/e/6/ee668a8b96a7fc367f89135101df6c90.png" />
```

C.5.2 LaTeX

LaTeX is a macro package built on top of TeX, which is a typesetting system. LaTeX does a professional job of typesetting mathematical equations, chemical formulae, etc. The American Mathematical Society uses LaTeX for its publications.

C.5.3 Math.php and texvc

The MediaWiki extension Math.php, is located in /var/lib/mediawiki/extensions/. It invokes the utility texvc, to take a mathematical equation in TeX format and generate the:

- md5sum (with a leading 'C' character),
- HTML string,
- MathML string, and
- PNG image file.

The Math.php extension directs texvc to use /var/lib/mediawiki/images/tmp/ as a scratch space.

The image files produced by texvc are stored in a directory tree under /var/lib/mediawiki/images/math/.

C.5.3.1 Update (2012)

The code that converts math equations from TeX format into PNG image files, has been removed from the MediaWiki 1.18 core, and is now packaged separately. This is described in http://www.mediawiki.org/wiki/Extension:Math and in http://www.mediawiki.org/wiki/Manual:Enable_TeX.

Caveat 1: The Debian package mediawiki-math-texvc installs texvc in /usr/bin/texvc. Math.php, however, expects to find texvc in /var/lib/mediawiki/extensions/Math/math/texvc.

The Math/math/ directory must be created.

```
root-shell# cd /var/lib/mediawiki/extensions/Math/
root-shell# mkdir math
root-shell# cd math/
root-shell# ln --symbolic /usr/bin/texvc texvc
root-shell# ls -l
lrwxrwxrwx 1 root root 14 Nov 12 12:57 texvc -> /usr/bin/texvc
```

If this is not done, the page be will rendered with error messages like:

```
Failed to parse (Missing texvc executable; please see math/README to configure.): y = mx + b
```

Caveat 2: The *xx*wiki.math database table has been removed from the MediaWiki 1.18 core. However, if Math.php is installed, then the missing math database table can be created by running:

```
root-shell# /usr/bin/php /usr/share/mediawiki/maintenance/update_farm.php \
simple.wpmirror.site
```

WP-MIRROR 0.4 automates the configuration required by the above two caveats.

C.5.4 Messages

When the simplewiki database does not exist, we get:

```
            Sorry! This site is experiencing technical difficulties.

   Try waiting a few minutes and reloading.

   (Can't contact the database server: Access denied for user
   'wikiuser'@'localhost' to database 'simplewiki' (localhost))
...
```

When the simplewiki database exists, but the page table does not contain the article, we get:

```
shell$ lynx http://simple.wpmirror.site/index.php/Algebra
...
Algebra

   From simplewiki
   Jump to: navigation, search

   There is currently no text in this page. You can search for this page
   title in other pages, search the related logs, or edit this page.
   Retrieved from "http://simple.wpmirror.site/index.php/Algebra"
...
```

When the article is in the database, but Math.php does not regard texvc as executable, we get:

```
shell$ lynx http://simple.wpmirror.site/index.php/Algebra
...
   to do when there are only one or two variables. The graph is often a
   line, and if the line does not bend or go straight up-and-down it can be
   described by the basic formula Failed to parse (Missing texvc executable;
   please see math/README to configure.): y = mx + b
   where b is the y-intercept of the graph and m is the slope. This formula
   applies to the coordinates of the graph or Failed to parse (Missing texvc
   executable; please see math/README to configure.): (x, y)
...
```

C.6 Prioritizing Tasks (Obsolete)

Which is better? 1) First download images and then import articles, or 2) the reverse? The author prefers the first, and here is the reasoning.

When a browser accesses a wikipedia article featuring an image, that image must first be resized. These resized images, called 'thumb's, come in different sizes, and once created they are stored (cached) in a directory tree under `/var/lib/mediawiki/images/thumb/`.

The image processing required to produce these thumbs is quite demanding in terms of time and memory. Moreover, if one uses `convert` from `ImageMagick` (instead of `gm convert` from `GraphicsMagick` and `inkscape`), the image processing can also hang the system.

So here are the design choices:

- **First Import Articles and Then Download Images.** In this case, the thumbs displayed with a given article will be created when that article is first accessed with a browser. More precisely, to serve an article, the `apache2` web-server calls the `MediaWiki PHP` script `/var/lib/mediawiki/index.php` which, for each missing thumb, forks a `convert` process, then blocks until the thumbs are all available, and finally renders the whole article. This means the browsing experience will be marred by delays, time-outs, and even system crashes when accessing certain articles.
- **First Download Images and Then Import Articles.** Alternatively, the thumbs are all created during importation. While all this image processing may be a pain up front, thereafter the thumbs will be readily available to the web-server. This means the browsing experience will be more enjoyable.

The author of WP-MIRROR choose the later design option.

Update 2013: As of WP-MIRROR 0.6, `importDump.php` is no longer used. Best performance is now achieved by first importing articles, then importing images.

C.7 Scraping Image File Names from `xchunk`s (Obsolete)

`fsm-file-scrape` scrapes image file names from `xchunk`s. How this is done is a little complicated because there is, as far as I know, no grammar (such as Bachus-Naur Form) that describes a wikipedia page. Basically, we have a lot of special cases.

Update 2013: As of WP-MIRROR 0.6, `fsm-file-scrape` is no longer used. Rather, image file names are scraped from `xxwiki.imagelinks.il_to` and `commonswiki.image.img_name`.

C.7.1 Links, Gallery, Infobox, and other Templates

C.7.1.1 Link

About 65% of the image file names are found in links:

```
... [[File:foo.png| ... ]] ...
... [[Image:bar.png| ... ]] ...
... [[Media:baz.png| ... ]] ...
```

C.7.1.2 Gallery

About 10% are found in galleries, of which there are two kinds:

```
<gallery>
File:foo.png| ...
Image:bar.png| ...
Media:baz.png| ...
<\gallery>
```

and

```
{{gallery
|File:foo.png| ...
|Image:bar.png| ...
|Media:baz.png| ...
}}
```

Caveat: The File, Image, Media tags are not always capitalized.

See http://simple.wpmirror.site/index.php/Berlin for galleries (one of landmarks, one of flags).

C.7.1.3 Infobox Template

The remaining 25% of image file names are found mostly in the infobox template.

```
{{Infobox
...
| image_flag = foo.png
| image_coat = bar.png
| image_map  = baz.png
...
}}
```

or even

```
{{Infobox
...
| image_flag
= foo.png
| image_coat
= bar.png
| image_map
= baz.png
...
}}
```

with arbitrary amounts of white space. And beware the mixing of links with templates.

```
{{Infobox
...
| image = [[File:foo.png]]
|     map=[[Image:bar.png]]
|coat
= [[Media:baz.png]]
...
}}
```

See http://simple.wpmirror.site/index.php/Berlin for an infobox in the upper right corner of the page.

C.7.1.4 Other Templates

Very few (less than 1 per mil) are found in the multiple image and wide image templates.

```
{{multiple image
...
| image1 = foo.png
...
| image2 = bar.png
...
}}
```

See http://simple.wpmirror.site/index.php/Neptune for groups of three images using the multiple image template.

```
{{wide image|File:foo.png| ... }}
{{wide image|Image:foo.png| ... }}
```

See http://simple.wpmirror.site/index.php/London for a panoramic view of London using the wide image template.

Source: See http://en.wikipedia.org/wiki/Wikipedia:Picture_tutorial.

C.7.2 Image File Name Extensions

A candidate file name must be checked for a known file extension: bmp, bz2, dia, djvu, fig, gif, jpg, jpeg, ogg, png, tif, tiff, mid, mov, mp3, pdf, psp, svg, wav, and xcf. If such a file extension is not found, the candidate is discarded.

C.7.3 Normalizing Image File Names

C.7.3.1 Standards

RFC3986, Uniform Resource Identifier: Generic Syntax defines "percent-encoding" and lists the reserved characters.

C.7.3.2 White Space

Many image file names contain spaces. These spaces must be replaced with underscores. For example,

```
Arc en ciel.png
```

becomes

```
Arc_en_ciel.png
```

Any leading or trailing white space is also trimmed.

C.7.3.3 Special Characters

HTTP. According to HTTP/1.1 [RFC2616], page 17, the following characters are defined as 'separators':

```
separators     = "(" | ")" | "<" | ">" | "@"
               | "," | ";" | ":" | "\" | <">
               | "/" | "[" | "]" | "?" | "="
               | "{" | "}" | SP | HT
```

but offers three ways of quoting:

```
Comments can be included in some HTTP header fields by surrounding
the comment text with parentheses. Comments are only allowed in
fields containing "comment" as part of their field value definition.
In all other fields, parentheses are considered part of the field
value.

    comment         = "(" *( ctext | quoted-pair | comment ) ")"
    ctext           = <any TEXT excluding "(" and ")">

A string of text is parsed as a single word if it is quoted using
double-quote marks.

    quoted-string   = ( <"> *(qdtext | quoted-pair ) <"> )
    qdtext          = <any TEXT except <">>

The backslash character ("\") MAY be used as a single-character
quoting mechanism only within quoted-string and comment constructs.

    quoted-pair     = "\" CHAR
```

dash. According to the man page for dash (the default POSIX compliant /bin/sh for Debian scripts) the following characters have special meanings:

```
      Control operators:
            & && ( ) ; ;; | || <newline>
      Redirection operators:
            < > >| << >> <& >& <<- <>
      Reserved words:
            ! { }
      Special parameters:
            * @ # ? -(Hyphen.) $ ! 0 (Zero.)
      Command Substitution
            $(command)  `command`
      Shell patterns:
            ! * ? [ ]
      Builtins
            : . [ ]
      Character escape sequences (as defined in ANSI X3.159-1989):
            \a \b \f \n \r \t \v \\ \num (1-, 2-, or 3-digit octal)
```

but offers three ways of quoting:

```
  Quoting
    Quoting is used to remove the special meaning of certain characters or
    words to the shell, such as operators, whitespace, or keywords.  There are
    three types of quoting: matched single quotes, matched double quotes, and
    backslash.

  Backslash
    A backslash preserves the literal meaning of the following character, with
    the exception of newline.  A backslash preceding a newline is treated
    as a line continuation.

  Single Quotes
```

```
  Enclosing characters in single quotes preserves the literal meaning of all
  the characters (except single quotes, making it impossible to put single-
  quotes in a single-quoted string).

Double Quotes
  Enclosing characters within double quotes preserves the literal meaning of
  all characters except dollarsign ($), backquote (‘), and backslash (\).
  The backslash inside double quotes is historically weird, and serves to
  quote only the following characters:
        $ ‘ " \ <newline>.
  Otherwise it remains literal.
```

MySQL. According to the MySQL 5.5 Reference Manual, Table 9.1, the following are ‘special character escape sequences’.

```
  Special Character Escape Sequences
     \0 \’ \" \b \n \r \t \Z \\ \% \_
```

and also offers ways of quoting: matched backquotes (identifier quotes), matched single quotes (string quotes), matched double quotes (ANSI identifier quotes, non-ANSI string quotes), and backslash (escape a single character).

C.7.3.4 Image File Names

Nearly all of the special characters mentioned above can be found in image file names. This is especially true of images found in the en wikipedia.

Image file names containing such characters (and a few additional ones) should be escaped to avoid confusing the shell, MySQL, web browser, web proxy, and web server. By *escaped* we mean, that their special meaning is somehow removed. Usually this involves: prefixing the special character with a backslash (\), or enclosing it with matched single quotes or matched double quotes.

The special characters that we have studied are listed in Table C.1, Special Characters. The image file names containing special characters in the lower part of the table are successfully downloaded. The image file names containing special characters in the upper part of the table are not attempted.

Even a character that has been escaped for the shell may be problematic for other reasons:

- **ampersand (&)**: could be an HTML Special Entity Code, aka “AMP code” (like & © > <   ");
- **angle brackets (<,>)**: troublesome to parse because it might be an XML tag in the xchunk; and it must be replaced with an AMP code for use with cURL;
- **asterisk (*)**: shell wild card, can be escaped and downloaded with cURL, but then becomes troublesome to have on a UNIX file system where it is a wild card;
- **backquote (‘)**: shell command substitution; this can be escaped and downloaded, but it is always an hazard to have and to handle (think hard before trying the following slightly risky example, and remember that THE AUTHOR ASSUMES NO LIABILITY):

  ```
  shell$ curl http://foo‘ls‘bar.jpg
  ```

  and there seems to be a problem computing the md5sum if the other characters are unicode;
- **backslash (\)**: used as escape character for shell, HTTP, and MySQL; denotes line continuation for shell;
- **braces ({,})**: troublesome to parse because it might be a MediaWiki template in the xchunk; braces are also used by cURL’s “URL globbing parser” and therefore would have to be “URL encoded”;
- **colon (:)**: can be escaped and downloaded, but then becomes troublesome for common lisp where it indicates a package name;

Table C.1: Special Characters

Name	Char	Esc	Keep	shell	path	HTTP	lisp	MediaWiki	SQL	Other
ampersand	&	Y	N	control		GET				HTML AMP
angle brackets	<,>		N	redirect				xml tag		
asterisk	*	Y	N	wild	wild					
backquote	‘	Y	N	cmd subst					quote	
baskslash	\	Y	N	esc	dir	esc			esc	
braces	{,}	Y	N	reserved				template		
colon	:		N	builtin	sep	URI	package	sep		
percent	%		N						wild	
pipe	|		N	redirect			quote	sep		
question mark	?		N			GET	wild			
slash	/		N		dir	URI				
square brackets	[,]	Y	N	builtin				link		
tilde	∼		N	expand						
apostrophe	’	Y	Y	quote					quote	polipo
at	@	Y	Y	parameter						
caret	^	N	Y	regexp						
comma	,	N	Y							
dash	-	Y	Y	option						
dollar	$	Y	Y	regexp						
equal	=	N	Y			GET				
exclamation	!	Y	Y	reserved						
parentheses	(,)	Y	Y	control						
quote	”	Y	Y	quote						
semicolon	;	Y	Y	control					control	
underscore	_	Y	Y						wild	

- **percent (%)**: MySQL wild card; can appear in file names and URLs that have been “percent-encoded”, and are easy to download; however, they are troublesome to have in the database, and the Wikimedia Foundation has tried to stamp them out (there seem to be a few remaining in the en wikipedia);
- **pipe (|)**: shell redirection operator, and used by common lisp for quoting;
- **question mark (?)**: common lisp wild card; can be escaped and downloaded with cURL, but then becomes troublesome for common lisp where it is a wild card;
- **slash (/)**: confuses shell where it could indicate a directory or URL; and
- **square brackets ([,])**: troublesome to parse because it might be a MediaWiki link; square brackets are used by cURL’s “URL globbing parser” and therefore would have to be “URL encoded”
- **tilde (∼)**: used by shell for ‘Tilde Expansion’ (substituting user’s home directory).

Image file names containing the above characters are dropped from further consideration. It is possible that, in a future release, some of the above characters could be handled.

C.7.3.5 Apostrophe

The **apostrophe (’)** is an interesting case. It appears in about 1% of image file names, which makes getting it right a priority:

- Glottal stop Nisga’a Memorial Lava Bed Provincial Park.jpg,
- Irish surname Peter O’Toole.jpg,
- Possessive case Saint Peter’s Square from the dome.jpg.

Whereas backslash escapes only one character, a matched pair of apostrophes escapes the enclosed string. This escaping can itself be escaped by placing a backslash before each apostrophe (and this we do).

This appears to work well—unless your traffic passes through a caching web proxy. It turns out that an URL with an escaped apostrophe confuses the caching web proxy polipo, which then returns a file containing the following error message:

```
root-shell# cd /var/lib/mediawiki/images/bad-images/f/fa/
root-shell# ls -l Flag_of_the_People's_Republic_of_China.svg
-rw-r--r-- 1 root root 449 Nov 28 18:44 Flag_of_the_People's_Republic_of_China.svg
root-shell# cat Flag_of_the_People's_Republic_of_China.svg
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<title>Proxy result: 302 Redirected by external redirector.</title>
</head><body>
<h1>302 Redirected by external redirector</h1>
<p>The following status was returned:<br><br>
<strong>302 Redirected by external redirector</strong></p>
<hr>Generated Wed, 28 Nov 2012 18:44:08 EST by Polipo on <em>darkstar-7:8123</em>.
</body></html>
```

fsm-images-validate detects such files by scanning any file smaller than 1k for the string 302 Redirected. Any file meeting that criterion is then sequestered under the bad-images directory tree.

C.7.4 Manual Testing

It is possible to test manually whether or not special characters have been properly escaped:

1. Compute the md5sum,

```
shell$ printf %s US_\$10_Series_2004_reverse.jpg | openssl dgst -md5
(stdin)= 30294ae0d622e6e1e068660c639e3c85
```

2. See if cURL can download the file:

```
shell$ curl --output test.jpg \
http://upload.wikimedia.org/wikipedia/commons/3/30/US_\$10_Series_2004_reverse.jpg
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 28553  100 28553    0     0   233k      0 --:--:-- --:--:-- --:--:--  449k
```

3. Look in the output file (test.jpg in this case) to see if there is an error message. In this next example, we give the wrong directory path (/3/31/ instead of /3/30/) to provoke an error message:

```
shell$ curl --output test.jpg  \
http://upload.wikimedia.org/wikipedia/commons/3/31/US_\$10_Series_2004_reverse.jpg
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   285  100   285    0     0   2402      0 --:--:-- --:--:-- --:--:--  4672
shell$ cat test.jpg
<html>
 <head>
  <title>404 Not Found</title>
 </head>
 <body>
  <h1>404 Not Found</h1>
  The resource could not be found.<br /><br />
File not found: /v1/AUTH_43651b15-ed7a-40b6-b745-47666abf8dfe/
wikipedia-commons-local-public.31/3/31/US_\$10_Series_2004_reverse.jpg
 </body>
</html>
```

Note also the difference in file size. Error messages are less than 1k.

See Table C.2, Image File Names with Special Characters for examples that may be tested.

C.7.5 Downloading Image Files Efficiently

C.7.5.1 Checking if File is Already Downloaded

Apart from its first run, WP-MIRROR usually updates an existing mirror. Therefore most image files are already available, and their existence need only be confirmed to avoid repeating a lot of work.

The `md5sum` of each image file name is computed and used for deciding where the image should be stored and retrieved:

```
shell$ env printf %s file-name | openssl dgst -md5
```

The first two hexadecimal digits of the `md5sum` are used to create the directory tree under which the image files are stored. For example, hashing the normalized image file name `Arc_en_ciel.png` yeilds

```
shell$ env printf %s Arc_en_ciel.png | openssl dgst -md5
(stdin)= 00135a44372c142bd509367a9f166733
```

and therefore, `Arc_en_ciel.png` would be stored under `/var/lib/mediawiki/images/0/00/`. This is where WP-MIRROR would look to determine if the file already exists.

C.7.5.2 Eliminating Duplicates

When a normalized image file name is determined to be new (i.e. the file has not already been downloaded), it is added to an hash table: where the `key` is the normalized file name (e.g. `Arc_en_ciel.png`) and the `value` is the directory path (e.g. `/0/00/`). One advantage of using a hash table, is that it never holds two entries with the same key.

Design note. Of course, we must define what we mean when we say that two keys are the same. `Common Lisp` has four different tests for equality: `eq`, `eql`, `equal`, and `equalp`. In our hash table, the keys are file names which are strings. Therefore, we use the `equal` test, rather than the default `eql`. When comparing two strings:

- the value of `equal` is true if the two strings match character-by-character; whereas

Table C.2: Image File Names with Special Characters

Char	Version	File Name
Apostrophe (')	URL	http://simple.wpmirror.site/index.php/European_Union
	xchunk	Quai␣d'Orsay.jpg
	normalized	Quai_d\'Orsay.jpg
	md5sum	6090d234ab9c1e7d842f648136c69ab5
At (@)	URL	http://simple.wpmirror.site/index.php/Hatshepsut
	xchunk	KarnakTemple@LuxorEgypt␣obelisk2␣2007feb9-96␣byDanielCsorfoly.JPG
	normalized	KarnakTemple\@LuxorEgypt_obelisk2_2007feb9-96_byDanielCsorfoly.JPG
	md5sum	6fe4389ef3edc536c64d6754eafe7603
Caret (^)	URL	http://simple.wpmirror.site/index.php/House_of_Wettin
	xchunk	Banner␣of␣Saxony␣(1^1).svg
	normalized	Banner_of_Saxony_\(1^1\).svg
	md5sum	46e5c452f2d5c345430073bea9a70f4d
Comma (,)	URL	http://simple.wpmirror.site/index.php/Renewable_resource
	xchunk	Braine␣le␣Chateau,Belgium,moulin␣banal.JPG
	normalized	Braine_le_Chateau,Belgium,moulin_banal.JPG
	md5sum	00154570ef59b3074ea0b42adf0043f8
Dash (-)	URL	http://simple.wpmirror.site/index.php/Pre-Columbian
	xchunk	Uxmal-mexico.jpg
	normalized	Uxmal\-mexico.jpg
	md5sum	001d7a95430654f08049dc61987597a4
Dollar sign ($)	URL	http://simple.wpmirror.site/index.php/United_States_dollar
	xchunk	US␣$10␣Series␣2004␣reverse.jpg
	normalized	US_\$10_Series_2004_reverse.jpg
	md5sum	30294ae0d622e6e1e068660c639e3c85
Equal (=)	URL	http://simple.wpmirror.site/index.php/Planet
	xchunk	Gas␣giants␣and␣the␣Sun␣(1␣px␣=␣1000␣km).jpg
	normalized	Gas_giants_and_the_Sun_\(1_px_=_1000_km\).jpg
	md5sum	bee860e6fab92ee90a67c66b10377cb4
Exclamation (!)	URL	http://simple.wpmirror.site/index.php/Religion
	xchunk	Sikhs␣on␣the␣move!.jpg
	normalized	Sikhs_on_the_move\!.jpg
	md5sum	9805c1725a16fc06ef33b32686a284fa
Parentheses	URL	http://simple.wpmirror.site/index.php/Soprano
	xchunk	Maria␣Callas␣(La␣Traviata).JPG
	normalized	Maria_Callas_\(La_Traviata\).JPG
	md5sum	00d64fd7148552b18e24760e8d4fd99a
Quote (")	untested	
Semicolon (;)	URL	http://simple.wpmirror.site/index.php/St_Peter's_College,_Auckland
	xchunk	St␣Peter's␣College,␣Auckland;␣Bro␣O'Driscoll␣Building.JPG
	normalized	St_Peter\'s_College,_Auckland\;_Bro_O\'Driscoll_Building.JPG
	md5sum	1acdc6e79eb46e5f35a7591a5b7c3021
Underscore (_)	URL	http://simple.wpmirror.site/index.php/Neptune
	xchunk	Neptune_darkspot.jpg
	normalized	Neptune_darkspot.jpg
	md5sum	51dafcb702c1d37434859c4181dd2cae

- the value of eql is true only if the two strings are the same object (that is, occupy the same location in memory).

C.7.5.3 Generating the shell Script

Thanks to the hash table, there are no duplicates within any given ichunk. However, when one compares two or more ichunks, there may be duplicates. Since, ichunks may be run concurrently, we need a way to avoid downloading the same file multiple times. For this reason, the script for downloading an image file, first checks to see if the file is already available.

Also, image files may be stored either in a commons directory tree, or in a directory tree specific to that wikipedia. Every wikipedia, with the possible exception of the en wikipedia, stores most if its images in the commons. Therefore, it saves time to try downloading the image file from the commons first; and, failing that, go on to try downloading from the directory tree specific to that wikipedia.

For example, variables such as these:

```
language-code   <-- simple
path            <-- /0/00/Arc_en_ciel.png
key             <-- Arc_en_ciel.png
value           <-- /0/00/
```

are filled into a template to generate a working shell script.

WP-MIRROR 0.6 introduced a new template to make use of HTTP/1.1 "persistent connections" (see §C.7.5.4, Downloading with Persistent Connections):

```
#!/bin/sh
COMMONSPATH=http://upload.wikimedia.org/wikipedia/commons/
IMAGEPATH=http://upload.wikimedia.org/wikipedia/language-code/

sleep 1

curl -f -m 3000 \
$COMMONSPATH./path -o ./path \
...
if [ -e ./path ]; then
        echo ./path downloaded >> download.log
else
        curl -f -m 3000 $IMAGEPATH./path -o ./path
        if [ -e ./path ]; then
                echo ./path downloaded >> download.log
        else
                echo ./path failed >> failed.log
        fi
fi
...
```

WP-MIRROR 0.2 through 0.5, used a template that closely resembles that of the original wikix (which is described in §A.1.5, Changes in WP-MIRROR 0.2 (2011-12-25)):

```
#!/bin/sh
COMMONSPATH=http://upload.wikimedia.org/wikipedia/commons/
IMAGEPATH=http://upload.wikimedia.org/wikipedia/language-code/
...
if [ -e path ]; then
        echo ./path already exists >> exists.log
else
        curl -f -m 1000 -O --retry 0 $COMMONSPATH./path
        if [ -e key ]; then
                /bin/mkdir -p value
                /bin/mv ./key value
                echo ./path downloaded >> download.log
        else
                curl -f -m 1000 -O --retry 0 $IMAGEPATH./path
                if [ -e key ]; then
                        /bin/mkdir -p value
                        /bin/mv ./key value
                        echo ./path downloaded >> download.log
                else
                        echo ./path failed >> failed.log
                fi
        fi
fi
```

C.7.5.4 Downloading with Persistent Connections

The first call to cURL, the one that downloads image files from the Wikimedia Commons, has multiple filenames on one command line. This is in order to download them using a single TCP connection, as described in RFC2616, HTTP/1.1. Persistent connections are supported by cURL, and according to the documentation:

> PERSISTENT CONNECTIONS
>
> Specifying multiple files on a single command line will make curl transfer all of them, one after the other in the specified order.
>
> libcurl will attempt to use persistent connections for the transfers so that the second transfer to the same host can use the same connection that was already initiated and was left open in the previous transfer. This greatly decreases connection time for all but the first transfer and it makes a far better use of the network.
>
> Note that curl cannot use persistent connections for transfers that are used in subsequence[sic] curl invokes. Try to stuff as many URLs as possible on the same command line if they are using the same host, as that'll make the transfers faster. If you use an HTTP proxy for file transfers, practically all transfers will be persistent.
>
> http://curl.haxx.se/docs/manual.html, accessed 2012-12-18

Note that there is a limit to how many URLs can be put on one command line. See §C.7.5.5, Limiting Shell Command-line Length.

C.7.5.5 Limiting Shell Command-line Length

The operating system (e.g. Linux) imposes limits to how long a command-line may be. This can be determined by executing:

```
shell$ getconf ARG_MAX
2097152
```

Actually one must deduct for the length of the environment variables. This is done by executing:

```
shell$ echo $(( $(getconf ARG_MAX) - $(env | wc -c) ))
2097152
```

See http://www.cyberciti.biz/faq/argument-list-too-long-error-solution/.

On average, image file names are 27 characters. This can be estimated by executing:

```
root-shell# cd /var/lib/mediawiki/images/0/
root-shell# ls -R | wc -l
4245
root-shell# ls -R | wc -c
116146
root-shell# dc
116146 4245/pq
27
```

Thus:

```
$COMMONSPATH./path -o ./path \
```

on average will be $(2 \times 27) + 23 = 77$ characters long (including the newline), which means we can pack a maximum of $(2097152 - 18)/77 = 27235$ URLs into a single invocation of curl. It appears that the en wikipedia can easily exceed this limit.

Therefore, we impose our own limit of URLs per curl invocation.

C.7.5.6 Parameters for Persistent Connections

WP-MIRROR 0.6 introduces two new parameters: *mirror-ichunk-persistent-connection* with determines how many image files may be downloaded for each persistent connection (default 10); and *mirror-ichunk-post-connection-sleep*, which determines how many seconds the ichunk process will sleep before establishing the next persistent connection (default 1 second).

If maximizing performance were our only concern, then we would set *mirror-ichunk-persistent-connection* to 1000, and *mirror-ichunk-post-connection-sleep* to zero. However, the Wikimedia Foundation can block the IP address of someone who is downloading too aggressively. Their advice has been to avoid downloading more than one page per second:

> **Please do not use a web crawler**
>
> Please do not use a web crawler to download large numbers of articles. Aggressive crawling of the server can cause a dramatic slow-down of Wikipedia.
>
> Sample blocked crawler email
>
> IP address nnn.nnn.nnn.nnn was retrieving up to 50 pages per second from wikipedia.org addresses. Robots.txt has a rate limit of one per second set using the Crawl-delay setting. Please respect that setting. If you must exceed it a little, do so only during the least busy times shown in our site load graphs at http://stats.wikimedia.org/EN/ChartsWikipediaZZ.htm. It's worth noting that to crawl the whole site at one hit per second will take several weeks. The originating IP is now blocked or will be shortly. Please contact us if you want it unblocked. Please don't try to circumvent it - we'll just block your whole IP range.
>
> If you want information on how to get our content more efficiently, we offer a variety of methods, including weekly database dumps which you can load into MySQL and crawl locally at any rate you find convenient. Tools are also available which will do that for you as often as you like once you have the infrastructure in place. More details are available at http://en.wikipedia.org/wiki/Wikipedia:Database_download.
>
> Instead of an email reply you may prefer to visit #mediawiki at irc.freenode.net to discuss your options with our team.
>
> Note that the robots.txt currently has a commented out Crawl-delay:

```
## *at least* 1 second please. preferably more :D
## we're disabling this experimentally 11-09-2006
#Crawl-delay: 1
```

Please be sure to use an intelligent non-zero delay regardless.
http://en.wikipedia.org/wiki/Wikipedia:Database_download

Appendix D

Error Messages

This appendix is mostly of historical interest.

A main design objective for WP-MIRROR 0.4, that configuration of WP-MIRROR and its many dependencies should be entirely automated, led to a number of design decisions, including:

- the user interface should be far less verbose, ideally one line per checkpoint, and
- the log file should not need the highly verbose error messages seen in §D.4, Full Error Messages (Obsolete), because the Reference Manual (this document) is now available.

D.1 Error Codes

Table D.1: WP-MIRROR Error Codes

Code	Intr	Rem
None		

D.2 Error Codes (Obsolete)

- 1000 (`assert-asdf`)
- 1001 (`assert-clisp-features-p`)
- 1002 (`assert-concurrency-limit-xchunk-p`)
- 1003 (`assert-curl-max-time-p`)

  WP-MIRROR uses `curl` to download files. Sometimes however only a partial file is served. In this case `curl` blocks. However, `curl` does offer a time-out to break the block, namely, the `-m/--max-time <seconds>` option.

  Originally, the user was requested to append `--max-time 1000` to the configuration file `/root/.curlrc`. This was troublesome to entry level users, and would not pass `lintian`. Beginning with WP-MIRROR 0.4, you no longer need to configure `curl` if you are using the built-in alternative to `wikix` (which is recommended). This is because the option `-m 1000` is now appended to every `curl` command found in any shell script generated by WP-MIRROR. This renders configuration of `curl` unnecessary.
- 1004 (`assert-database-administrator-credentials-p`)
- 1005 (`assert-database-administrator-account-p`)
- 1006 (`assert-database-administrator-template-or-create`)
- 1007 (`assert-database-wikidb-p`)
- 1008 (`assert-database-wpmirrordb-p`)

  Beginning with WP-MIRROR 0.2, this function was renamed `assert-database-wpmirror-p`. Beginning with WP-MIRROR 0.3, this option was eliminated in favour of `assert-database-wpmirror-or-create`.

Table D.2: WP-MIRROR Error Codes (Obsolete)

Code	Intr	Rem
1000 (assert-asdf)	0.3	0.5
1001 (assert-clisp-features-p)	0.1	0.5
1002 (assert-concurrency-limit-xchunk-p)	0.1	0.5
1003 (assert-curl-max-time-p)	0.1	0.4
1004 (assert-database-administrator-credentials-p)	0.1	0.5
1005 (assert-database-administrator-account-p)	0.1	0.5
1006 (assert-database-administrator-template-or-create)	0.2	0.5
1007 (assert-database-wikidb-p)	0.1	0.5
1008 (assert-database-wpmirrordb-p)	0.1	0.2
1008 (assert-database-wpmirror-p)	0.2	0.3
1009 (assert-database-wpmirrordb-or-create)	0.1	0.2
1009 (assert-database-wpmirror-or-create)	0.2	0.5
1010 (assert-database-wpmirrordb-table-p)	0.1	0.2
1010 (assert-database-wpmirror-table-p)	0.2	0.5
1011 (assert-database-xxwiki-or-create)	0.2	0.5
1012 (assert-dbms-mysql-p)	0.1	0.5
1013 (assert-disk-space-p)	0.1	0.5
1013 (assert-disk-space-if-large-wikipedia-p)	0.5	0.5
1014 (assert-hdd-write-cache-disabled-p)	0.1	0.5
1015 (assert-images-directory-p)	0.1	0.5
1016 (assert-images-bad-directory-or-create)	0.1	0.5
1017 (assert-internet-access-to-wikimedia-site-p)	0.1	0.5
1018 (assert-mediawiki-adminsettings-p)	0.1	0.4
1019 (assert-mediawiki-extensions)	0.1	0.4
1020 (assert-mediawiki-import)	0.2	0.4
1021 (assert-mediawiki-localsettings-p)	0.1	0.5
1022 (assert-mediawiki-localsettings-image)	0.1	0.5
1023 (assert-mediawiki-localsettings-tex)	0.1	0.4
1024 (assert-mediawiki-localsettings-tidy)	0.1	0.5
1025 (assert-php5-suhosin-p)	0.1	0.4
1026 (assert-physical-memory-p)	0.1	0.5
1026 (assert-physical-memory-if-large-wikipedia-p)	0.5	0.5
1027 (assert-utilities-p)	0.1	0.5
1028 (assert-virtual-host-p)	0.2	0.5
1029 (assert-virtual-host-name-resolution-p)	0.2	0.5
1030 (assert-working-directory-or-create)	0.1	0.5
1031 (warn-if-detect-proxy)	0.1	0.5

- 1009 (assert-database-wpmirrordb-or-create)
  Beginning with WP-MIRROR 0.2, this function was renamed assert-database-wpmirror-or-create.
- 1009 (assert-database-wpmirror-or-create)
- 1010 (assert-database-wpmirrordb-table-p)
  Beginning with WP-MIRROR 0.2, this function was renamed assert-database-wpmirror-table-p.
- 1010 (assert-database-wpmirror-table-p)
- 1011 (assert-database-xxwiki-or-create)
- 1012 (assert-dbms-mysql-p)
- 1013 (assert-disk-space-p)
  Beginning with WP-MIRROR 0.5, this function was renamed assert-disk-space-if-large-wikipedia-p.
- 1013 (assert-disk-space-if-large-wikipedia-p)
- 1014 (assert-hdd-write-cache-disabled-p)

- 1015 (assert-images-directory-p)
- 1016 (assert-images-bad-directory-or-create)
- 1017 (assert-internet-access-to-wikimedia-site-p)
- 1018 (assert-mediawiki-adminsettings-p)

  Beginning with MediaWiki 1.19 no longer uses AdminSettings.php. Therefore, WP-MIRROR 0.4 and later, do not assert it.
- 1019 (assert-mediawiki-extensions)

  Beginning with WP-MIRROR 0.4, this function is no longer used, because MediaWiki extensions are directly loaded from the configuration file LocalSettings.php.
- 1020 (assert-mediawiki-import)
- 1021 (assert-mediawiki-localsettings-p)
- 1022 (assert-mediawiki-localsettings-image)
- 1023 (assert-mediawiki-localsettings-tex)

  Beginning with MediaWiki 1.18, the $wgUseTex option is no longer used. Therefore, WP-MIRROR 0.4 and later, do not assert it.
- 1024 (assert-mediawiki-localsettings-tidy)
- 1025 (assert-php5-suhosin-p)

  Beginning with WP-MIRROR 0.4, this function is no longer used, because php5-suhosin was dropped from Debian GNU/Linux 7.0 (wheezy).
- 1026 (assert-physical-memory-p)

  Beginging with WP-MIRROR 0.5, this function was renamed assert-physical-memory-if-large-wikipedia
- 1026 (assert-physical-memory-if-large-wikipedia-p)

  For the largest wikipedias (the top ten as of 2012), WP-MIRROR will not let you start without at least 4G main memory. This is because database performance will slow to a crawl without adequate memory.
- 1027 (assert-utilities-p)

  WP-MIRROR makes use of a couple dozen existing software utilities. This was due to a design decision akin to 'Do not reinvent the wheel'. WP-MIRROR will not let you start in mirror mode if any are missing (see the full error message for the list of utilities).
- 1028 (assert-virtual-host-p)
- 1029 (assert-virtual-host-name-resolution-p)
- 1030 (assert-working-directory-or-create)
- 1031 (warn-if-detect-proxy)

D.3 Full Error Messages

None.

D.4 Full Error Messages (Obsolete)

```
+------------------------------------------------------------------------+
| ERROR:       1000 (assert-asdf)                                        |
| ABSTRACT:    The installed version of clisp is missing key features.   |
| DESCRIPTION: Like all languages, clisp comes with extensions and       |
|              packages that offer features well beyond the minimum for  |
|              compliance with standards.  WP-MIRROR relies on:          |
|              (:asdf2 :asdf :clisp :clx :common-lisp-controller :gettext|
|              :i18n :loop :regexp :screen :syscalls)                    |
|              This error message occurred because the cl-asdf package   |
|              has not been installed, or has not been configured.       |
|              Please run                                                |
|                                                                        |
|                root-shell# aptitude install cl-asdf                    |
|                                                                        |
|              Then read the documentation, which can usually be found   |
|              under /usr/share/doc/cl-asdf/.  Information on            |
|              configuration is usually found in the file README.Debian  |
|              or in a directory containing HTML pages.  Configuration   |
|              may entail adding the following lines to /root/.clisprc:  |
|                                                                        |
|         (load #P"/usr/share/common-lisp/source/cl-asdf/asdf.lisp")     |
|         (push #P"/usr/share/common-lisp/systems/" asdf:*central-registry*)|
|                                                                        |
| GNU/LINUX:   To see the features your clisp installation offers, try:  |
|                shell$ clisp -q                                         |
|                [1]> *features*                                         |
| REFERENCE:   See the WP-MIRROR README for more on installation.        |
+------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------+
| ERROR:       1001 (assert-clisp-features-p)                            |
| ABSTRACT:    The installed version of clisp is missing key features.   |
| DESCRIPTION: Like all languages, clisp comes with extensions and packages|
|              that offer features well beyond the minimum for compliance with|
|              standards.  WP-MIRROR relies on these:                    |
|              (:asdf2 :asdf :clisp :clx :common-lisp-controller :gettext|
|              :i18n :loop :regexp :screen :syscalls)                    |
| GNU/LINUX:   To see what features your clisp installation offers, try: |
|                shell$ clisp -q -q                                      |
|                [1]> *features*                                         |
| REFERENCE:   See the WP-MIRROR README for more on planning your software.|
+------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1002 (assert-concurrency-limit-xchunk-p)                        |
| ABSTRACT:    Unknown innodb-file-format                                      |
| DESCRIPTION: InnoDB has two file formats (as of 2012):                       |
|              o Antelope is uncompressed and handles concurrency well.       |
|                We limit concurrency to lesser of number of CPUs and three.  |
|              o Barracuda is compressed using zlib and does NOT handle        |
|                concurrency well (frequent deadlocks).                        |
|                We limit concurrency to one (that is, we process xchunks      |
|                one-by-one).                                                  |
| REFERENCE:   See the WP-MIRROR README for more on InnoDB.                    |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1003 (assert-curl-max-time-p)                                   |
| ABSTRACT:    Unable to find timeout option in configuration file for curl.   |
| DESCRIPTION: WP-MIRROR uses curl to download files.  Sometimes however       |
|              only a partial file is served.  In this case curl blocks.       |
|              However, curl does offer a time-out to break the block.         |
| GNU/LINUX:   The configuration file for curl is                              |
|                /root/.curlrc                                                 |
|              In this file you should append                                  |
|                --max-time 1000                                               |
| REFERENCE:   See WP-MIRROR README for more on planning your system.          |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1004 (assert-database-administrator-credentials-p)              |
| ABSTRACT:    Unable to find database administrator credentials.              |
| DESCRIPTION: WP-MIRROR reads through the MediaWiki configuration file        |
|              looking for both wikiuser and administrator credentials.        |
|              WP-MIRROR will not let you start without finding these          |
|              credentials there.  WP-MIRROR needs database administrator      |
|              credentials for just two purpose:  1) to grant permissions to   |
|              the wikiuser account---permissions to submit create, read,      |
|              update, delete (CRUD) queries to the wpmirror database, and     |
|              2) to submit the query                                          |
|                mysql> SHOW ENGINE INNODB STATUS G                             |
|              Wikiuser credentials for all other database access.             |
|              Indirectly, one of the MediaWiki maintenance scripts invoked    |
|              by WP-MIRROR, namely, rebuildImages.php, needs database         |
|              administrator credentials.                                      |
| NOTE:        MySQL's out-of-the-box configuration has no password set for    |
|              the database administrator account.  Some people think this is |
|              safe, others deem it unwise.  Please set a password, and enter |
|              it into the MediaWiki configuration file.                       |
| GNU/LINUX:   1) Set the database administrator password:                     |
|              shell$ mysql -u root                                            |
|              mysql> UPDATE mysql.user SET password=PASSWORD('new_pwd')       |
|                     WHERE user='root';                                       |
|              mysql> FLUSH PRIVILEGES;                                        |
|              2) The MediaWiki configuration file is                          |
|                /etc/mediawiki/AdminSettings.php                              |
|              Edit the administrator credentials in this file                 |
|                $wgDBadminuser="root";                                        |
|                $wgDBadminpassword="new_pwd";                                 |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
|              See MySQL 5.1 Reference Manual, available on-line.              |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1005 (assert-database-administrator-account-p)                  |
| ABSTRACT:    Unable to access database administrator account.                |
| DESCRIPTION: WP-MIRROR directly accesses the database administrator          |
|              for just one purpose:  to grant permissions to the wikiuser     |
|              account---permissions to submit create, read, update, delete    |
|              (CRUD) queries to the wpmirror database.                        |
|              WP-MIRROR uses wikiuser credentials for all other database      |
|              access.  Indirectly, one of the MediaWiki maintenance           |
|              scripts used by WP-MIRROR, namely, rebuildImages.php,           |
|              needs database administrator credentials.                       |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1006 (assert-database-template-or-create)                       |
| ABSTRACT:    Unable to find the wikidb database.                             |
| DESCRIPTION: WP-MIRROR uses the newly created wikidb database (and its       |
|              mostly empty tables) as a template for creating a separate      |
|              database for each member of *mirror-languages*.                 |
|              The template is a mysqldump of wikidb, and it is stored as      |
|                /usr/share/mediawiki/maintenance/template.sql.                |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1007 (assert-database-wikidb-p)                                 |
| ABSTRACT:    Unable to find the wikidb database.                             |
| DESCRIPTION: WP-MIRROR uses the newly created wikidb database (and its       |
|              mostly empty tables) as a template for creating a separate      |
|              database for each member of *mirror-languages*.                 |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1008 (assert-database-wpmirror-p)                               |
| ABSTRACT:    Unable to find the wpmirror database.                           |
| DESCRIPTION: WP-MIRROR creates state information about the progress of the   |
|              mirroring, and uses InnoDB to store, retrieve and share it.     |
|              This information is stored as records (one per file) in the     |
|              database table wpmirror.file.                                   |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1009 (assert-database-wpmirror-or-create)                       |
| ABSTRACT:    Unable to create the wpmirror database.                         |
| DESCRIPTION: WP-MIRROR creates state information about the progress of the   |
|              mirroring, and uses InnoDB to store, retrieve and share it.     |
|              This information is stored as records (one per file) in the     |
|              database table wpmirror.file.                                   |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1010 (assert-database-wpmirror-table-p)                         |
| ABSTRACT:    Unable to find tables in wpmirror database.                     |
| DESCRIPTION: WP-MIRROR creates state information about the progress of the   |
|              mirroring, and uses InnoDB to store, retrieve and share it.     |
|              This information is stored as records (one per file) in the     |
|              database table wpmirror.file.                                   |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1011 (assert-database-xxwiki-or-create)                         |
| ABSTRACT:    Unable to find or create one of the xxwiki databases.           |
| DESCRIPTION: WP-MIRROR mirrors needs to be sure that a separate database     |
|              exists for each member of *mirror-languages*.  These databases  |
|              must exist before running MediaWiki maintenance scripts, such   |
|              as importDump.php which inserts articles from a dump file into |
|              the database.                                                   |
|              WP-MIRROR monitors also access these databases to determine     |
|              how many articles, images, etc. have been imported.            |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+-------------------------------------------------------------------------------+
| ERROR:       1012 (assert-dbms-mysql-p)                                       |
| ABSTRACT:    MediaWiki is not using MySQL which WP-MIRROR requires.           |
| DESCRIPTION: WP-MIRROR creates state information about the progress of the    |
|              mirroring, and uses InnoDB to store, retrieve and share it.      |
|              The database is called wpmirror and it is accessed using        |
|              the same crediantials (host, user, and password info) that      |
|              MediaWiki uses for its own database wikidb.  Indeed,            |
|              WP-MIRROR reads through the MediaWiki configuration file to     |
|              obtain these credentials.  So, at least for time being,         |
|              WP-MIRROR expects MediaWiki to use MySQL.                        |
| DESIGN:      MediaWiki is the software that will maintain and serve the      |
|              articles that you mirror.  MediaWiki is usually configured to   |
|              store its articles in one of five database management           |
|              systems (DBMS):  MySQL, postgres, sqlite, mssql, and            |
|              ibm_db2.  WP-MIRROR however uses MySQL only, for reasons        |
|              of performance.                                                  |
|              WP-MIRROR has two main modes: mirror and  monitor.               |
|              The mirror maintains state, mostly about the files in its       |
|              working directory, while the monitor(s) read state to present a|
|              set of progress bars.                                            |
|              The design issue is that of concurrency---how to let mirrors    |
|              write and monitors read without data loss or resource           |
|              contention.  Semaphores or locks are used, and one may lock at  |
|              different levels:  row, page, extent, table, or database (in    |
|              order from fine to course granularity).  Course granularity     |
|              locks can cause deadlocks and writer starvation.  Not all       |
|              storage engines offer locks of fine granularity.                |
|                o bdb      - page-level                                      |
|                o ibm db2  - page-level and extent-level                     |
|                o innodb   - row-level (MVCC)                                |
|                o mssql    - row-level with escalation to page-level and     |
|                                table-level                                   |
|                o myisam   - table-level                                     |
|                o oracle   - row-level (MVCC)                                |
|                o postgres - row-level (MVCC)                                |
|                o sqlite   - table-level (actually file-level)               |
|              WP-MIRROR uses InnoDB because it offers multi-version           |
|              concurrency control (MVCC), that is, row-level locking without |
|              escalation to coarser granularity locks, together with Oracle   |
|              style consistent non-locking reads.                              |
|              Another design issue is that of scalability---how well the      |
|              storage engine performs as the number of records increases      |
|              greatly.  If the WP-MIRROR option *xchunk-page-count* is        |
|              left at its default value of 1000, then wpmirror will have      |
|              only tens of thousands of records.  If the option is set to 1,  |
|              then wpmirror will have tens of millions of records.  So it     |
|              is important that the storage engine scale well over several    |
|              orders of magnitude.                                             |
| GNU/LINUX:   The configuration files for WP-MIRROR are                        |
|                /etc/wp-mirror/default.conf and /etc/wp-mirror/local.conf      |
| NOTE:        Support for postgres could be added later given evidence of     |
|              significant demand.                                              |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+-------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1013 (assert-disk-space-p)                                      |
| ABSTRACT:    Insufficient disk space for building mirror of biggest wiki's.  |
| DESCRIPTION: For the largest wiki's (the top ten as of 2012), WP-MIRROR      |
|              will not let you start without at least 100 G free disk space.  |
| PLANNING:    Disk space requirements for mirroring the latest revisions of   |
|              en wiki are approximately (as of 2012):                         |
|                o working directory -  100G                                   |
|                o InnoDB database   -  200G                                   |
|                o images directory  -    2T                                   |
|              Given the size requirements, it would be best to put this all   |
|              on a separate disk.                                             |
| GNU/LINUX:   The images directory is usually                                 |
|                /var/lib/mediawiki/images/                                    |
|              This can (and should) be a symbolic link to a separate disk.    |
|              The working directory for WP-MIRROR should be placed under      |
|              the images directory.                                           |
| REFERENCE:   See the WP-MIRROR README for more on planning your disk space   |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1013 (assert-disk-space-if-large-wikipedia-p)                   |
| ABSTRACT:    Insufficient disk space for building mirror of biggest wiki's.  |
| DESCRIPTION: For the largest wiki's (the top ten as of 2012), WP-MIRROR      |
|              will not let you start without at least 100 G free disk space.  |
| PLANNING:    Disk space requirements for mirroring the latest revisions of   |
|              en wiki are approximately (as of 2012):                         |
|                o working directory -  100G                                   |
|                o InnoDB database   -  200G                                   |
|                o images directory  -    2T                                   |
|              Given the size requirements, it would be best to put this all   |
|              on a separate disk.                                             |
| GNU/LINUX:   The images directory is usually                                 |
|                /var/lib/mediawiki/images/                                    |
|              This can (and should) be a symbolic link to a separate disk.    |
|              The working directory for WP-MIRROR should be placed under      |
|              the images directory.                                           |
| REFERENCE:   See the WP-MIRROR README for more on planning your disk space   |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1014 (assert-hdd-write-cache-disabled-p)                        |
| ABSTRACT:    Data could lost during a power failure.                         |
| DESCRIPTION: MediaWiki stores its articles in InnoDB, which is               |
|              MySQL's ACID compliant storage engine that is used for          |
|              transactions.  The issue here regards Durability (the 'D' in    |
|              'ACID').  A committed transaction should be stored in a way     |
|              that is resistant to many kinds of failure, including power     |
|              outage.  Transactions that 'commit' must actually be written to|
|              disk, and not remain in the hard-disk-drive's write cache,      |
|              where is may be lost during system failure.  It is therefore    |
|              important that the disk's write cache be disabled.              |
| GNU/LINUX:   An hard-drive's write cache can be disabled with a command like|
|                root-shell# hdparm -W0 /dev/sda                               |
|              To avoid forgetting, this command should be made part of the    |
|              system boot process.  This can be done by appended it to        |
|              /etc/rc.local                                                   |
| REFERENCE:   See the WP-MIRROR README for more on planning your disk space  |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1015 (assert-images-directory-p)                                |
| ABSTRACT:    Unable to find mediawiki images directory.                      |
| DESCRIPTION: WP-MIRROR needs access to the images directory so that the      |
|              image files that it downloads can found by MediaWiki.           |
|              Usually this directory is '/var/lib/mediawiki/images/'.         |
|              MediaWiki is the software that will maintain and help serve     |
|              the articles that you mirror.  MediaWiki maintains a directory |
|              tree where images, thumbs (resized images), and math (formulae |
|              rendered by TeX as PNG images) are stored.                      |
| PLANNING:    Storage requirements for the images directory for mirroring     |
|              the latest revisions of en wiki are approximately (as of        |
|              2012):                                                          |
|                o image files    -  2T (two million image files)              |
|                o thumbs         - 50G (several million files)                |
|                o math           -  1G (half a million files)                 |
|              The images directory should probably be put on a separate       |
|              disk.                                                           |
| GNU/LINUX:   The images directory is usually /var/lib/mediawiki/images/.     |
|              This can (and should) be a symbolic link to a separate disk.    |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1016 (assert-images-bad-directory-or-create)                    |
| ABSTRACT:    Unable to create bad-images directory for WP-MIRROR.           |
| DESCRIPTION: Many of the downloaded images are incomplete or corrupt.  A     |
|              directory is needed to sequester these image files.             |
| GNU/LINUX:   The bad-images directory is usually                             |
|                /var/lib/mediawiki/images/bad-images/                         |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1017 (assert-internet-access-to-wikimedia-site-p)               |
| ABSTRACT:    Unable to download wikipedia articles and images.               |
| DESCRIPTION: Wikipedia articles are made available in the form of            |
|              a large compressed 'dump' file.  A million image files may also|
|              be downloaded (individually and not as a single dump).          |
| PLANNING:    Bandwidth requirements for mirroring the latest revisions of    |
|              en wiki are approximately (as of 2012):                         |
|                o dump file - 8G (one file containing millions of articles)   |
|                o images    - 2T (two million files each containing an image)|
| NOTE:        If this traffic must go through a caching proxy, there may be   |
|              problems.  Some caching proxies crash if a file exceeds its     |
|              available memory.  In this case, the dump file must be manually|
|              copied into WP-MIRRORs working directory.  Some caching         |
|              proxies might not have disk space adequate for caching 2TB of   |
|              images.  In this case, the cached images should be removed each|
|              day (perhaps by using a daily cron job).  Best of all, is to    |
|              by-pass the proxy altogether.  One good method is to use the    |
|              port-forwarding feature of ssh to connect to a box outside      |
|              the network served by the proxy.                                |
| REFERENCE:   See the WP-MIRROR README for more on planning your internet     |
|              access.                                                         |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1018 (assert-mediawiki-adminsettings-p)                         |
| ABSTRACT:    Unable to find the admin config file for MediaWiki.             |
| DESCRIPTION: WP-MIRROR creates state information about the progress of the   |
|              mirroring, and uses InnoDB to store, retrieve and share it.     |
|              The database is called wpmirror and it is accessed using       |
|              the same crediantials (host, user, and password info) that      |
|              MediaWiki uses for its own database wikidb.  Indeed,            |
|              WP-MIRROR reads through the MediaWiki configuration file to     |
|              obtain these credentials.                                       |
| GNU/LINUX:   The configuration file for MediaWiki is                         |
|                /etc/mediawiki/AdminSettings.php                              |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1019 (assert-mediawiki-extensions)                              |
| ABSTRACT:    MediaWiki extensions not installed or enabled.                  |
| DESCRIPTION: Many wikipedia articles use templates to format special content|
|              such as citations, footnotes, and poems.  These are available  |
|              in the mediawiki-extensions package.  Without these            |
|              extensions, the templates will fail to render, and you will get|
|              quite a mess consisting of scores of nested braces.             |
| GNU/LINUX:   Please install the mediawiki-extensions package, then go to     |
|                /etc/mediawiki-extensions/extensions-available/.              |
|              This directory contains a set of symbolic links.                |
|              Copy all these links to                                         |
|                /etc/mediawiki-extensions/extensions-enabled/.                |
|                root-shell# cd /etc/mediawiki-extensions/extensions-enabled/  |
|                root-shell# cp -a ../extensions-available/* .                 |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1020 (assert-mediawiki-import)                                  |
| ABSTRACT:    MediaWiki 'Import.php' patched version not installed.           |
| DESCRIPTION: MediaWiki v1.15 has a buggy version of Import.php that          |
|              causes thousands of error messages like:                        |
|                                                                              |
|              PHP Warning:  xml_parse(): Unable to call handler in_() in      |
|              /usr/share/mediawiki/includes/Import.php on line 437            |
|              PHP Warning:  xml_parse(): Unable to call handler out_() in     |
|              /usr/share/mediawiki/includes/Import.php on line 437            |
|                                                                              |
| GNU/LINUX:   WP-MIRROR should automatically download the patched version.    |
|              If it does not, then you may download it from:                  |
|                                                                              |
|              <http://svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/        |
|              includes/Import.php?view=co>                                    |
|                                                                              |
|              Then copy it to /usr/share/mediawiki/includes/.                 |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1021 (assert-mediawiki-localsettings-p)                         |
| ABSTRACT:    Unable to find the configuration file for MediaWiki.            |
| DESCRIPTION: WP-MIRROR creates state information about the progress of the   |
|              mirroring, and uses InnoDB to store, retrieve and share it.     |
|              The database is called wpmirror and it is accessed using       |
|              the same crediantials (host, user, and password info) that      |
|              MediaWiki uses for its own databases.  Indeed, WP-MIRROR        |
|              reads through the MediaWiki configuration files to obtain       |
|              these credentials.                                              |
| GNU/LINUX:   The configuration files for MediaWiki are                       |
|                /etc/mediawiki/LocalSettings.php and                          |
|                /etc/mediawiki/LocalSettings_wp-mirror.php                    |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1022 (assert-mediawiki-localsettings-image)                     |
| ABSTRACT:    MediaWiki not configured to use gm and rsvg.                    |
| DESCRIPTION: By default MediaWiki converts images (e.g. makes thumbs) by     |
|              using ImageMagick.  Unfortunately, ImageMagick grabs far        |
|              too much scratch space in main memory, and this results in poor|
|              performance and system failures.  Much better is to use         |
|              GraphicsMagick (aka gm).  Also, scalable vector graphics        |
|              SVG files are converted into PNG files.  It is beter to use     |
|              rsvg than the default.                                          |
| GNU/LINUX:   The configuration file for MediaWiki is                         |
|                /etc/mediawiki/LocalSettings.php                              |
|              Please install gm, and if                                       |
|                /etc/mediawiki/LocalSettings_wpmirror.php is not installed,   |
|              edit LocalSettings.php by appending the lines                   |
|        $wgUseImageMagick = false;                                            |
|        $wgCustomConvertCommand = '/usr/bin/gm convert %s -resize %wx%h %d'; |
|        $wgSVGConverter = 'rsvg';                                             |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1023 (assert-mediawiki-localsettings-tex)                       |
| ABSTRACT:    MediaWiki not configured to use TeX.                            |
| DESCRIPTION: Many Wikipedia articles contain mathematical formulae.  These   |
|              are best rendered using TeX.  TeX turns the math into small     |
|              PNG image files.  If TeX is not used, then the math will be     |
|              rendered rather poorly using character strings.                 |
| GNU/LINUX:   The configuration file for MediaWiki is                         |
|                /etc/mediawiki/LocalSettings.php                              |
|              If /etc/mediawiki/LocalSettings_wpmirror.php is not installed   |
|              then please install the mediawiki-math package (Debian), and    |
|              edit the MediaWiki configuration file by appending the line     |
|                $wgUseTeX = true;                                             |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1024 (assert-mediawiki-localsettings-tidy)                      |
| ABSTRACT:    MediaWiki not configured to use tidy.                           |
| DESCRIPTION: Many wikipedia articles contain complicated templates that,     |
|              without tidy, will produce badly formatted pages.  In           |
|              particular, you will find '<p>' and '<pre>' tags in the HTML    |
|              after every citation.  The resulting mess will be hard to read.|
|              Tidy is an HTML syntax checker and reformatter, and is needed   |
|              for generating readable pages.                                  |
| GNU/LINUX:   The configuration file for MediaWiki is                         |
|                /etc/mediawiki/LocalSettings.php                              |
|              Please install tidy, and if                                     |
|                /etc/mediawiki/LocalSettings_wpmirror.php is not installed,   |
|              edit LocalSettings.php by appending the line                    |
|                $wgUseTidy = true;                                            |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1025 (assert-php5-suhosin-p)                                    |
| ABSTRACT:    PHP5 not configured to use suhosin.                             |
| DESCRIPTION: MediaWiki is written in PHP, and is not entirely safe.          |
|              Suhosin, to paraphrase its documentation, is an advanced        |
|              protection module for PHP5.  It was designed to protect         |
|              servers and users from known and unknown flaws in PHP           |
|              applications and the PHP core.                                  |
|              When you run importDump.php, you can expect to find many        |
|              alert messages in the syslog, such as this:                     |
|                                                                              |
|              suhosin[21288]: ALERT - script tried to disable memory_limit by|
|              setting it to a negative value -1 bytes which is not allowed    |
|              (attacker 'REMOTE_ADDR not set', file 'unknown')                |
|                                                                              |
| GNU/LINUX:   The configuration file for php5-suhosin is                      |
|                /etc/php5/conf.d/suhosin.ini.                                 |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+-------------------------------------------------------------------------------+
| ERROR:       1026 (assert-physical-memory-p)                                  |
| ABSTRACT:    Insufficient physical memory found building biggest wiki's.      |
| DESCRIPTION: For the largest wiki's (the top ten as of 2012), WP-MIRROR       |
|              will not let you start without at least 4G main memory.          |
|              Database performance will slow to a crawl without adequate       |
|              memory.                                                          |
|              MediaWiki stores its articles in InnoDB, which is                |
|              MySQL's ACID compliant storage engine.  InnoDB organizes both    |
|              its disk space and its memory into 16K pages.  InnoDB also       |
|              keeps its records sorted.  Since most records are small,         |
|              several can fit in a page.  Accessing a record entails           |
|              accessing the right page.  In worst case, each time a record is  |
|              created, updated, or deleted, a page would have to be read from  |
|              disk, modified, and written back to disk.  Moreover, as records  |
|              are inserted, over-full pages are split up, and as records are   |
|              deleted, under-full pages are merged.  So accessing a record     |
|              could entail accessing several pages.  This could cause a        |
|              tremendous amount of disk I/O, which would hurt performance.     |
|              The answer is to keep a large number of pages in memory, so      |
|              that a page can be read in once, then quickly accessed multiple  |
|              times before writing it back to disk.  So InnoDB maintains a     |
|              large buffer pool in main memory.  When a page is read from      |
|              disk, it will be put into the buffer pool, where it will stay a  |
|              while.  InnoDB also keeps its own administrative data in the     |
|              buffer pool.  So the larger the buffer pool, the better.         |
| PLANNING:    Install at least 4G DRAM onto your mother board.  Install much   |
|              more if you can afford it.  If your mother board does not        |
|              accept that much, then replace it (for it is antiquated).        |
| NOTE:        Modern CPUs and operating systems can organize main memory into  |
|              pages of different sizes.  For example, while Intel usually      |
|              organizes memory into 4K pages, it also offers huge pages        |
|              that are 2M or 4M.  Modern operating systems will swap the       |
|              smaller pages to a swap space on disk, according to some aging   |
|              algorithm, but will leave the huge pages alone.  Given that      |
|              InnoDB has its own memory management algorithm, there is no      |
|              advantage to having the OS swap anything held in the buffer      |
|              pool---and actually swapping would hurt performance.             |
|              So it is highly recommended (though not currently required) to   |
|              allocate at least 3G of memory for huge pages, and to            |
|              configure InnoDB to use them for its buffer pool.                |
|              Leave at least 1G memory for other processes.  Importing the     |
|              dump into wikidb entails creating thumbs (resized images).       |
|              As some image files are over 100M, the resizing can easily       |
|              occupy memory several times that amount.                         |
| REFERENCE:   See the WP-MIRROR README for more on planning your memory.       |
+-------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1026 (assert-physical-memory-if-large-wikipedia-p)              |
| ABSTRACT:    Insufficient physical memory found building biggest wiki's.     |
| DESCRIPTION: For the largest wiki's (the top ten as of 2012), WP-MIRROR      |
|              will not let you start without at least 4G main memory.         |
|              Database performance will slow to a crawl without adequate      |
|              memory.                                                         |
|              MediaWiki stores its articles in InnoDB, which is               |
|              MySQL's ACID compliant storage engine.  InnoDB organizes both   |
|              its disk space and its memory into 16K pages.  InnoDB also      |
|              keeps its records sorted.  Since most records are small,        |
|              several can fit in a page.  Accessing a record entails          |
|              accessing the right page.  In worst case, each time a record is|
|              created, updated, or deleted, a page would have to be read from|
|              disk, modified, and written back to disk.  Moreover, as records|
|              are inserted, over-full pages are split up, and as records are |
|              deleted, under-full pages are merged.  So accessing a record    |
|              could entail accessing several pages.  This could cause a       |
|              tremendous amount of disk I/O, which would hurt performance.    |
|              The answer is to keep a large number of pages in memory, so     |
|              that a page can be read in once, then quickly accessed multiple|
|              times before writing it back to disk.  So InnoDB maintains a    |
|              large buffer pool in main memory.  When a page is read from     |
|              disk, it will be put into the buffer pool, where it will stay a|
|              while.  InnoDB also keeps its own administrative data in the    |
|              buffer pool.  So the larger the buffer pool, the better.        |
| PLANNING:    Install at least 4G DRAM onto your mother board.  Install much  |
|              more if you can afford it.  If your mother board does not       |
|              accept that much, then replace it (for it is antiquated).       |
| NOTE:        Modern CPUs and operating systems can organize main memory into|
|              pages of different sizes.  For example, while Intel usually     |
|              organizes memory into 4K pages, it also offers huge pages       |
|              that are 2M or 4M.  Modern operating systems will swap the      |
|              smaller pages to a swap space on disk, according to some aging  |
|              algorithm, but will leave the huge pages alone.  Given that     |
|              InnoDB has its own memory management algorithm, there is no     |
|              advantage to having the OS swap anything held in the buffer     |
|              pool---and actually swapping would hurt performance.            |
|              So it is highly recommended (though not currently required) to  |
|              allocate at least 3G of memory for huge pages, and to           |
|              configure InnoDB to use them for its buffer pool.               |
|              Leave at least 1G memory for other processes.  Importing the    |
|              dump into wikidb entails creating thumbs (resized images).      |
|              As some image files are over 100M, the resizing can easily      |
|              occupy memory several times that amount.                        |
| REFERENCE:   See the WP-MIRROR README for more on planning your memory.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1027 (assert-utilities-p)                                       |
| ABSTRACT:    Unable to find all of the software utilities required.          |
| DESCRIPTION: WP-MIRROR makes use of a couple dozen existing software         |
|              utilities.  This was due to a design decision akin to 'Do not  |
|              reinvent the wheel'.  WP-MIRROR will not let you start in       |
|              mirror mode if any of the following utilities are missing:      |
|              o bunzip2, chown, chmod, cat, cp, cron, curl, env,              |
|                gm, hdparm, md5sum, mv, mysql, mysqldump,                     |
|                openssl, php, printf, rm, rsvg, wget,                         |
|              o wikix (if you do not use the built-in default),               |
|              o /etc/mediawiki/adminsettings.php,                             |
|                /etc/mediawiki/localsettings.php,                             |
|                /etc/mediawiki/localsettings_wpmirror.php,                    |
|              o /usr/share/mediawiki/includes/Import.php,                     |
|              o /usr/share/mediawiki/maintenance/importDump_farm.php,         |
|                /usr/share/mediawiki/maintenance/rebuildImages_farm.php,      |
|                /usr/share/mediawiki/maintenance/update_farm.php,             |
|                /usr/share/mediawiki/maintenance/importDump.php,              |
|                /usr/share/mediawiki/maintenance/rebuildImages.php,           |
|                /usr/share/mediawiki/maintenance/update.php.                  |
| GNU/LINUX:   A package management system (e.g. Debian) can provide easy      |
|              installation and maintenance of a consistent set of utilities.  |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1028 (assert-virtual-host-p)                                    |
| ABSTRACT:    Virtual host wpmirror.site not found.                           |
| DESCRIPTION: WP-MIRROR sets up a virtual host to let you access your new     |
|              mirrors.  For example, you access your mirror of the simple     |
|              wiki by giving your browser http://simple.wpmirror.site/.       |
|              The virtual host container should be found in                   |
|                /etc/apache2/sites-available/wpmirror.site.conf.              |
|              There should also be a symbol link to it found in               |
|                /etc/apache2/sites-enabled/wpmirror.site.conf.                |
|              And the virtual host configuration should be loaded into        |
|              Apache2.                                                        |
|                                                                              |
|              root-shell# apache2ctl -S 2>&1 | grep mediawiki                 |
|                       port 80 namevhost wpmirror.site (/etc/apache2/sites-en|
|              abled/wpmirror.site.conf:1)                                     |
|                                                                              |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1029 (assert-virtual-host-name-resolution-p)                    |
| ABSTRACT:    Virtual host wpmirror.site not found in /etc/hosts.             |
| DESCRIPTION: Apache2 needs to resolve the name wpmirror.site into the IP     |
|              address ::1 (or 127.0.0.1 if you prefer IPv4).  This can be     |
|              done with a DNS server, such as bind, but that is atypical for  |
|              a PC.  Name resolution is most easily done by appending lines   |
|              to /etc/hosts like these:                                       |
|                                                                              |
|              ::1 locahost wpmirror.site www.wpmirror.site                    |
|              ::1 meta.wpmirror.site simple.wpmirror.site                     |
|              ::1 en.wpmirror.site zh.wpmirror.site                           |
|                                                                              |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1030 (assert-working-directory-or-create)                       |
| ABSTRACT:    Unable to create working directory for WP-MIRROR.               |
| DESCRIPTION: WP-MIRROR uses a directory to generate and maintain tens of     |
|              thousands of files used by the mirroring process.               |
| PLANNING:    Storage requirements for the working directory for mirroring    |
|              the latest revisions of en wiki are approximately (as of        |
|              2012):                                                          |
|                o dump file              -  8G                                |
|                o decompressed dump file - 37G                                |
|                o xchunks                - 37G                                |
|                o ichunks                -  5G                                |
| GNU/LINUX:   The working directory is usually                                |
|                /var/lib/mediawiki/images/wp-mirror/                          |
| REFERENCE:   See the WP-MIRROR README for more on planning your system.      |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
| ERROR:       1031 (warn-if-detect-proxy)                                     |
| ABSTRACT:    Web proxy detected.  Manually check if it can handle traffic.   |
| DESCRIPTION: Caching web proxies offer many benefits: economize band-width,  |
|              reduce latency, improve security, log web traffic, block        |
|              inappropriate sites, and bridge between IPv6 and IPv4 networks. |
|              They also have weaknesses: unable to handle downloading very    |
|              large files, and need large disk space for cache.               |
| PLANNING:    1) Dump file.  Dump files containing the latest revisions of    |
|              en wiki are 8G (as of 2012).  Some caching web proxies crash    |
|              when downloading a file larger than their available memory.     |
|              You may need to obtain the dump file some other way, then       |
|              manually copy it to WP-MIRRORs working directory.               |
|              2) Images.  en wiki references about two million image files,   |
|              which collectively occupy about 2T (as of 2012).  Downloading  |
|              the image files may stuff your proxy's cache.  This can be      |
|              managed by deleting image files from the cache on a daily basis |
|              (using a cron job say).                                         |
| GNU/LINUX:   The WP-MIRROR working directory is usually                      |
|                /var/lib/mediawiki/images/wp-mirror/                          |
| REFERENCE:   See the WP-MIRROR README for more on caching web proxies.       |
+------------------------------------------------------------------------------+
```

Appendix E

Experiments (Autumn 2010—Spring 2011)

This is a transcript of my original notes, edited, and formatted for LaTeX. In the notes below, the terms darkstar-4, darkstar-5, and darkstar-7, are host names.

Source: /root/sysadmin/2010_09_sysadmin.log.

Date: 2010-Sept-12 through 2011-Apr

Re: How to build a mirror of the http://en.wikipedia.org/.

E.1 Introduction

Purpose. Mirror the latest revision of http://en.wikipedia.org/. The web pages, when loaded into MySQL might occupy about 0.2T. The images might also occupy an additional 1.0T. We do not want to mirror all its revisions, user talk, etc. as the entire en.wikipedia.org site might require over 10T. Likewise we do not want to mirror all languages as this might exceed 100T.

HDD. By default MySQL stores its tables under /var/lib/mysql/. Since darkstar-5's /var/ partition is currently 100GB (and already holds about 50GB due to mirroring of part of the Debian archive), we need to store the database tables elsewhere. We shall settle upon a new HDD dedicated to storing the MySQL tables. Currently we need a bit over 1T. So to allow for future growth we should purchase a 1.5T or 2T HDD.

Cooling. The existing two HDDs are running a little hot (50-55C, design limit is 60C). Addition of a third HDD will make it worse. We will need a fan with higher air flow.

Structure. The HDD shall have two LVM2 partitions: one for the database, one for the images:

ibdata0: InnoDB on LVM2 on LUKS on RAID on whole disks

images0: ReiserFS on LVM2 on LUKS on RAID on whole disks

We shall use the InnoDB storage engine. InnoDB supports transactions and is the ACID compliant storage engine for MySQL. The table space for InnoDB can be stored either as a file on a file system, or directly on a raw disk or partition. There is some speed advantage to the later. This is because InnoDB is, in effect, a journaling file system; and, therefore, there is no advantage to storing the InnoDB table space on a journaling file system such as ReiserFS.

The image files for the wikipedia are, however, stored in a file system, rather than in a database. This is a design decision by the Wikimedia Foundation that recognizes that web servers are better at caching files than database contents.

Here are the storage requirements:

enwiki-yyyymmdd-		2010 09-16	2011 01-15	03-21	10-07
pages-articles.xml.bz2	a	6.2G	6.5	6.7	7.8
pages-articles.xml	a	28	29	30	34
wikidb in InnoDB	b	50*	54**	1xx	
images/	c	690	745	780	
pages-articles (rows)	d	10.3m	10.8	11.1	11.7
<title> is "File:..."	e			0.87	0.85
image count (*)	f	1.24	1.46	1.49	
images - download.log+	g			34.5k	
from exists.log	g			1.65m	
wikix failed.log	g			287k	

Notes:

a) `ls -lh <file>`
b) see InnoDB diskspace computation (next Note below)
c) `df -h | grep database0` (includes [0-9a-f], math, thumb)
d) `cat <file> | grep "<page>" | wc -l`
e) `cat <file> | grep "<title>" | grep "File:" | wc -l`
f) mysql> `SELECT COUNT(*) FROM image;`
g) `wc -l *log` (duplicates seen in exists.log and failed.log)
∗) imported 6.5m/10.3m pages before fiasco with pagelinks indices.
∗∗) xml parse errors prevented importation of 3 million pages.
+) 2011-03-21 images downloaded: 18k new, 16k previously failed.

Note: the diskspace occupied by InnoDB data is computed as follows:

1. vg5 has two logical volumes

```
root-shell# lvdisplay -v vg5 | grep LE
Current LE       75000       <--- ibdata0 (75000*4MB=300000MB)
Current LE      401931       <--- images0
```

   each logical extent (LE) is 4MB in size
2. the total space available to InnoDB is 300000Mraw = 314572800000 bytes

```
root-shell# cat /etc/mysql/conf.d/custom.cnf | grep Mraw
#innodb_data_file_path           = /dev/mapper/vg5-ibdata0:300000Mnewraw
innodb_data_file_path           = /dev/mapper/vg5-ibdata0:300000Mraw
```

3. after inserting records we can see how much space InnoDB has left for example, after importation of 2011-01-15 dump, we get

```
mysql> SHOW TABLE STATUS LIKE 'page'\G
Data_free: 256567672832
```

4. finally, we take the difference and reduce to GB (314572800000-256567672832)/(1024*1024*1024) = 54G

Note: failed.log contains a very great number of duplicates: any file name containing blanks, will be attempted with underscores as

```
./1/10/River South New Caledonia.JPG failed
./2/20/River_South_New_Caledonia.JPG failed
```

DRAM. We eventually learned the hard way that 2GB DRAM is not enough. The system freezes for a minute or two to perform a lot of swapping when /usr/bin/convert runs to resize images (make thumbs). We add 4GB for a total of 6GB, choosing DDR2 800 to match existing DIMMs. Later on we learned that convert (ImageMagick) uses an unreasonable amount of scratch space, and that it is much better to use /usr/bin/gm convert (GraphicsMagick).

E.2 Hardware

E.2.1 PARTS

Qty	Description
1	Seagate ST320005N4A1AS-RK (2.0T SATA HDD) S/N 5XW0SCXV
1	Antec TriCool 120mm (3-speed case fan, set on high)
1	OCZ OCZ2G8004GK (2x2GB DDR2 DIMM 800 5-5-5 1.8V)

Install packages:

```
root-shell# aptitude install hddtemp mdadm cryptsetup lvm2 reiserfsprogs
root-shell# aptitude install mysql-server mysql-client mysqltuner
root-shell# aptitude install atsar
root-shell# aptitude install mediawiki mediawiki-extensions mediawiki-math tidy
root-shell# aptitude install php5-suhosin      <-- if not already installed with php5
root-shell# aptitude install libssl-dev build-essential curl
root-shell# aptitude install graphicsmagick librsvg2-bin librsvg-common
```

E.2.2 H/W Test

Check that the new fan is cooling the HDDs (allow 20 min to settle):

```
root-shell# aptitude install hddtemp
root-shell# hddtemp /dev/sd[a-z]
/dev/sda: ST31500341AS: 42<B0>C
/dev/sdb: ST31500341AS: 37<B0>C
/dev/sdc: ST32000542AS: 36<B0>C
```

Yes!
Check new disk for bad blocks using:

```
root-shell# badblocks -c 102400 -o /tmp/badblocks.txt -s -t random -v -w /dev/sdc
Checking for bad blocks in read-write mode
From block 0 to 1953514583
Testing with random pattern: done
Reading and comparing: done
Pass completed, 0 bad blocks found.
```

This took about fourteen hours (seven to write, seven to verify).

E.2.3 Partitions

We shall use whole disks, no `cfdisk` partitions.

E.2.4 RAID

We use `raid1` as the next layer just in case we later wish to mirror it with a second disk.

```
root-shell# aptitude install mdadm
root-shell# mdadm --create /dev/md4 --verbose \
--level=1 --raid-devices=2 --metadata=1.0 --bitmap=internal \
--name=database0 --auto=md /dev/sdc missing
```

where

- **level=1** means `raid1` (mirrored),

- **raid-devices=2** means raid set will have two active disks,
- **metadata=1.0** means version-1 format superblock located at the end of each disk (if array will be larger than 2T, then default=0.90 will not do),
- **bitmap=internal** means a write-intent log is stored near the superblock (resync greatly optimized; full resync takes over 25 hours if no bitmap),
- **name=database0** means the RAID array will have a name (possible with version-1 format superblock)
- **auto=md** means a non-partitionable array, and
- **/dev/sdc** and **missing** are the disks.

mdadm responded with the following message:

```
mdadm: size set to 1953514448K
mdadm: array /dev/md4 started.
```

Let us collect some data:

```
shell$ cat /proc/mdstat
md4 : active (auto-read-only) raid1 sdc[0]
      1953514448 blocks super 1.0 [2/1] [U_]
      bitmap: 0/466 pages [0KB], 2048KB chunk
```

```
root-shell# mdadm --detail /dev/md4
/dev/md4:
        Version : 01.00
  Creation Time : Sun Sep 12 23:10:05 2010
     Raid Level : raid1
     Array Size : 1953514448 (1863.02 GiB 2000.40 GB)
  Used Dev Size : 3907028896 (3726.03 GiB 4000.80 GB)
   Raid Devices : 2
  Total Devices : 1
Preferred Minor : 4
    Persistence : Superblock is persistent

  Intent Bitmap : Internal

    Update Time : Sun Sep 12 23:10:05 2010
          State : active, degraded
 Active Devices : 1
Working Devices : 1
 Failed Devices : 0
  Spare Devices : 0

           Name : darkstar-5:database0  (local to host darkstar-5)
           UUID : 7a801ff2:bd55f4c9:83164cba:aa29323b
         Events : 0

    Number   Major   Minor   RaidDevice State
       0       8       32        0      active sync   /dev/sdc
       1       0        0        1      removed
```

```
root-shell# mdadm --detail --scan
ARRAY /dev/md2 level=raid1 num-devices=2 metadata=01.00 name=darkstar-5:cryptroot
 UUID=a7cc74e7:0a5b34fe:0e2ce5f0:05237600
ARRAY /dev/md3 level=raid1 num-devices=2 metadata=01.00 name=darkstar-5:archive1
 UUID=09c2f411:76ed3beb:8e526ddc:e370eb70
ARRAY /dev/md4 level=raid1 num-devices=2 metadata=01.00 name=darkstar-5:database0
 UUID=7a801ff2:bd55f4c9:83164cba:aa29323b
```

```
root-shell# mdadm --detail --scan >> /etc/mdadm/mdadm.conf
```

WARNING: There is a bug in mdadm. You must edit mdadm.conf and replace "metadata=01.00" with "metadata=1.0", like so:

```
root-shell# mdadm --detail --scan
ARRAY /dev/md2 level=raid1 num-devices=2 metadata=1.0 name=darkstar-5:cryptroot
 UUID=a7cc74e7:0a5b34fe:0e2ce5f0:05237600
ARRAY /dev/md3 level=raid1 num-devices=2 metadata=1.0 name=darkstar-5:archive1
 UUID=09c2f411:76ed3beb:8e526ddc:e370eb70
ARRAY /dev/md4 level=raid1 num-devices=2 metadata=1.0 name=darkstar-5:database0
 UUID=7a801ff2:bd55f4c9:83164cba:aa29323b
```

Test /etc/mdadm/mdadm.conf:

```
root-shell# mdadm --stop /dev/md4
mdadm: stopped /dev/md4
root-shell# mdadm --assemble --scan
mdadm: no devices found for /dev/md0
mdadm: no devices found for /dev/md1
mdadm: /dev/md4 has been started with 1 drive (out of 2).
```

E.2.5 (Optionally) Add Second Disk To Raid Array

We created /dev/md4 with only one disk. One can later add a second.

```
root-shell# mdadm /dev/md4 --add /dev/sdd
root-shell# cat /proc/mdstat
md4 : active raid1 sdd[2] sdc[0]
      1953514448 blocks super 1.0 [2/1] [U_]
      [>....................]  recovery =  3.2% (48038208/1465039472)
 finish=245.5min speed=96166K/sec
      bitmap: 3/466 pages [1864KB], 2048KB chunk
```

Resync should take about 40 hours for a 2T array.

E.2.6 LUKS

Encryption is unnecessary for wikipedia. However, the MySQL storage engine InnoDB stores all tables in a single "table space". This means that other database backed services (e.g. drupal, sendmail, etc.) will be stored on the same disk. This would be a security hole.

Install packages:

```
root-shell# aptitude install cryptsetup
```

Policy. Encrypt all storage.

Create LUKS partition:

```
root-shell# cryptsetup --cipher aes-xts-plain --key-size 512 luksFormat /dev/md4
WARNING!
========
This will overwrite data on /dev/md4 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase: pass-phrase
Verify passphrase: pass-phrase
Command successful.
```

Where *pass-phrase* should be replaced with a secret pass-phrase of your own.
Open the LUKS partition with the pass phrase entered above:

```
root-shell# cryptsetup luksOpen /dev/md4 xts_database0
Enter LUKS passphrase:
key slot 0 unlocked.
Command successful.
```

```
root-shell# ls /dev/mapper
control    vg0-home  vg0-swap  vg0-usr  vg4-archive1  xts_database0
cryptroot  vg0-root  vg0-tmp   vg0-var  xts_archive1
```

Add a second key to the LUKS partition (optionally):

```
root-shell# cryptsetup luksAddKey /dev/md4 /etc/cryptpass
Enter any LUKS passphrase:
key slot 0 unlocked.
Command successful.
```

Examine the LUKS partition header:

```
root-shell# cryptsetup luksDump /dev/md4
LUKS header information for /dev/md4

Version:        1
Cipher name:    aes
Cipher mode:    xts-plain
Hash spec:      sha1
Payload offset: 4040
MK bits:        512
MK digest:      e5 29 4e 23 e3 93 d8 7f 07 0d a6 85 cb 07 a9 2f 81 1b 60 eb
MK salt:        b0 85 90 3f f7 25 75 ce 03 f5 9e c6 ad c0 f5 ae
                f2 66 9f 98 68 97 fc 11 fc f5 b7 cd 92 a6 d6 a2
MK iterations:  10
UUID:           3348fa04-4abf-42e8-906b-1bc9dba9b580

Key Slot 0: ENABLED
        Iterations:             359864
        Salt:                   b9 84 59 4c ed 89 14 fb 38 63 fc 9d b5 a9 96 a0
                                a0 06 73 ce 59 22 cb dc bf 12 05 9c 83 37 03 cd
        Key material offset:    8
        AF stripes:             4000
Key Slot 1: ENABLED
        Iterations:             364323
        Salt:                   93 6c e0 9b 99 52 48 0f e9 bf 3f ca a9 e5 11 40
                                93 7c b1 bc 2f 01 40 cb 71 2a 68 b1 af 75 19 dc
        Key material offset:    512
        AF stripes:             4000
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
```

To unlock this partition during boot, there are two options: interactive and automatic:
a) For interactive, give "none" as the key-file:
Edit /etc/crypttab (for /etc/init.d/cryptdisks-early):

```
root-shell# cat /etc/crypttab
#<target dev> <source dev> <key file> <options>
xts_database0 /dev/md4     none       luks,tries=3
```

b) For automatic, supply a key file:

```
root-shell# emacs /etc/keys/luks_key_md4
My S3cr3t P@ssphr@s3
```

Deny non-root access by setting file permissions:

```
root-shell# chmod 600 /etc/keys/luks_key_md4
```

Edit /etc/crypttab (for cryptdisks-early):

```
#<target dev> <source dev> <key file>             <options>
xts_database0 /dev/md4     /etc/keys/luks_key_md4 luks,tries=3
```

Reboot to test. Yes!

E.2.7 LVM2

Install packages:

```
root-shell# aptitude install lvm2
```

Create physical volume:

```
root-shell# pvcreate /dev/mapper/xts_database0
  Physical volume "/dev/mapper/xts_database0" successfully created
```

Create volume group:

```
root-shell# vgcreate vg5 /dev/mapper/xts_database0
  Volume group "vg5" successfully created
```

```
root-shell# vgdisplay vg5
  --- Volume group ---
  VG Name               vg5
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               1.82 TB
  PE Size               4.00 MB
  Total PE              476931
  Alloc PE / Size       0 / 0
  Free  PE / Size       476931 / 1.82 TB
  VG UUID               XiXr0n-BsIq-SugA-ohBy-k0o1-tXUl-t5wCqd
```

Create logical volumes:

```
root-shell# lvcreate --extents 75000 --name ibdata0 vg5
  Logical volume "ibdata0" created
root-shell# lvcreate --extents 401931 --name images0 vg5
  Logical volume "images0" created
```

```
root-shell# ls /dev/mapper/
control    vg3-home  vg3-swap  vg3-usr  vg5-ibdata0  xts_database0
cryptroot  vg3-root  vg3-tmp   vg3-var  vg5-images0
```

```
root-shell# vgdisplay vg5
  --- Volume group ---
  VG Name               vg5
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  9
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                2
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               1.82 TB
  PE Size               4.00 MB
  Total PE              476931
  Alloc PE / Size       476931 / 1.82 TB
  Free  PE / Size       0 / 0
  VG UUID               XiXr0n-BsIq-SugA-ohBy-k0o1-tXUl-t5wCqd
```

```
root-shell# lvdisplay vg5
  --- Logical volume ---
  LV Name                /dev/vg5/ibdata0
  VG Name                vg5
  LV UUID                jQSCdj-xImA-wZIG-bY0P-M4Ck-v0Ep-SXppHW
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                292.97 GB
  Current LE             75000
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:10

  --- Logical volume ---
  LV Name                /dev/vg5/images0
  VG Name                vg5
  LV UUID                f38Moc-NSEd-oBkN-AIxe-qdOX-l9VY-h88sMT
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                1.53 TB
  Current LE             401931
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:11
```

E.2.8 ReiserFS

Install packages:

```
root-shell# aptitude install reiserfsprogs
```

Create the file system:

```
root-shell# mkfs.reiserfs /dev/vg5/images0
mkfs.reiserfs 3.6.19 (2003 www.namesys.com)
...
Guessing about desired format.. Kernel 2.6.26-2-amd64 is running.
Format 3.6 with standard journal
Count of blocks on the device: 411577344
Number of blocks consumed by mkreiserfs formatting process: 20772
Blocksize: 4096
Hash function used to sort names: "r5"
Journal Size 8193 blocks (first block 18)
Journal Max transaction length 1024
inode generation number: 0
UUID: 3c9eb1d3-4b82-4580-86f2-a4516b240f54
ATTENTION: YOU SHOULD REBOOT AFTER FDISK!
        ALL DATA WILL BE LOST ON '/dev/vg5/images0'!
Continue (y/n):
Initializing journal - 0%....20%....40%....60%....80%....100%
Syncing..ok
...
ReiserFS is successfully created on /dev/vg5/images0.
```

```
root-shell# mkdir /database0
root-shell# mkdir /database0/images0
```

Add one line to /etc/fstab:

```
/dev/vg5/images0  /database0/images  reiserfs  defaults  0  2
```

and mount the file system:

```
root-shell# mount /database0/images
root-shell# chown www-data:www-data /database0/images
root-shell# ls -l /database0/ | grep images0
drwxr-xr-x 4 www-data www-data  80 2010-09-13 05:21 images0
```

E.3 Configuring MySQL

Install packages:

```
root-shell# aptitude install hdparm mysql-server mysql-client mysqltuner
```

HDD. Turn off write-caching for the disk that holds the InnoDB table space (or raw partitions). This is to make transactions durable (the 'D' in ACID). To make this happen during system boot, edit /etc/hdparm.conf to read:

```
root-shell# cat /etc/hdparm.conf
/dev/sdc {
        write_cache = off
}
```

and reboot. Actually, although hdparm.conf ran during boot but I still found the HDD with write caching on. So I edited /etc/rc.local to read:

```
root-shell# cat /etc/rc.local
hdparm -W0 /dev/sdc
```

then rebooted. This got the job done.

Password. Set passwords (initially users root and test have no password):

```
shell$ mysql -u root mysql
mysql> SET PASSWORD = PASSWORD('root_password');
mysql> SET PASSWORD FOR 'test'@'localhost' = PASSWORD('test_password');
mysql> FLUSH PRIVILEGES;
mysql> QUIT;
```

Time Zone. Load time zone information (so we can set default to UTC).

```
shell$ mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root -p mysql
```

Edit /etc/mysql/conf.d/custom.cnf (to override default values found in /etc/mysql/my.cnf). This we shall refer to this as our baseline configuration.

```
shell$ cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone      = UTC
character-set-server   = utf8
collation-server       = utf8_general_ci
default-storage-engine = innodb
# Enable InnoDB Plugin instead of the Built-in InnoDB (features)
ignore-builtin-innodb
plugin-load=innodb=ha_innodb_plugin.so;innodb_trx=ha_innodb_plugin.so;
innodb_locks=ha_innodb_plugin.so;innodb_lock_waits=ha_innodb_plugin.so;
innodb_cmp=ha_innodb_plugin.so;innodb_cmp_reset=ha_innodb_plugin.so;
innodb_cmpmem=ha_innodb_plugin.so;innodb_cmpmem_reset=ha_innodb_plugin.so
plugin_dir=/usr/lib/mysql/plugin
[mysql]
default-character-set  = utf8
default-collation      = utf8_general_ci
```

Note: The value of plugin-load must be typed all on one line with no spaces.

Run benchmarks:

```
root-shell# cd /usr/share/mysql/sql-bench
root-shell# perl run-all-tests --user=test --password=test_password --log
root-shell# cd output
```

Drop test user and database for security:

```
shell$ mysql -u root -p mysql
mysql> DROP USER 'test'@'localhost';
mysql> DROP DATABASE test;
mysql> QUIT;
```

E.4 Configuring hugepages

Distribution: Debian lenny

We allocated 2GB hugepages to MySQL.

Reason: We have 6GB DRAM

- need 2GB for convert (which makes thumbs from images),
- need 2GB for Xorg, KDE 3.5, and all else,

- leaving 2GB to be dedicated to MySQL.

Note: PHP seems to have a memory leak, gradually eating up memory. It is best to allow 1GB for this, and to kill jobs (e.g. rebuildImages.php, importDump.php) with Ctrl-C every 50,000 records.

MAJOR UPGRADE.
Distribution: Debian squeeze
We allocate 3GB hugepages to MySQL.
Reason: We have 6GB DRAM

- we now use gm convert which uses much less scratch space,
- need 3GB for Xorg, KDE 4.4.5 (plama-desktop), and all else,
- leaving 3GB to be dedicated to MySQL.

Note: convert (ImageMagick) (default) uses too much scratch space (>2 GB).
References:
http://www.python.com/news/1326/performance-tuning-hugepages-in-linux/
http://wiki.debian.org/Hugepages
http://www.ibm.com/developerworks/linux/library/l-mem26/
http://dev.mysql.com/doc/refman/5.1/en/large-page-support.html
http://developer.postgresql.org/pgdocs/postgres/kernel-resources.html
InnoDB has its own memory manager. InnoDB pages are 16KB (rather than Intel's 4KB), and InnoDB has its own algorithm for moving pages to and from disk. The Linux swap algorithm would interfere (e.g by moving a 4KB page to the swap partition, when InnoDB needs its 16KB page to stay together in DRAM).

Intel (like most CPU manufacturers) offers hugepages (aka superpages, largepages) of size 2MB or 4MB. Hugepages stay in DRAM and are not touched by the Linux swap algorithm. There is a system control parameter for reserving hugepages, and InnoDB will use hugepages if they have been so reserved.

If you notice your system periodically slowing to a crawl, you should run diagnostics with the System Activity Report:

```
root-shell# aptitude install atsar
```

Atsar runs every 10 minutes as a cron job as /etc/cron.d/atsar, and each day's reports are stored in /var/log/atsar/atsaXX (where XX is the day of the month).

```
shell$ atsar -p         <-- paging
shell$ atsar -r         <-- memory
shell$ atsar -P         <-- process load
shell$ atsar -u         <-- CPU utilization
shell$ atsar -D         <-- Disk activity
```

In my case there were sudden fits of page swapping, and the page tables occupied a lot of RAM.

```
shell$ cat /proc/meminfo | grep PageTables
PageTables:      28496 kB
```

Hugepages reduce number of Translation Lookaside Buffer (TLB) misses. Hugepages are 2MB instead of 4KB, and they are locked in RAM and cannot be swapped out. Best yet, MySQL InnoDB can and will use them if available.

Check if Linux (> 2.6.23) supports it:

```
shell$ cat /proc/meminfo | grep Huge
HugePages_Total:     0
HugePages_Free:      0
HugePages_Rsvd:      0
HugePages_Surp:      0
Hugepagesize:     2048 kB
```

Create a group for users of hugepages:

```
root-shell# groupadd hugepage
root-shell# getent group hugepage
hugepage:x:1001:
root-shell# adduser mysql hugepage
Adding user ‘mysql’ to group ‘hugepage’ ...
Adding user mysql to group hugepage
Done.
root-shell# getent group hugepage
hugepage:x:1001:mysql
```

To specify number of hugepages to be reserved, edit /etc/sysctl.conf, append lines:

```
# Hugepages can reduce Translation Lookaside Buffer (TLB) misses.
#
# Let us allocate 3GB of hugepages to mysql.
#
# Allocate 2MiB hugepages. (default: 0)
#  512 2MB hugepages = 1024MB = 1GB
# 1024 2MB hugepages = 2048MB = 2GB
# 1536 2MB hugepages = 3072MB = 3GB
# 2048 2MB hugepages = 4096MB = 4GB
vm.nr_hugepages = 1536
# Add the gid of the group hugepage(1001) to give access to its users
vm.hugetlb_shm_group = 1001
# Maximum shared memory segment size (default: 33554432 bytes = 32MB)
# 1073741824 bytes = 1024MB = 1GB
# 2147483648 bytes = 2048MB = 2GB
# 3221225472 bytes = 3072MB = 3GB
# 4294967296 bytes = 4096MB = 4GB
kernel.shmmax = 3221225472
# Total amount of shared memory (default: 2097152 4KB pages = 8GB)
#  262144 4KB pages = 1024MB = 1GB
#  524288 4KB pages = 2048MB = 2GB
#  786432 4KB pages = 3072MB = 3GB
# 1048576 4KB pages = 4096MB = 4GB
kernel.shmall = 786432
```

Create a mount point for file system:

```
root-shell# mkdir /hugepages
```

Edit /etc/fstab (mode 1770 allows every user in group to create files but not unlink or rename each other's files):

```
hugetlbfs  /hugepages  hugetlbfs mode=1770,gid=1001 0 0
```

Reboot. a) It is an easy way to allocate hugepages before memory gets fragmented. b) Instead of rebooting, I tried:

```
root-shell# sysctl -p
```

and got no hugepages allocated. c) So I exited several large processes, tried again, and got only 8 hugepages. d) So I exit the KDM session, did console login as root, ran:

```
root-shell# sysctl -p
```

and got all of them.

```
root-shell# cat /proc/meminfo | grep Huge
HugePages_Total:  1536
HugePages_Free:   1536
HugePages_Rsvd:      0
HugePages_Surp:      0
Hugepagesize:     2048 kB
```

Allow users in the hugepage group to lock up to 3GB. Edit /etc/security/limits.conf, append the line:

```
#<domain>       <type>  <item>      <value>
@hugepage       hard    memlock     1024000
@hugepage       soft    memlock     1024000
mysql           -       memlock     1024000
```

To enable hugepage support in MySQL, edit /etc/mysql/conf.d/custom.cnf by appending:

```
[mysqld]
# Enable large page support. InnoDB will use it automatically for its
# buffer_pool and its additional_memory_pool.
large-pages
```

```
root-shell# /etc/init.d/mysql restart
Stopping MySQL database server: mysqld.
Starting MySQL database server: mysqld.
Checking for corrupt, not cleanly closed and upgrade needing tables..
```

```
shell$ mysql -u root -p
mysql> SHOW VARIABLES LIKE 'large_page%';
+---------------------------+---------+
| Variable_name             | Value   |
+---------------------------+---------+
| large_page_size           | 2097152 |
| large_pages               | ON      |
+---------------------------+---------+
2 rows in set (0.00 sec)
```

Debug: If you get the following message in /var/log/syslog:

```
# cat /var/log/syslog | grep InnoDB
Sep 14 23:30:45 darkstar-5 mysqld[4427]: InnoDB: HugeTLB: Warning: Failed to
 allocate 33570816 bytes. errno 12
Sep 14 23:30:45 darkstar-5 mysqld[4427]: InnoDB HugeTLB: Warning: Using
 conventional memory pool
```

this means that either a) the hugepage space is too small for the MySQL RAM footprint, in which case, reduce the innodb_buffer_pool_size a bit (from 3000M down to 2998M):

```
shell$ cat /etc/mysql/conf.d/custom.cnf | grep pool
innodb_buffer_pool_size         =2998M  # default  8M.
innodb_additional_mem_pool_size =   8M  # default  8M (plugin), 1M (built-in).
```

```
shell$ mysql -u root -p
mysql> SHOW VARIABLES LIKE '%pool%';
+---------------------------------+------------+
| Variable_name                   | Value      |
+---------------------------------+------------+
| innodb_additional_mem_pool_size | 8388608    |
| innodb_buffer_pool_size         | 3143630848 |
+---------------------------------+------------+
2 rows in set (0.00 sec)
```

or b) your settings in /etc/security/limits.conf are being ignored. To be sure that the memlock limits are being read, go to /etc/pam.d/ and make sure that:

```
session required  pam_limits.so
```

for PAM is being used. That is, ensure that the above line is included in the following files and is not commented out:

/etc/pam.d/su
/etc/pam.d/login
/etc/pam.d/other
/etc/pam.d/sshd

This did NOT work for me. So I added the following code to /usr/bin/mysqld_safe right below the other calls to ulimit.

```
# <edit>
# Here I am trying to use "large-pages" (linux "hugepage")
ulimit -l unlimited
# </edit>
```

This did the trick.

E.5 Configuring MediaWiki

Install packages:

```
root-shell# aptitude install mediawiki mediawiki-extensions mediawiki-math tidy
root-shell# aptitude install php5-suhosin      <-- if not already installed with php5
root-shell# aptitude install graphicsmagick librsvg2-bin librsvg-common
root-shell# aptitude install graphicsmagick-imagemagick-compat <-- maybe not wise
```

Note: php5-suhosin (PHP-hardening project) is recommended because MediaWiki uses PHP code, and some of it has security risks. Every time I run importDump.php, I see the following alert in my logs.

```
Feb 15 12:04:24 darkstar-5 suhosin[20029]: ALERT - script tried to disable
memory_limit by setting it to a negative value -1 bytes which is not allowed
(attacker 'REMOTE_ADDR not set', file 'unknown')
```

MediaWiki tries to disable memory limits requested by the user.

Now let us setup up MediaWiki.

Edit /etc/mediawiki/apache.conf to set an useful alias by uncommenting the following line, then restart the web server.

```
root-shell# cat /etc/mediawiki/apache.conf | grep Alias
Alias /mediawiki /var/lib/mediawiki
```

```
root-shell# /etc/init.d/apache2 restart
```

Open browser to http://localhost/mediawiki/config/index.php, and fill out the form.

```
Site config

  Wiki name:           Wikipedia
  Contact e-mail:      webmaster@localhost
  Language:            en-English
  Copyright/license:   * No license metadata
  Admin username:      WikiSysop
  Password:            ********
  Password confirm:    ********
  Object caching:      (*) No caching
  Memcached servers:   <blank>

E-mail, e-mail notification and authentication setup

  E-mail features:                    (*) Disabled
  User-to-user e-mail:                (*) Disabled
  E-mail notification about changes:  (*) Disabled
  E-mail address authentication:      (*) Disabled

Database config

  Database type:        (*) MySQL
  Database host:        localhost
  Database name:        wikidb
  DB username:          wikiuser
  DB password:          ********
  DB password confirm:  ********
  Superuser account:    [x] Use superuser account
  Superuser name:       root
  Superuser password:   ********

MySQL specific options

  Database table prefix:    <blank>
  Storage Engine:           InnoDB
  Database character set:   MySQL 4.1/5.0 binary
```

Press "Install MediaWiki!"

Move the settings, edit as follows, and apply file protections:

```
root-shell# cd /etc/mediawiki
root-shell# mv /var/lib/mediawiki/config/LocalSettings.php .
root-shell# emacs LocalSettings.php
$wgShowExceptionDetails = true;

$wgDBadminuser      = "root";          <-- set this to run maintenance scripts
$wgDBadminpassword  = "root_password";     such as rebuildImages

$wgDBTableOptions   = "ENGINE=InnoDB, DEFAULT CHARSET=binary";
$wgDBmysql5         = true;            <-- charset for MySQL 4.1/5.0

$wgEnableUploads    = true;            <-- set these to make image thumbs
$wgUseImageMagick   = false;           <-- over-commits VIRT and RES memory
$wgCustomConvertCommand = "/usr/bin/gm convert %s -resize %wx%h %d";
$wgSVGConverter     = 'rsvg';          <-- converts SVG to PNG

$wgUseTeX           = true;            <-- set this to use mediawiki-math
$wgUseTidy          = true;            <-- set this to clean up citations

root-shell# chown www-data:www-data LocalSettings.php
root-shell# chmod 600 LocalSettings.php
root-shell# rm LocalSettings.php~
```

Next, enable the MediaWiki extensions. If you forget this, then all the templates "..." will fail to render, and you will get quite a mess.

```
root-shell# cd /etc/mediawiki-extensions/extensions-enabled
root-shell# cp -a ../extensions-available/* .
```

Note, if you forget to use tidy, then the HTML will contain a paragraph tag "<p>" and a "<pre>" tag after every citation, and the resulting layout will detract from the reading experience.

Note, graphicsmagick-imagemagick-compat purges imagemagick and replaces /usr/bin/convert with a link to gm. Supposedly, this requires no changes to other code (e.g. the MediaWiki maintenance scripts). If it worked, it would be useful because some of the MediaWiki maintenance scripts call convert (to rasterize SVG into PNG). However, when I tried it, I found that convert either hung or caused a segmentation fault. So I reinstalled imagemagick and purged graphicsmagick-imagemagick-compat.

It is best to entirely avoid using convert (ImageMagick), because it often overcommits virtual and physical memory (run top and watch the VIRT and RES columns), which prevents other processes from running, which in turn causes the kernel to kill unresponsive processes at random and thereby hang the machine.

With our settings, importDump.php calls gm convert (GraphicsMagick) to make thumbnail images; and calls rsvg to convert Scaled Vector Graphics (SVG) images into PNG images (since SVG is not yet standardized, most browsers will not render it).

Finally, store the images on a separate drive

```
root-shell# cd /var/lib/mediawiki
root-shell# mv images /database0/.                  <-- this is my RAID array
root-shell# ln -s /database0/images images          <-- don't forget this
root-shell# cd /database0/images
root-shell# mkdir archive temp thumb tmp
root-shell# cd ..
root-shell# chown -R www-data:www-data images
```

Go to http://localhost/mediawiki/index.php/Main_Page. Done.

If you mess up the importation of Wikipedia dumps (next section), and want to start over. The easiest method is to drop the wikidb database, and reinstall mediawiki.

```
shell$ mysql -u root -p
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| wikidb             |
+--------------------+
3 rows in set (0.09 sec)
mysql> DROP DATABASE wikidb;
Query OK, 35 rows affected (1.88 sec)
mysql> USE mysql;
mysql> SELECT user,host FROM user;
+------------------+-----------------------+
| user             | host                  |
+------------------+-----------------------+
| wikiuser         | %                     |
| root             | 127.0.0.1             |
| root             | darkstar-5            |
| debian-sys-maint | localhost             |
| root             | localhost             |
| wikiuser         | localhost             |
| wikiuser         | localhost.localdomain |
+------------------+-----------------------+
7 rows in set (0.00 sec)
mysql> DROP USER 'wikiuser'@'%';
mysql> DROP USER 'wikiuser'@'localhost';
mysql> DROP USER 'wikiuser'@'localhost.localdomain';
mysql> exit;
```

Update (2012-Nov): For MediaWiki 1.17 and later versions, maintenance scripts now respects a --memory-limit option

```
shell$ cat /usr/share/mediawiki/maintenance/Maintenance.php
...
        /**
         * Normally we disable the memory_limit when running admin scripts.
         * Some scripts may wish to actually set a limit, however, to avoid
         * blowing up unexpectedly. We also support a --memory-limit option,
         * to allow sysadmins to explicitly set one if they'd prefer to override
         * defaults (or for people using Suhosin which yells at you for trying
         * to disable the limits)
         * @return string
         */
...
```

Source: http://www.mediawiki.org/wiki/Manual:Maintenance_scripts.

E.6 Experiments with MWdumper.jar—Round 1

Install packages:

```
root-shell# aptitude purge mediawiki mediawiki-extensions mediawiki-math
```

Go to http://download.wikimedia.org/enwiki/latest/ and look-up the names of the dump files, and download them:

```
shell$ mkdir wikipedia
shell$ cd wikipedia
shell$ wget http://download.wikimedia.org/enwiki/latest/enwiki-latest-md5sums.txt
shell$ wget http://download.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2
shell$ wget http://download.wikimedia.org/enwiki/latest/enwiki-latest-pagelinks.sql.gz
shell$ gunzip enwiki-latest-pagelinks.sql.gz
```

Estimate the number of page, revision, text, and pagelink records:

```
shell$ cat enwiki-latest-pages-articles.xml | grep "<page>" | wc -l
10355225
shell$ cat enwiki-latest-pagelinks.sql | tr -cd "(" | wc -c
522713440
```

Importation of 10M pages and 500M links will be the main challenge.
There are two distinct methods:

- MWdumper.jar - only used for importing initial set of pages, and
- importDump.php - best used for importing latest revisions.

MWdumper.jar must start with a clean database. If you first tried importDump.php but aborted after seeing that it is too slow, then some tables in wikidb contain records that must first be removed. Go back and drop wikidb and reinstall MediaWiki.
Import with MWdumper.jar. This requires a clean wikidb. Note that it only imports pages, and that links must be imported separately.

```
shell$ wget htpp://download/wikimedia.org/tools/mwdumper.jar
shell$ wget htpp://download/wikimedia.org/tools/README.txt
```

For a dry run:

```
shell$ java -jar mwdumper.jar --format=sql:1.5 enwiki-latest-pages-articles.xml.bz2
```

This failed (circa page 309,000 in a lengthy article about Croatia) due to page data that results in a malformed SQL query.
For import, load pages with:

```
shell$ java -jar mwdumper.jar --format=sql:1.5 enwiki-latest-pages-articles.xml.bz2 \
 | mysql -u wikiuser -p wikidb
```

After running MWdumper.jar, expect to see records only in the following database tables: interwiki, objectcache, page, revision, site_stats, text, user, and user_groups.
If MWdumper.jar supplies a bad SQL query, MySQL gives an error message:

```
ERROR 1064 (42000) at line 6115: You have an error in your SQL syntax;
check the manual that corresponds to your MySQL server version for the
right syntax to use near ''==Earliest history==\nThe details of the
arrival of the [[Croats]] are scarcely' at line 1
make: *** [mwdumper] Error 1
```

In this case, truncate the relevant tables:

```
shell$ mysql -u wikiuser -p wikidb
mysql> TRUNCATE TABLE page;
mysql> TRUNCATE TABLE revision;
mysql> TRUNCATE TABLE text;
mysql> exit;
```

Then try again using the force flag `-f`:

```
shell$ java -jar mwdumper.jar --format=sql:1.5 enwiki-latest-pages-articles.xml.bz2 \
 | mysql -f -u wikiuser -p wikidb
```

This also failed (circa page 309,000).
Summary of results:

- Failed every time (due to unparsable page).
- Must find and remove bad page from `dump` file.
- Must start over with empty database tables.
- Fast, but does not populate the `pagelinks` table.
- Must run `rebuildall.php` which takes days or weeks.
- Failed every time (due to memory leak that hangs PC)

Recommendation: Do not use.
References:
1) http://www.mediawiki.org/wiki/Manual:Importing_XML_dumps
2) http://www.mediawiki.org/miki/Mwdumper

E.7 Experiments with InnoDB

Purpose: Determine the optimal configuration of hardware, MySQL, and InnoDB.

E.7.1 Experimental Method

Overall method: Measure the time required to load 500m `pagelinks` v. a variety of configurations.

Load `pagelinks` with:

```
shell$ gunzip enwiki-latest-pagelinks.sql.gz
shell$ mysql -u wikiuser -p wikidb < enwiki-latest-pagelinks.sql
```

Take measurements with:

```
mysql> SELECT CURRENT_TIMESTAMP(),COUNT(*),MAX(pl_from),
    -> COUNT(*)/MAX(pl_from) AS links_per_page FROM wikidb.pagelinks;
+---------------------+----------+--------------+----------------+
| current_timestamp() | count(*) | max(pl_from) | links_per_page |
+---------------------+----------+--------------+----------------+
| 2010-09-17 19:56:03 | 51373348 |      1228877 |        41.8051 |
+---------------------+----------+--------------+----------------+
1 row in set (8 min 31.87 sec)
```

and

```
shell$ atsar -D
08:40:01  partition   busy  read/s Kbyt/r   write/s Kbyt/w  avque avserv _part_
23:50:01  sdc (8-32)   90%   65.10   24.6    225.72   22.6   3.42   3.09 ms
03:30:02  sdc (8-32)   80%   45.58   24.5    154.86   21.5   4.10   3.97 ms
21:00:02  sdc (8-32)   91%   65.08   23.0    227.03   20.1   3.34   3.12 ms
```

Experimental method:

1. **Setup:**
   - Edit /etc/mysql/conf.d/custom.cnf (if using hugepages)
     ```
     root-shell# sysctl -p                          <-- to reserve hugepages
     root-shell# reboot                             <-- if memory already fragmented
     ```
   - Edit /etc/sysctl.conf (as described below)
     ```
     root-shell# /etc/init.d/mysql restart
     root-shell# aptitude install mediawiki mediawiki-extensions
     ```
   - Setup mediawiki (as described above)
2. **Run:**
   - In one terminal run the test:
     ```
     shell$ mysql -u root -p wikidb < enwiki-latest-pagelinks.sql
     ```
   - In another terminal collect data:
     ```
     shell$ mysql -u wikiuser -p wikidb
     mysql> SELECT CURRENT_TIMESTAMP(),COUNT(*),MAX(pl_from),
         -> COUNT(*)/MAX(pl_from) AS links_per_page FROM wikidb.pagelinks;
     ```
3. **Tear-down:**
   - Purge database:
     ```
     mysql> TRUNCATE TABLE wikidb.pagelinks;
     mysql> DROP DATABASE wikidb;                <-- if ibdata1 is to be relocated
     root-shell# /etc/init.d/mysql stop
     root-shell# rm /var/lib/mysql/ibdata1        <-- ibdata1 may be on another disk
     root-shell# rm /var/lib/mysql/ib_logfile*
     ```
   - Purge packages:
     ```
     root-shell# aptitude purge mediawiki mediawiki-extensions
     ```

E.7.2 EXPERIMENT.0 Baseline configuration

Code: D1F+L5P8H- (D=disks, F=filesys, L=log[MB], P=pool[MB], H=hugepages)

- **D1** /var/lib/mysql/ibdata1 and /var/lib/mysql/ib_logfile* are on the same disk,
- **F+** /var/lib/mysql/ibdata1 is a file on a file system,
- **L5** /var/lib/mysql/ib_logfile* are 5M each,
- **P8** InnoDB buffer pool is 8M, and
- **H-** hugepages are not used.

Configuration files:

```
root-shell# cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone      = UTC
default-storage-engine = innodb
default-character-set  = utf8
default-collation      = utf8_general_ci
[mysql]
default-character-set  = utf8
```

Measurements:

```
Time   1   1  hours, insert  5.0M   5.0M links,  1390  1390 links/sec
Time   2   1  hours, insert  7.4M   2.4M links,  1030   670 links/sec
Time  12  10  hours, insert 22.4M  14.0M links,   510   390 links/sec
kill.
```

E.7.3 EXPERIMENT.1 Put ibdata1 on separate disk (but still on a file system)

Code: D2F+L5P8H-

- **D2** /var/lib/mysql/ibdata1 and /var/lib/mysql/ib_logfile* are on separate disks

Configuration files:

```
shell$ cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone      = UTC
default-storage-engine = innodb
default-character-set  = utf8
default-collation      = utf8_general_ci
innodb_data_home_dir   =
innodb_data_file_path  = /database0/images0/ibdata1:10M:autoextend
[mysql]
default-character-set  = utf8
```

Measurements:

```
Time   1   1  hours, insert  6.9M   6.9M links,  1920  1920 links/sec
Time   2   1  hours, insert 11.0M   4.1M links,  1530  1140 links/sec
Time  12  10  hours, insert 24.0M  13.0M links,   560   360 links/sec
kill.
```

E.7.4 EXPERIMENT.2 Store ibdata1 on raw LVM2 partition (not a file system)

```
root-shell# chown mysql /dev/mapper/vg5-ibdata0
```

Code: D2F-L5P8H-

- **F-** /var/lib/mysql/ibdata1 does not use a file system (it is written directly on a raw partition)

Configuration files:

```
shell$ cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone      = UTC
default-storage-engine = innodb
default-character-set  = utf8
default-collation      = utf8_general_ci
innodb_data_home_dir   =
innodb_data_file_path  = /dev/mapper/vg5-ibdata0:100000Mnewraw  # <-- note
#innodb_data_file_path  = /dev/mapper/vg5-ibdata0:100000Mraw
[mysql]
default-character-set  = utf8
```

Create the table space on the raw partition. Here we set file size to 100GB, which takes about 20 minutes to physically write.

```
root-shell# /etc/init.d/mysql start
```

Progress is logged to /var/log/syslog.

```
root-shell# cat /var/log/syslog | grep InnoDB | tail
Sep 28 11:56:54 darkstar-5 mysqld[649]: 100928 11:56:54  InnoDB: Setting file
 /dev/mapper/vg5-ibdata0 size to 102400 MB
Sep 28 11:56:54 darkstar-5 mysqld[649]: InnoDB: Database physically writes the
 file full: wait...
Sep 28 12:00:24 darkstar-5 mysqld[649]: InnoDB: Progress in MB: 100 200 300
```

Stop mysqld, edit custom.cnf, and start again

```
root-shell# /etc/init.d/mysql stop
root-shell# cat /etc/mysql/conf.d/custom.cnf | grep raw
#innodb_data_file_path  = /dev/mapper/vg5-ibdata0:100000Mnewraw
innodb_data_file_path  = /dev/mapper/vg5-ibdata0:100000Mraw     # <-- note
root-shell# /etc/init.d/mysql start
```

Measurements:

```
Time   1   1  hours, insert  7.4M   7.4M links,  2060  2060 links/sec
Time   2   1  hours, insert 12.2M   4.8M links,  1690  1330 links/sec
Time  12  10  hours, insert 24.1M  11.9M links,   560   330 links/sec
kill.
```

E.7.5 EXPERIMENT.3 Increase size of buffer pool

Code: D2F-L5P100H-
Configuration files:

```
shell$ cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone               = UTC
default-storage-engine          = innodb
default-character-set           = utf8
default-collation               = utf8_general_ci
innodb_data_home_dir            =
#innodb_data_file_path           = /dev/mapper/vg5-ibdata0:100000Mnewraw
innodb_data_file_path           = /dev/mapper/vg5-ibdata0:100000Mraw
innodb_buffer_pool_size         = 100M # default  8M.
innodb_additional_mem_pool_size =   2M # default  1M.
[mysql]
default-character-set           = utf8
```

Measurements:

```
Time   1   1  hours, insert 10.1M  10.1M links,  2800  2800 links/sec
Time   2   1  hours, insert 16.6M   6.5M links,  2300  1800 links/sec
Time  12  10  hours, insert 49.6M  33.0M links,  1150   920 links/sec
kill.
```

E.7.6 EXPERIMENT.4 Increase size of log file and log buffer

Code: D2F-L250P100H-

Configuration files:

```
shell$ cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone               = UTC
default-storage-engine          = innodb
default-character-set           = utf8
default-collation               = utf8_general_ci
innodb_data_home_dir            =
innodb_data_file_path           = /dev/mapper/vg5-ibdata0:100000Mraw
innodb_buffer_pool_size         = 100M # default  8M.
innodb_additional_mem_pool_size =   2M # default  1M.
innodb_log_file_size            = 250M # default  5M.
innodb_log_buffer_size          =   8M # default  1M.
[mysql]
default-character-set           = utf8
```

Measurements:

```
Time   1   1  hours, insert 13.2M  13.2M links,  3670  3670 links/sec
Time   2   1  hours, insert 19.4M   6.2M links,  2690  1720 links/sec
Time  12  10  hours, insert 53.4M  34.0M links,  1240   940 links/sec
kill.
```

E.7.7 EXPERIMENT.5 Increase size of buffer pool

Code: D2F-L250P1000H-

Configuration files:

```
shell$ cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone               = UTC
default-storage-engine          = innodb
default-character-set           = utf8
default-collation               = utf8_general_ci
innodb_data_home_dir            =
#innodb_data_file_path           = /dev/mapper/vg5-ibdata0:100000Mnewraw
innodb_data_file_path           = /dev/mapper/vg5-ibdata0:100000Mraw
innodb_buffer_pool_size         =1000M # default  8M.
innodb_additional_mem_pool_size =   2M # default  1M.
innodb_log_file_size            = 250M # default  5M.
innodb_log_buffer_size          =   8M # default  1M.
[mysql]
default-character-set           = utf8
```

Measurements:

```
Time   1   1  hours, insert  36.0M  36.0M links, 10000 10000 links/sec
Time   2   1  hours, insert  46.4M  10.4M links,  6440  2890 links/sec
Time  12  10  hours, insert 131.8M  85.4M links,  3050  2370 links/sec
kill.
```

E.7.8 EXPERIMENT.6 Enable hugepages

Code: D2F-L250P1000H+

Configuration files:

```
shell$ getent group hugepage
hugepage:x:1001:mysql
root-shell# mkdir /hugepages
root-shell# chown root:hugepage hugepages
```

```
root-shell# tail /etc/security/limits.conf
@hugepage        -       memlock 1024000
mysql            -       memlock 1024000
root-shell# tail /usr/bin/mysqld_safe
ulimit -l unlimited
shell$ cat /etc/fstab | grep hugepage
hugetlbfs          /hugepages         hugetlbfs   mode=1770,gid=1001 0  0
shell$ tail /etc/sysctl.conf
vm.nr_hugepages      =        512 # default:        0 2MiB      =  0MB.
vm.hugetlb_shm_group =       1001
kernel.shmmax        = 1073741824 # default: 33554432 bytes     = 32MB.
kernel.shmall        =     262144 # default:  2097152 4KB pages =  8GB.
```

```
shell$ cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone               = UTC
default-storage-engine          = innodb
default-character-set           = utf8
default-collation               = utf8_general_ci
innodb_data_home_dir            =
#innodb_data_file_path           = /dev/mapper/vg5-ibdata0:100000Mnewraw
innodb_data_file_path           = /dev/mapper/vg5-ibdata0:100000Mraw
innodb_buffer_pool_size         = 1000M # default  8M.
innodb_additional_mem_pool_size =    2M # default  1M.
innodb_log_file_size            =  250M # default  5M.
innodb_log_buffer_size          =    8M # default  1M.
large-pages
[mysql]
default-character-set           = utf8
```

Measurements:

```
shell$ cat /proc/memstat
...
PageTables:      24904 kB
...
HugePages_Total:   512
HugePages_Free:     11
HugePages_Rsvd:      2
HugePages_Surp:      0
Hugepagesize:     2048 kB
```

```
Time   1   1  hours, insert  36.3M  36.3M links, 10080 10080 links/sec
Time   2   1  hours, insert  48.0M  11.7M links,  6670  3250 links/sec
Time  12  10  hours, insert 124.3M  76.3M links,  2880  2120 links/sec
kill.
```

E.7.9 EXPERIMENT.6b Repeat. Run to completion.

Measurements:

```
    Time  1.1 1.1 hours, insert  36.6M  36.6M links,  9240  9240 links/sec
    Time   2  0.9 hours, insert  46.4M   9.8M links,  6440  3020 links/sec
    Time  12  10  hours, insert 128.7M  82.3M links,  2980  2290 links/sec
    Time  48  36  hours, insert 353.0M 224.3M links,  2040  1730 links/sec
    Time  72  24  hours, insert 466.8M 113.8M links,  1800  1320 links/sec
    Time  83  11  hours, insert 499.7M  32.9M links,  1670   830 links/sec
    done.
```

Note: During hours 9 to 44, a cron job ran:

```
shell$ ps -ewf
... /usr/share/mdadm/checkarray --cron --all --quiet
```

E.7.10 EXPERIMENT.6c Repeat. Do not disable keys.

In enwiki-latest-pagelinks.sql, near the beginning and the end, respectively, one will find the two following commands. I suspect that the second takes 2-3 hours:

```
...
/*!40000 ALTER TABLE `pagelinks` DISABLE KEYS */;
...
/*!40000 ALTER TABLE `pagelinks` ENABLE KEYS */;
...
```

Code: D2F-L250P1000H+K+
Configuration files: (same as Experiment 6)
Data: enwiki-latest-pagelinks.sql, but remove line 38, which reads

```
/*!40000 ALTER TABLE `pagelinks` DISABLE KEYS */;
```

Measurements:

```
    Time   1   1  hours, insert  34.7M  34.7M links,  9640  9640 links/sec
    Time   2   1  hours, insert  46.1M  11.4M links,  6400  3170 links/sec
    Time  12  10  hours, insert 129.4M  83.3M links,  3000  2310 links/sec
    Time  24  12  hours, insert 212.2M  82.8M links,  2460  1920 links/sec
    Time  42  18  hours, insert 320.4M 108.2M links,  2120  1670 links/sec
    Time  66  24  hours, insert 419.1M  98.7M links,  1760  1140 links/sec
    Time  90  24  hours, insert 499.7M  80.6M links,  1540   930 links/sec
    done.
```

Conclusion:
Apparently disabling keys has little or no effect on InnoDB. Presumably this matters more for MyISAM.

E.7.11 EXPERIMENT.7 Increase innodb_buffer_pool_size with filesystem, hugepage

Code: D2F+L250P49H+
Configuration files:

```
shell$ cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone               = UTC
default-storage-engine          = innodb
default-character-set           = utf8
default-collation               = utf8_general_ci
innodb_data_home_dir            =
innodb_data_file_path           = /database0/mysql/ibdata1:1024M:autoextend
innodb_buffer_pool_size         =   49M # default  8M.
innodb_additional_mem_pool_size =    2M # default  1M.
innodb_log_file_size            =  250M # default  5M.
innodb_log_buffer_size          =    8M # default  1M.
large-pages
[mysql]
default-character-set           = utf8
```

Measurements:

```
Time  4.5 4.5 hours, insert  30M  30M links, 1850 1850 links/sec
Time 15.5 11  hours, insert  62M  32M links, 1110  810 links/sec
Time 23.5  8  hours, insert  76M  14M links,  900  490 links/sec
Time 39.5 16  hours, insert 100M  24M links,  700  420 links/sec
Time 51.5 12  hours, insert 121M  21M links,  650  490 links/sec
Time 60.5  9  hours, insert 134M  13M links,  620  400 links/sec
kill.
```

E.7.12 EXPERIMENT.8 Increase innodb_buffer_pool_size again with filesystem, hugepage

Code: D2F+L250P999H+

Configuration files:

```
shell$ cat /etc/mysql/conf.d/custom.cnf
[mysqld]
default-time-zone               = UTC
default-storage-engine          = innodb
default-character-set           = utf8
default-collation               = utf8_general_ci
innodb_data_home_dir            =
innodb_data_file_path           = /database0/mysql/ibdata1:1024M:autoextend
innodb_buffer_pool_size         =  999M # default  8M.
innodb_additional_mem_pool_size =    2M # default  1M.
innodb_log_file_size            =  250M # default  5M.
innodb_log_buffer_size          =    8M # default  1M.
large-pages
[mysql]
default-character-set           = utf8
```

Measurements:

```
   Time   4   4  hours, insert  68M  68M links, 4720 4720 links/sec
   Time   8   4  hours, insert 100M  32M links, 3470 2220 links/sec
   Time  20  12  hours, insert 187M  87M links, 2600 2010 links/sec
   Time  38  18  hours, insert 286M  99M links, 2100 1530 links/sec
   Time  62  24  hours, insert 389M 103M links, 1740 1190 links/sec
   Time  86  24  hours, insert 466M  77M links, 1510  890 links/sec
   Time 100  14  hours, insert 500M  34M links, 1390  670 links/sec
   done.
```

Table E.1: InnoDB Experiments—Codes

Code	Description	Default
D1	ib_data0, ib_logfile* all on same disk	Y
D2	ib_data0 is on a disk separate from ib_logfile*	
F+	ib_data0 is on a file system (ReiserFS)	Y
F-	ib_data0 is on a raw partition	
L5	ib_logfile* are 5M each	Y
L250	ib_logfile* are 250M each	
P8	ib_data0 is 8M	Y
P50	ib_data0 is 50M	
P100	ib_data0 is 100M	
P1000	ib_data0 is 1000M	
H-	hugepages not enabled	Y
H+	hugepages enabled	

Storage of pagelinks:

- ibdata1 49GB,
- ibdata1 53588525056 bytes/499678288 links = 107 bytes/link,
- pagelinks.sql 16845703455 bytes/499678288 links = 33 bytes/link,

E.7.13 Conclusions

Configuration of MySQL can yield order-of-magnitude improvements in database performance.

By far, the largest performance gain (2-5x) came from increasing the size of the InnoDB buffer pool. The purchase of more DRAM is probably money well spent.

Lesser gains were obtained by: 1) using separate disks for the InnoDB table space and log files (1.07), 2) using larger InnoDB log files (1.08), and 3) putting the table space on a raw partition (1.07).

No gains were seen from enabling hugepages.

The results are tabulated in Table E.2, InnoDB Experimental Results—Performance Ratios and Table E.3, InnoDB Experimental Results—Summary; and charted in Figure E.1, InnoDB Experiments—Importing imagelinks Table.

Update (2012): The configuration currently running on darkstar-5 is (D2F-L250P3000H+). This is like Experiment 6, except the buffer pool is 3G instead on 1G.

E.8 Experiments with importDump.php—Round 1

These experiments were among the earliest (Summer 2010). This is mostly a transcript of the authors log from that time (see References at the end of this section).

Install packages: Same as §E.6, Experiments with MWdumper.jar—Round 1.

Download dump files: Same as §E.6, Experiments with MWdumper.jar—Round 1.

Decompress: 6.3G expands to 27G.

```
shell$ bunzip2 -c enwiki-latest-pages-articles.xml.bz2 > enwiki-latest-pages-articles.xml
```

For a dry run, edit /usr/share/mediawiki/maintenance/importDump.php, and set

```
var $dryRun = true;
```

and run:

Table E.2: InnoDB Experimental Results—Performance Ratios

Exp.	Configuration	Time [h]	Pagelinks Pagelinks	Base Ratio	Pair Ratio	Comment
Base	D1F+L5P8H-	12	22.4	1.00	1.00	
1	D2F+L5P8H-	12	24.0	1.07	1.07	separate disks
2	D2F-L5P8H-	12	24.1	1.08	1.00	
3	D2F-L5P100H-	12	49.6	2.21	2.06	larger buffer pool
4	D2F-L250P100H-	12	53.4	2.38	1.00	
5	D2F-L250P1000H-	12	131.8	5.88	2.47	larger buffer pool
7	D2F+L250P50H+	60.5	134		1.00	
8	D2F+L250P1000H+	63	389		2.90	larger buffer pool
5	D2F-L250P1000H-	12	131.8	5.88	1.00	
6b	D2F-L250P1000H+	12	128.7	5.75	0.98	hugepages
3	D2F-L5P100H-	12	49.6	2.21	1.00	
4	D2F-L250P100H-	12	53.4	2.38	1.08	larger log file
8	D2F+L250P1000H+	86	466		1.00	
6b	D2F-L250P1000H+	83	499.7		1.07	raw partition

Table E.3: InnoDB Experimental Results—Summary

Code	Configuration	Presumed Benefit	Measured Ratio
P1000	Larger InnoDB buffer pool	Reduce disk activity	2-5x
D2	Separate disks for InnoDB table space and log files	Parallelize disk activity	1.07
L250	Larger InnoDB log files	Reduce disk activity	1.08
F-	Use raw partition for InnoDB table space	Eliminate double journaling	1.07
H+	Enable hugepages	Eliminate swapping and reduce TLB activity	0.98

```
shell$ php /usr/share/mediawiki/maintenance/importDump.php \
 < enwiki-latest-pages-articles.xml > importDump.log
```

For import, set

```
var $dryRun = false;
```

and run:

```
shell$ php /usr/share/mediawiki/maintenance/importDump.php \
 < enwiki-latest-pages-articles.xml > importDump.log
```

The code that does the work is class WikiImporter which can be found in /usr/share/mediawiki/includes/SpecialImport.php.

MySQL configuration (see §E.7, Experiments with InnoDB) improved importation speed (see Table E.4, Speed of MWdumper.jar v. importDump.php). Importation is bound by the effort spent searching index B-trees for pagelinks to enforce UNIQUE INDEX and UNIQUE KEY constraints, and this is unnecessary for the initial bulk importation. Hence, the alternative MWdumper.jar.

Error messages:

Figure E.1: InnoDB Experiments—Importing imagelinks Table

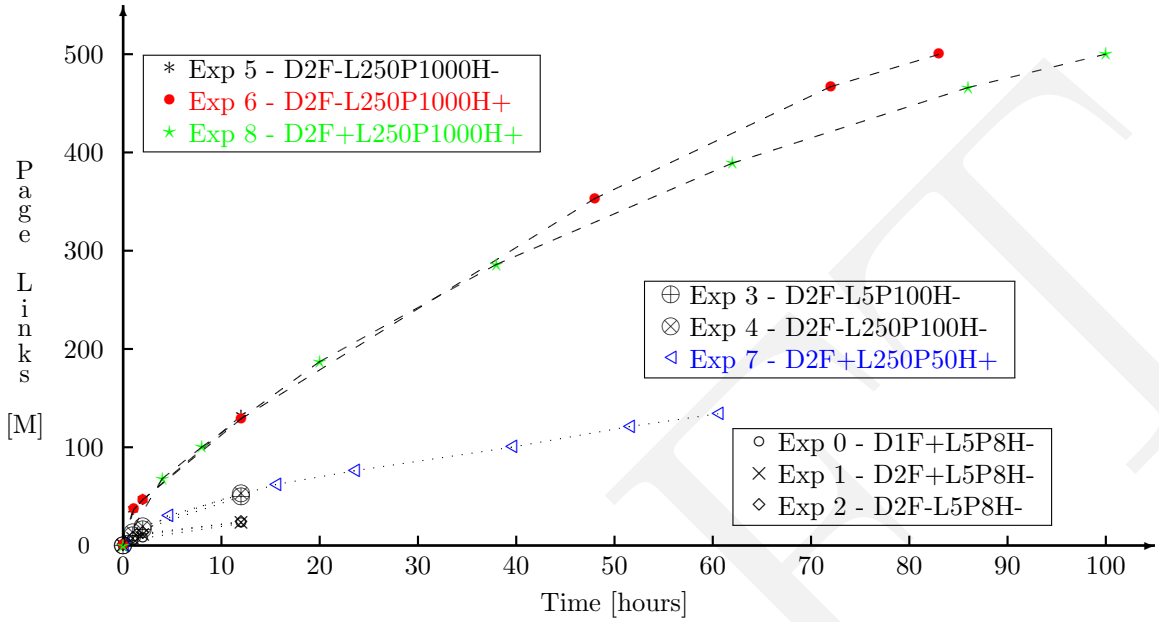


Table E.4: Speed of MWdumper.jar v. importDump.php

Method	Dry Run	MySQL	Rate [page/s]	Est. Time [s]	Est. Time [d]	Outcome
mwdumper.jar	true	N/A	400	25,000	0.3	Fail
	false	Exper.6	200	50,000	0.6	Fail 309,000
importDump.php	true	N/A	400	25,000	0.3	
	false	Baseline	2.5	4,000,000	47	
		Exper.6	29	345,000	4.0	1-309,000
			1.0			Fail 509,200
		Exper.6	395			

1) There were problems with translating math from TEX format to PNG images. Many warning messages appeared, like this:

```
Warning: shell_exec(): Unable to execute '/usr/bin/texvc '/var/lib/mediawiki/images/tmp'
'/var/lib/mediawiki/images/tmp' 'w(x)' 'UTF-8'' in /usr/share/mediawiki/includes/Math.php
on line 70
```

This was probably due to insufficient DRAM (these messages would come in a burst when memory got low). Also seen was:

```
Failed to parse (Can't write to or create math output directory): w(x)
```

Try again:
After 509,200 pages, importDump.php aborted with the message:

```
<p>Set <tt>$wgShowExceptionDetails = true;</tt> in LocalSettings.php to
show detailed debugging information.</p>
```

and from the log file:

```
root-shell# cat /var/log/syslog | grep mysql
```

it looks like mysqld died after running out of memory. Restarting the daemon:

```
root-shell# /etc/init.d/mysql start
```

caused InnoDB to replay transactions in the double-write buffer, identify 1 transaction with 5 rows for rollback, replay 100 transactions in the logfile, and then rollback the 5 rows.

Try again:

It starts over, and continues well past the point it crashed. After 670,978 pages, importDump.php aborted, and again it seems that mysqld ran out of memory. Same replay and rollback, except rollback 1 transaction with 7 rows.

Summary of results:

- Failed every time (due to unparsable page).
- Must find and remove bad page from dump file.
- Can resume (rapidly skips over imported pages).
- Slow, but does pagelinks.
- Do not need rebuildall.php.
- Failed every time anyway (due to memory leak).

Lesson learned: Import no more than 100k pages at a time.

Recommendation: Split large dump file into xchunks, import each xchunk *seriatim*, and tolerate failure of a few xchunks.

Historical note: The notion of splitting large dump files, was the idea that motivated the development of WP-MIRROR.

E.9 Experiments with MWdumper.jar—Round 2

Whereas the slow speed of importDump.php (about 1 page/sec, if 500m pagelinks must be searched) in comparison to the speed of mwdumper.jar (about 200 pages/sec when piped into MySQL; 2,500 pages/sec when writing an SQL file; 4,800 pages/sec when writing to /dev/null); and

Whereas they both crash: importDump.php (when out of memory) and mwdumper.jar (when it encounters pages that constitute invalid XML; or, if valid XML, yield invalid SQL INSERT statements);

Resolved that it makes more sense to write some software to remove the offending pages from enwiki-latest-pages-articles.xml and then try using mwdumper.jar for importation. It seems best to avoid MySQL until mwdumper.jar has successfully produced an SQL file.

To that end, I wrote remove_page.lisp that can be given the start and end of a range of pages to be removed. It is reasonably fast (about 2,100 pages/sec).

Process:

```
shell$ java -jar mwdumper.jar --format=sql:1.5 enwiki-latest-pages-articles.xml
 > enwiki-latest-pages-articles.sql
```

This crashed after 309,000 pages; so remove pages 309,000 through 310,000, and try again.

```
shell$ remove_page.lisp --start 309000 --end 310000 < enwiki-latest-pages-articles.xml \
 > enwiki-309k-310k-pages-articles.xml
shell$ java -jar mwdumper.jar --format=sql:1.5 enwiki-309k-310k-pages-articles.xml > /dev/null
```

This crashed after 1,172,000 pages, so remove pages 1172k-1173k, try again.
This crashed after 1,398,000 pages, etc.
This crashed after 2,218,000 pages
This crashed after 2,296,000 pages
This crashed after 2,755,000 pages
This crashed after 5,704,000 pages
This crashed after 5,749,000 pages
This crashed after 6,261,000 pages
This crashed after 6,397,000 pages
This crashed after 6,397,000 pages again (obviously not the same page)
This crashed after 8,124,000 pages
Done. (12 crashes. It took two days to get a valid SQL file)
Now direct output into an SQL file (instead of /dev/null). This will take a bit longer (about 2,000 pages/sec instead of 4,800 pages/sec).

```
shell$ java -jar mwdumper.jar --format=sql:1.5 enwiki-8124k-8125k-pages-articles.xml \
 > enwiki-8124k-8125k-pages-articles.sql
shell$ mysql -u root -p wikidb
mysql> DROP DATABASE wikidb;
mysql> quit;
```

Dropping the wikidb database took several minutes.

```
# aptitude purge mediawiki mediawidi-extensions mediawiki-math
```

Repeat MediaWiki installation. Load SQL file into wikidb.

```
shell$ mysql -u root -p wikidb < enwiki-8124k-8125k-pages-articles.sql
Enter password: ********
ERROR 1062 (23000) at line 26237: Duplicate entry '0-' for key 2
```

In a different term box:

```
shell$ mysql -u wikiuser -p wikidb
Enter password: ********
mysql> SELECT COUNT(*) FROM wikidb.page;
+----------+
| count(*) |
+----------+
|  2389001 |
+----------+
1 row in set (11.13 sec)
```

This crashed after 2,389,000 pages. "line 26237: Duplicate".
So remove pages 2,389k through 2,390k, and try again.

```
shell$ remove_page.lisp --start 2389000 --end 2390000 < enwiki-8124k-8125k-pages-articles.xml \
 > enwiki-2389k-2390k-pages-articles.xml
shell$ java -jar mwdumper.jar --format=sql:1.5 enwiki-2389k-2390k-pages-articles.xml \
 > enwiki-2389k-2390k-pages-articles.sql
shell$ mysql -u root -p wikidb
mysql> DROP DATABASE wikidb;
mysql> quit;
```

Repeat MediaWiki installation.

```
shell$ mysql -u root -p wikidb < enwiki-2389k-2390k-pages-articles.sql
Enter password: ********
ERROR 1062 (23000) at line 26851: Duplicate entry '0-' for key 2
```

This crashed after 2,462,000 pages. "line 26851: Duplicate". Repeat
This crashed after 3,258,000 pages. "line 33565: Duplicate". Repeat
This crashed after 3,266,000 pages. "line 33632: Duplicate". Repeat
This crashed after 3,273,000 pages. "line 33690: Duplicate". Repeat
This crashed after 3,332,000 pages. "line 34182: Duplicate". Repeat
This crashed after 3,402,000 pages. "line 34756: Duplicate". Repeat
Quit. (the last 144,000 took 5 days; importDump.php does 100k/day)

Give up on mwdumper.jar. We inserted pages with page_id up to 7,950,000. We shall insert the pagelinks for those pages; that is, just 160m pagelinks (1.5 days), enough so that max(pl_from) exceeds 8,000,000.

```
shell$ mysql -u root -p wikidb < latest/enwiki-latest-pagelinks.sql
```

The remaining 6.9m pages shall be inserted by importDump.php (two months).

```
root-shell# php /usr/share/mediawiki/maintenance/importDump.php \
 < latest/enwiki-latest-pages-articles.xml > importDump.log
```

Monitor progress with:

```
mysql> SELECT MAX(page_id),7950022 AS mstart,MAX(page_id)-7950022 AS mdiff,
    -> COUNT(*) AS pages,3402001 AS pstart,COUNT(*)-3402001 AS pdiff FROM page;
+--------------+---------+-------+---------+---------+-------+
| max(page_id) | mstart  | mdiff | pages   | pstart  | pdiff |
+--------------+---------+-------+---------+---------+-------+
|      7951792 | 7950022 |  1770 | 3403814 | 3402001 |  1813 |
+--------------+---------+-------+---------+---------+-------+
mysql> SELECT MAX(pl_from),8124748 AS start,MAX(pl_from)-8124748 AS diff FROM pagelinks;
+--------------+---------+------+
| max(pl_from) | start   | diff |
+--------------+---------+------+
|      8124748 | 8124748 |    0 |
+--------------+---------+------+
```

Note: the measurement above was taken just after 100k pages.

After a 2 1/2 days (25 pages/sec), the system hung and had to be rebooted. InnoDB took about 10 minutes to complete replay/rollback.

```
mysql> SELECT MAX(page_id),7950022 AS mstart,MAX(page_id)-7950022 AS mdiff,
    -> COUNT(*) AS pages,3402001 AS pstart,COUNT(*)-3402001 AS pdiff FROM page;
    -> SELECT MAX(pl_from),8124748 AS start,MAX(pl_from)-8124748 AS diff FROM pagelinks;
+--------------+---------+--------+---------+---------+--------+
| max(page_id) | mstart  | mdiff  | pages   | pstart  | pdiff  |
+--------------+---------+--------+---------+---------+--------+
|      8239053 | 7950022 | 289031 | 3691075 | 3402001 | 289074 |
+--------------+---------+--------+---------+---------+--------+
+--------------+---------+--------+
| max(pl_from) | start   | diff   |
+--------------+---------+--------+
|      8239053 | 8124748 | 114305 |
+--------------+---------+--------+
mysql> EXPLAIN SELECT COUNT(*) FROM pagelinks\G
        rows: 176926947
```

So it appears that about 1/3 of the pages have been loaded. In other words:
This hung after 8,239,053 / 3,691,075 (MAX(page_id)/COUNT(*)). Try again.
The first 2.88m are quickly replayed (480 pages/sec).
This crashed after 8,634,486 / 4,086,508 loaded. Try again.

```
+--------------+---------+--------+---------+---------+--------+
| max(page_id) | mstart  | mdiff  | pages   | pstart  | pdiff  |
+--------------+---------+--------+---------+---------+--------+
|      8634486 | 7950022 | 684464 | 4086508 | 3402001 | 684507 |
+--------------+---------+--------+---------+---------+--------+
+--------------+---------+--------+
| max(pl_from) | start   | diff   |
+--------------+---------+--------+
|      8634486 | 8124748 | 509738 |
+--------------+---------+--------+
```

This crashed after 8,840,722 / 4,292,744 loaded. Try again.

```
+--------------+---------+--------+---------+---------+--------+
| max(page_id) | mstart  | mdiff  | pages   | pstart  | pdiff  |
+--------------+---------+--------+---------+---------+--------+
|      8840722 | 7950022 | 890700 | 4292744 | 3402001 | 890743 |
+--------------+---------+--------+---------+---------+--------+
+--------------+---------+--------+
| max(pl_from) | start   | diff   |
+--------------+---------+--------+
|      8840722 | 8124748 | 715974 |
+--------------+---------+--------+
```

Abort. Too many pages had messed up equations (due to texvc failing to execute for lack of memory), and too many pages had messed up references (for unknown reasons). Also it seemed that running mwdumper.jar followed by importDump.php more than doubled the MAX(page_id) for the same number of pages.

So I give up on using the (probably) unsupported mwdumper.jar, and return to using the supported importDump.php only.

E.10 Experiments with importDump.php—Round 2

This time I also run top in a terminal box because importDump.php requests more and more memory until first texvc cannot run and then later the system hangs.

The plan is this: after each 100k pages (or if the swap partition gets down to around 100MB), kill importDump.php, wait for InnoDB to flush its buffer pool pages, then restart importDump.php. That way texvc should not fail for lack of memory. It will probably be necessary to kill and restart importDump.php daily.

So let us test the kill and restart behavior:

```
root-shell# php /usr/share/mediawiki/maintenance/importDump.php \
 < latest/enwiki-latest-pages-articles.xml
^C
```

and in another terminal:

```
mysql> SELECT MAX(page_id),COUNT(*) AS pages FROM page; SELECT MAX(pl_from),
    -> COUNT(*) AS links FROM pagelinks;
+--------------+-------+
| max(page_id) | pages |
+--------------+-------+
|        45593 | 45593 |
+--------------+-------+
+--------------+---------+
| max(pl_from) | links   |
+--------------+---------+
|        45593 | 4830549 |
+--------------+---------+
```

Restart. Test if the number of pages and pagelinks is preserved, that is, if rerunning importDump.php causes any mess. Result: rerunning did not cause max(page_id) to differ from pages, as least for the pages already imported. Good!

Assuming 2.5 pages/sec, importing en wikipedia should take 7 weeks to complete. However I expect it to be less because the number of pagelinks per page will decline (and therefore offset the usual 1/log N trend) from a bit over 100 links/page to bit less than 50.

I am still troubled by failure to exec texvc when I try to import too many pages (say 100k) at a time. Is there a memory leak? Is garbage collection not happening? I do not know. So, I wrote split-page.lisp to break the large file, enwiki-latest-pages-articles.xml, into smaller files containing 50k pages each, named:

enwiki-latest-pages-articles.page000m000k.xml,
enwiki-latest-pages-articles.page000m050k.xml, and so on.

```
shell$ ./split-page.lisp -v --count 50000 enwiki-latest-pages-articles.xml
root-shell# php /usr/share/mediawiki/maintenance/importDump.php \
 < lisp/enwiki-latest-pages-articles.page000m000k.xml
shell$ rm enwiki-latest-pages-articles.page000m000k.xml
root-shell# php /usr/share/mediawiki/maintenance/importDump.php
 < lisp/enwiki-latest-pages-articles.part000m050k.xml
shell$ rm enwiki-latest-pages-articles.page000m050k.xml
```

and so on. Assuming 100k pages per day, importing should take 14 weeks. Actually, it is going faster than that (about 10 pages/sec), but only I wait between runs for InnoDB to flush its buffer pool pages (which takes several minutes).

The 7-12 pages/sec rate held for the first 3 million pages, and thereafter fell. After 5 million pages, the rate was 1-2 pages/sec.

E.11 Experiments with wikix

Install packages:

```
root-shell# aptitude install libssl-dev build-essential curl
```

Download wikix.tar.gz:

```
shell$ tar -xzpdf wikix.tar.gz
shell$ cd wikix
shell$ make
```

Generate scripts (to download images) with wikix:

```
shell$ wikix < enwiki-latest-pages-articles.xml > image
/bin/sh: line 1:  1066 Segmentation fault   wikix < ../latest/enwiki-latest-pages-articles.xml
make: *** [wikix] Error 139
```

The wikix process creates a bash script (about 2GB) for downloading images using cURL. Images are stored in a 16x16 directory tree according to some kind of hash, if /etc/mediawiki/LocalSettings.php contains the default

```
#$wgHashedUploadDirectory = false;'.
```

The wikix script failed after about 6.3 million pages were processed. It will be necessary to split the xml file and try to process the last 4 million pages separately. This was done by first splitting enwiki-latest-pages-articles.xml into files with 1m pages each, and then extracting the images from them.

```
shell$ ./split_page.lisp -c 1000000 enwiki-latest-pages-articles.xml
shell$ wikix < enwiki-latest-pages-articles.page006m.xml > image.page006m
Segmentation fault
shell$ wikix < enwiki-latest-pages-articles.page007m.xml > image.page007m
shell$ wikix < enwiki-latest-pages-articles.page008m.xml > image.page008m
shell$ wikix < enwiki-latest-pages-articles.page009m.xml > image.page009m
shell$ wikix < enwiki-latest-pages-articles.page010m.xml > image.page010m
```

Then splitting enwiki-latest-pages-articles.page006m.xml into ten parts:

```
shell$ ./split_page.lisp -c 100000 enwiki-latest-pages-articles.page006m.xml
shell$ wikix < enwiki-latest-pages-articles.page006m000k.xml > image.page006m000k
shell$ wikix < enwiki-latest-pages-articles.page006m100k.xml > image.page006m100k
shell$ wikix < enwiki-latest-pages-articles.page006m200k.xml > image.page006m200k
shell$ wikix < enwiki-latest-pages-articles.page006m300k.xml > image.page006m300k
Segmentation fault
shell$ wikix < enwiki-latest-pages-articles.page006m400k.xml > image.page006m400k
etc.
```

Note: the first million pages are the largest ones:

```
shell$ ls -lh en*m.xml
... kmiller 5.2G ... enwiki-latest-pages-articles.page000m.xml
... kmiller 2.9G ... enwiki-latest-pages-articles.page001m.xml
... kmiller 2.6G ... enwiki-latest-pages-articles.page002m.xml
... kmiller 2.4G ... enwiki-latest-pages-articles.page003m.xml
... kmiller 2.4G ... enwiki-latest-pages-articles.page004m.xml
... kmiller 2.4G ... enwiki-latest-pages-articles.page005m.xml
... kmiller 2.2G ... enwiki-latest-pages-articles.page006m.xml
... kmiller 2.4G ... enwiki-latest-pages-articles.page007m.xml
... kmiller 2.4G ... enwiki-latest-pages-articles.page008m.xml
... kmiller 2.3G ... enwiki-latest-pages-articles.page009m.xml
... kmiller 1.7G ... enwiki-latest-pages-articles.page010m.xml
```

and the first million pages contain the most images:

```
shell$ ls -lh i*m
... kmiller 653M ... image.page000m
... kmiller 300M ... image.page001m
... kmiller 248M ... image.page002m
... kmiller 219M ... image.page003m
... kmiller 199M ... image.page004m
... kmiller 170M ... image.page005m
... kmiller  49M ... image.page006m    <-- up to seg fault
... kmiller 148M ... image.page007m
... kmiller  97M ... image.page008m
... kmiller  81M ... image.page009m
... kmiller  45M ... image.page010m
```

This data inspired a more careful survey that lead to: Figure F.3, Size Distribution by Age of Article on en Wikipedia, and Figure F.4, Size Distribution by ichunk on en Wikipedia.

E.12 Experiments with Downloading Images

Download images. This is done by running the scripts generated by wikix.

```
shell$ ./image.page000m
```

This does not run smoothly, because cURL often stalls whenever the server does not serve the entire file. Hence, I must kill the job (Ctrl-C), remove the partially downloaded image file, edit image00 to remove the successfully processed bash scripts (i.e. those prior to the file that was partially downloaded), and then run the script again.

It took over a week to download image.page000m. Also, about 5-10% of the images ended up in failed.log.

To gain speed, run two in parallel (in separate terminals). For example:

```
shell$ ./image.page001m      <-- in first terminal
shell$ ./image.page002m      <-- in second terminal
```

When I replaced my i386 processor with an amd64, and installed the corresponding Debian distribution, I discovered that cURL now runs rather differently. Under the i386 version, cURL seems to ignore the --retry 7 option; while under the amd64 version, cURL respects it. Thus for each image, we now see one or two of the following messages:

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
Warning: Transient problem: FTP error Will retry in 1 seconds. 7 retries left.
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
Warning: Transient problem: FTP error Will retry in 2 seconds. 6 retries left.
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
Warning: Transient problem: FTP error Will retry in 4 seconds. 5 retries left.
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
Warning: Transient problem: FTP error Will retry in 8 seconds. 4 retries left.
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
Warning: Transient problem: FTP error Will retry in 16 seconds. 3 retries
Warning: left.
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
Warning: Transient problem: FTP error Will retry in 32 seconds. 2 retries
Warning: left.
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
Warning: Transient problem: FTP error Will retry in 64 seconds. 1 retries
Warning: left.
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
curl: (22) The requested URL returned error: 404
```

This means that cURL now takes 2-4 minutes per image, which means that importing 1.5 million images would take, say, 10 years (200 sec/image * 1.5 million images = 300 million sec = 10 years). Therefore, I edited the bash scripts to replace `--retry 7` with `--retry 1` to get a factor 64 speedup (i.e. 2 months). If one uses emacs or xemacs the command would be `M-x replace-string`.

The idea is not to download every image on the first pass, but to maximize the number of images downloaded per day. Two months later, when the next wikipedia dump becomes available, one makes another pass anyway. One should think of the image download process as akin to raking leaves—one does not get every leave on the first pass, nor does one waste time trying to do so.

E.13 Experiments with rebuildImages.php—Round 1

Image Importation. If the images are downloaded after the importation of pages, then the image table in wikidb will not be populated. Therefore, it is necessary to run the maintenance script:

```
root-shell# php /usr/share/mediawiki/maintenance/rebuildImages.php --missing
```

This needs MySQL admin access and will therefore fail for lack of permissions unless the following two lines are added to /etc/mediawiki/LocalSettings.php (or to /etc/mediawiki/AdminSettings.php).

```
$wgDBadminuser      = "root";
$wgDBadminpassword  = "*********";   (replace the asterisks, obviously)
```

The rebuildImages.php script processes 10 images/sec, so it will take about 24 hours to process each 1 million images. There are about 1.5 million images.

The script rebuildImages.php crashed a few times, with this message:

```
PHP Fatal error:  Call to a member function recordUpload() on a non-object
in /usr/share/mediawiki/maintenance/rebuildImages.php on line 196
```

This always involves an image file which name includes the percent "%" character. In each case, there is a similarly named image file without the "%", so deleting the offending image file and restarting the script is appropriate.

As mentioned above, PHP seems to have a memory leak, and occupies ever more DRAM. After about 50,000 records, rebuildImages.php gives the following error message for each image record inserted:

```
PHP Warning:  proc_open(): fork failed - Cannot allocate memory
in /usr/share/mediawiki/includes/parser/Tidy.php on line 101
```

It is best to kill the rebuildImage.php process with CTRL-C, which will free up over 1GB DRAM, and then restart it.

E.14 Experiments with Corrupt Images

Sometimes cURL will download only part of the image file. This can occur when the download is broken off by web server; and cURL emits the message

```
curl: (18) transfer closed with 746274 bytes remaining to read
```

When this happens, the script generated by wikix will store the corrupt file anyway. This later becomes a nuisance when the pages are imported, and gm convert generates error messages like

```
/usr/bin/gm convert: Corrupt JPEG data: premature end of data segment
 (/var/lib/mediawiki/images/0/0c/Hurricane_Dennis_10_july_2005_1615Z.jpg).
```

One way to deal with the situation is to preemptively search and destroy the corrupt image files. This is done with gm identify like so

```
root-shell# cd /var/lib/mediawiki/images/0/0c/
root-shell# gm identify -verbose Hurricane_Dennis_10_july_2005_1615.jpg | grep identify
/usr/bin/gm identify: Corrupt JPEG data: premature end of data segment
 (/var/lib/mediawiki/images/0/0c/Hurricane_Dennis_10_july_2005_1615Z.jpg).
```

One must write a maintenance script to iterate over all 1.5 million image files. Do not try gm identify -verbose * as that will just heavily load your CPUs for a minute, then crash glibc and produce a backtrace.

The ones identified as corrupt, can be downloaded again using cURL with perhaps greater success.

MediaWiki offers two utilities: one for dealing with bad image file names, cleanupImages.php; and one for corrupt image files, checkImages.php:

1) cleanupImages.php. Improperly named image files (e.g. file names containing the character "%") can be caught by running:

```
root-shell# php /usr/share/mediawiki/maintenance/cleanupImages.php
root-shell# php /usr/share/mediawiki/maintenance/cleanupImages.php --fix
```

This reads the image database table at about 2400 rows/sec. It found no bad filenames. This cannot be right (unless I had already deleted all such files manually—which is possible).

2) checkImages.php. Corrupt images (e.g. truncated due to faulty download) can be caught by running:

```
root-shell# php /usr/share/mediawiki/maintenance/checkImages.php
```

This reads the image database table at about 25 rows/sec. I let it run for two hours, yet it found nothing. This cannot be right, because importDump.php calls gm convert which reports many corrupt images.

Neither cleanupImages.php nor checkImages.php seem to get the job done.

I will have to write my own script.

E.15 Experiments with the objectcache

Sometimes importDump.php appears to stop, with top showing no activity for PHP, and significant activity for MySQL. This is usually because InnoDB sometimes stops taking queries to concentrate upon flushing tens of thousands of dirty pages from the InnoDB buffer pool to disk.

Occasionally, the delay is hours long. Investigation revealed that MediaWiki sometimes submits a query to delete a million or so rows from the objectcache database table. For example,

```
mysql> SHOW ENGINE INNODB STATUS\G
...
DELETE /* mediawikiBagOStuff::_doquery 127.0.0.1 */ FROM
‘objectcache‘ WHERE exptime < ’20110517051938’
...
```

In this case, InnoDB shows high levels of read and delete

```
mysql> SHOW ENGINE INNODB STATUS\G
...
Number of rows inserted 9536, updated 5830, deleted 1190986, read 1242549
0.00 inserts/s, 0.00 updates/s, 774.23 deletes/s, 774.23 reads/s
...
```

and the objectcache table shows millions of rows

```
mysql> SHOW TABLE STATUS LIKE ’objectcache’\G
...
Rows: 2390302
...
```

Even after the deleting stops, InnoDB will still need an hour to merge all the underfilled InnoDB pages. update.php also deletes the cache.

```
root-shell# php /usr/share/mediawiki/maintenance/update.php
```

Recommendation: The best way to deal with this situation is to preemptively deplete the cache by deleting a thousand rows after processing each file. WP-MIRROR does this whenever the objectcache exceeds a threshold of 10,000 rows.

E.16 Experiments with rebuildImages.php—Round 2

InnoDB can automatically recognize when records are being inserted in LINEAR or in RANDOM order; and InnoDB has algorithms for each case. So, it is worth looking at the image table in wikidb, and the insertion process.

```
mysql> SHOW CREATE TABLE image\G
...
Create Table: CREATE TABLE `image` (
  `img_name` varbinary(255) NOT NULL DEFAULT '',
...
  PRIMARY KEY (`img_name`),
...
) ENGINE=InnoDB DEFAULT CHARSET=binary
```

So the image table records are stored in alphabetical order of img_name. Now the rebuildImages.php inserts image records in the order that it finds them on disk, which is unlikely to be alphabetical since image.page000m, etc. downloads them and writes them to disk in the order that wikix finds them in enwiki-yyyymmdd-pages-articles.xml.

Therefore, the image records are inserted in RANDOM order. Since the average image record length is 840B,

```
mysql> SHOW TABLE STATUS LIKE 'image'\G
...
Avg_row_length: 840
...
```

and since randomly filled InnoDB database pages are on average a bit over half full, each 16KB page will hold about 10 records. (Note: when a database page is over 15/16 full, InnoDB splits it into two half-full database pages.) Thus to insert one image record, InnoDB must first read a page containing say 10 records and cache it in the buffer pool (in DRAM). And, to insert 2 million image records, will require InnoDB to read (or to create by splitting) over 200k database pages into the buffer pool, and later flush the modified database pages (first to the log file, then the double-write buffer, and finally to the image table stored in InnoDB table space). This causes much more disk I/O than if the records had been inserted sequentially (LINEAR).

If the buffer pool is too small, then the same database page will have to be repeatedly read, modified, flushed when it becomes the least recently used (LRU algorithm), and read again (thrashing). A large buffer pool that can hold the entire image table greatly speeds the process. 4GB will do nicely. 3GB (2998M/16K = 187k pages = 1.8 million rows in the image table).

Experiment: Insert an additional 200k images.

After inserting 100k images, the buffer pool is about half occupied. The buffer pool hit rate is perfect (1000/1000) which means that the entire image table is sitting in the buffer pool (which in our case is 3GB). In the buffer pool, less than half of the pages are marked modified, which is because modified pages are being flushed to disk and marked clean:

```
mysql> SHOW ENGINE innodb STATUS\G
...
----------------------
BUFFER POOL AND MEMORY
----------------------
Total memory allocated 348629496; in additional pool allocated 1082624
Dictionary memory allocated 333200
Buffer pool size   192000
Free buffers       109397
Database pages     81887                                <-- database pages
Modified db pages  35426
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages read 37501, created 44386, written 622566        <-- read + created
0.00 reads/s, 0.81 creates/s, 6.26 writes/s
Buffer pool hit rate 1000 / 1000                       <-- hit rate
...
```

Table E.5: Processes that Cause Resource Contention

Task	Process	Consumes	Competes with	Daemon	Cron
Image download	cURL	network I/O	apt-mirror, bittorrent.		daily
Page importation	InnoDB	disk I/O	mdadm/checkarray, nepomuk,		weekly
			strigi,	Yes	
			updatedb.mlocat.		daily
	php, gm	cpu	plasma-desktop with compositing.	Yes	

Since the image table fits entirely into the buffer pool, 'pages read' plus 'pages created' (37501+44386=81887) equals the total pages in the buffer pool.

Note, however, that pages written (622566) is much greater, apparently, one flush per COMMIT. In other words, the Built-in InnoDB GROUP COMMIT feature is not working. Disk I/O is therefore far higher than anticipated.

Recommendation: Use at least 3G DRAM. Use InnoDB Plug-in rather than Built-in InnoDB.

E.17 Experiments with Resource Contention

Some processes cause resource contention that slows down the image download and page importation process. Such process should be suspended, killed, or deinstalled. See Table E.5, Processes that Cause Resource Contention.

The daily apt-mirror job usually goes quickly, although one should expect to download hundreds of megabytes per day during the months leading up to the announcement of the next stable release. In which case, the image download process will suffer for an hour or two each night.

Running bittorrent while downloading image files, results in many partial downloads (corrupt image files). Recommendation: do not use.

If you store your database and images on a RAID array (and you should), then mdadm/checkarray will run weekly for about one day. During that day, disk performance will be reduced. Just let it happen.

nepomuk collects metadata from all the files on the system. It impairs disk performance. (And do you really need it?)

The plasma-desktop is surprisingly demanding of the CPU. This is especially true if compositing is enabled (default). Compositing means that windows are rendered as translucent, with any underlying window images, and with the underlying desktop wallpaper showing through. Only the window with the focus is opaque. To produce this effect, window images are alpha-blended with the desktop wallpaper using, OpenGL or Xrender. This is attractive, and it lets the user see all his overlapping and hidden windows at a glance, without moving or restacking them. However, importDump.php is especially impaired, not only because it is written in php, but because it invokes gm to perform the image resizing (to make thumbs). Recommendation: consider suspending desktop effects.

strigi is a daemon for indexing and searching your personal data. It hammers the system (both disk and CPU). Among other things, it calculates the sha1sum for every file on your system—including the 1.5 million image files you just downloaded. (And do you really need it?) Recommendation: deinstall.

mlocate is another file indexing and searching utility. The daily updatedb.mlocate job reduces the page importation rate by half for several hours (but would complete in less than an hour on an unloaded system). It also slows down desktop response while you are working. If it wakes up while you are working, execute:

```
root-shell# ps -efw | grep mloc
root      1572   799  0 07:49 ?       00:00:00 /bin/bash /etc/cron.daily/mlocate
root      1579  1572  2 07:49 ?       00:00:29 /usr/bin/updatedb.mlocate
root      1660  7742  0 08:09 pts/45  00:00:00 grep mloc
root-shell# kill -15 1579
```

or schedule the cron job for a time when you will not be working. Also be sure that at most one of the two packages, `mlocate` and `locate`, are installed.

E.18 Upgrade from Debian Lenny to Squeeze

The upgrade from Debian lenny to squeeze entailed an update of `MediaWiki`. That the updated `MediaWiki` has a different database schema, was learned when the web browser gave errors like the following:

```
A database query syntax error has occurred. This may indicate a bug in
the software. The last attempted database query was:
  (SQL query hidden)
from within function "OutputPage::addCategoryLinks". Database returned
error "1146: Table 'wikidb.page_props' doesn't exist (localhost)".
```

and

```
A database query syntax error has occurred. This may indicate a bug in
the software. The last attempted database query was:
  (SQL query hidden)
from within function "Article::updateCategoryCounts". Database returned
error "1146: Table 'wikidb.category' doesn't exist (localhost)".
```

The database was updated by running

```
root-shell# php /usr/share/mediawiki/maintenance/update.php
```

However, `update.php` took a long time to complete. While the missing tables were created quickly (which dealt with the web errors); populating the `wikidb.category` table took an hour; populating the `rev_parent_id` field took several more.

E.18.1 Fiasco with `pagelinks`

After six hours, I canceled `update.php`, and then inadvertently dropped the `pl_from` index for the `pagelinks` table by giving the command:

```
ALTER TABLE `pagelinks`
DROP INDEX pl_from
ADD INDEX pl_namespace(pl_namespace, pl_title, pl_from);
```

This foolish command took over 10 hours.

The intended `pagelinks` table, had I let `update.php` run to completion, would have been:

```
mysql> SHOW CREATE TABLE PAGELINKS\G
...
Create Table: CREATE TABLE `pagelinks` (
  `pl_from` int(10) unsigned NOT NULL DEFAULT '0',
  `pl_namespace` int(11) NOT NULL DEFAULT '0',
  `pl_title` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
  UNIQUE KEY `pl_from` (`pl_from`,`pl_namespace`,`pl_title`)
  UNIQUE KEY `pl_namespace` (`pl_namespace`,`pl_title`,`pl_from`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

What remained to do, aside from fixing the mistake, was to make pl_namespace, tl_namespace, and il_to indices UNIQUE (see /usr/share/mediawiki/maintenance/archives/patch-pl-tl-il-uniq for the following code):

```
DROP INDEX /*i*/pl_namespace ON /*_*/pagelinks;
CREATE UNIQUE INDEX /*i*/pl_namespace ON /*_*/pagelinks (pl_namespace, pl_title, pl_from);
DROP INDEX /*i*/tl_namespace ON /*_*/templatelinks;
CREATE UNIQUE INDEX /*i*/tl_namespace ON /*_*/templatelinks (tl_namespace, tl_title, tl_from);
DROP INDEX /*i*/il_to ON /*_*/imagelinks;
CREATE UNIQUE INDEX /*i*/il_to ON /*_*/imagelinks (il_to, il_from);
```

To obtain count and timing information, I decided to run these commands by hand: first the three DROP commands, and then the CREATE commands in the order of increasing count:

TABLE	COUNT(*)	TIME DROP INDEX	Time CREATE UNIQUE INDEX
imagelinks	1817324	1 min 38.21 sec	3 min 10.50 sec
templatelinks	29108059	29 min 19.64 sec	32 min 20.52 sec
pagelinks	148496964	< 10 hours	> 1 week

Originally I should have let update.php run to completion, because killing it left pagelinks without the pl_namespace index.

Making pl_namespace UNIQUE proved to be a BAD IDEA(tm). Each time I inserted a new pagelinks record (with importDump.php), InnoDB took over two hours to establish row locks on every existing row (all 150 million). This can be seen by running:

```
mysql> SHOW ENGINE INNODB STATUS\G
...
------------
TRANSACTIONS
------------
Trx id counter 0 35221413
Purge done for trx's n:o < 0 35221411 undo n:o < 0 0
History list length 0
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 0 0, not started, process no 6518, OS thread id 139948387165952
MySQL thread id 75, query id 73206 localhost root
---TRANSACTION 0 0, not started, process no 6518, OS thread id 139948386764544
MySQL thread id 74, query id 73209 localhost root
show engine innodb status
---TRANSACTION 0 35221411, ACTIVE 1983 sec, process no 6518, OS thread id 139948390319872 fetching
mysql tables in use 1, locked 1
222735 lock struct(s), heap size 25081840, 42555962 row lock(s), undo log entries 5
MySQL thread id 70, query id 73203 localhost root Sending data
SELECT /* LinksUpdate::getExistingLinks 127.0.0.1 */  pl_namespace,pl_title  FROM `pagelinks`  WHER
Trx read view will not see trx with id >= 0 35221412, sees < 0 35221412
--------
```

Note that the last transaction has established over 42 million row locks and will continue until it has established all 150 million. This is a query submitted by importDump.php presumeably to enforce uniqueness. When I loaded more pages with importDump.php, mysqld slowed to a crawl (about 10,000x slower!).

Next we restored the pl_from index with:

```
mysql> CREATE UNIQUE INDEX pl_from ON pagelinks (pl_from, pl_namespace, pl_title);
```

The next night, there were two power outages. MySQL failed to restart due to a bad sector on the HDD, which was in the InnoDB double-write buffer. So I repaired the sector (write zeros on it), and rolled back the transaction in the double-write buffer, like so:

```
root-shell# hdparm -W0 /dev/sdc                        <-- should do every boot for InnoDB
root-shell# smartctl -a /dev/sdc | less                <-- LBA 134696376 failed to read
root-shell# hdparm --read-sector 134696376 /dev/sdc    <-- failed to read
root-shell# hdparm --repair-sector 134696376 --yes-i-know-what-i-am-doing /dev/sdc
root-shell# /etc/init.d/mysql restart
root-shell# smartctl -a /dev/sdc | less                <-- cleared error, but no remap
```

InnoDB recommends disabling write caching:

```
root-shell# emacs /etc/hdparm.conf
/dev/sdc {
       write_cache = off
}
```

If that fails then try:

```
root-shell# emacs /etc/rc.local
hdparm -W0 /dev/sdc
```

E.18.2 Post-Mortem Analysis and Lessons Learned

Post-mortem analysis:

1. Trying to create UNIQUE indices for pagelinks caused weeks of heavy disk I/O, mostly with the double-write buffer (which occupies about 100 InnoDB pages of 16K each). This probably caused the sectors to wear down to the point that they became unreadable.
2. According to syslog files, the ib_logfile[01] filled up, and the whole transaction had to be rolled back.
3. During index creation, all write access to the pagelinks table was blocked and timed out, which consequently blocked insertion of additional pages.
4. During index creation, files are written to tmpdir (/tmp) which in my case is only 1000M, which is too small to hold the largest column pl_title of the pagelinks table (table is 10G).
5. Since pl_title used UTF8 encoding, InnoDB tried to rebuild the entire table in a temporary table (MySQL Bug #39833). Consider BINARY encoding.
6. The Built-in InnoDB rebuilds the table in a temporary table, and when complete, drops the original table and renames the temporary table. The copy process, which had to establish uniqueness as each row was added to the temporary table, turned out to be an enormous task for such a large table.
7. Crash recovery turns out to be non-trivial for index creation. Secondary indices are simply dropped. Cluster indices leave behind a temporary table, but there is no guidance on how to continue the index creation. In either case, it is simplest to start over.

We learn the following lessons:

1. The pagelinks table of 10G is far too large to attempt index creation when the /tmp partition is 1G and buffer-pool is 3G.
2. Better to TRUNCATE the imagelinks table, run update.php, finish inserting pages, TRUNCATE the imagelinks table again, then download the latest imagelinks dump file, and insert the dump.
3. I should have enabled InnoDB Plugin instead of Built-in InnoDB, which would have allowed much faster index creation, as well as, change buffering for inserts.

E.18.3 Starting Over

Date: 2011-Mar-01

Due to the pagelinks index fiasco, I made the decision to:

1. DROP the wikidb database (but keep all the images),
2. upgrade mysql-server (squeeze uses MySQL 5.1, lenny uses MySQL 5.0),
3. enable InnoDB Plugin instead of the Built-in InnoDB,
   - faster locking for multi-core processor,
   - fast index creation,
   - change buffer for insertions,
   - group commit works,
   - adaptive flushing of database pages,
4. upgrade MediaWiki (squeeze has many updates over lenny),
   - run update.php (no apparent memory leak) on empty wikidb,
5. download the latest enwiki-latest-pages-articles.xml.bz2, and
6. reinsert all the wikidb records.

Reinsertion worked as follows:

E.18.3.1 DUMP

Download latest wikipedia dump (time: a few hours):

```
shell$ wget http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2
shell$ bunzip2 enwiki-latest-pages-articles.xml.bz2
shell$ cat enwiki-latest-pages-articles.xml | grep "<page>" | wc -l
```

Note: The latest dump, as of 2011-03-08, is dated 2011-01-17 and has 10.861 million pages.

Dump date	Pages [million]
2010-09-16	10.355
2011-01-17	10.861
2011-09-01	11.577
2011-10-07	11.687

E.18.3.2 SPLIT

Split dump into conveniently sized files (1 million pages per file for image processing; 50 thousand pages per file for page insertion), and generate the image download scripts (time: a few hours):

```
shell$ ./split_page_v0.3.lisp --count=1000000 enwiki-latest-pages-articles.xml
shell$ ./split_page_v0.3.lisp --count=50000 enwiki-latest-pages-articles.xml
shell$ wikix < enwiki-latest-pages-articles.page000m.xml > image.page000m
shell$ wikix < enwiki-latest-pages-articles.page001m.xml > image.page001m
...
shell$ chmod 755 image.page*
```

There is still one page in enwiki...page006m that causes wikix to crash. A work-around is to use the 50k split files (enwiki...page006m300k crashes, so image.page006m300k ends with bad syntax):

```
shell$ wikix < enwiki-latest-pages-articles.page006m000k.xml > image.page006m000k
shell$ wikix < enwiki-latest-pages-articles.page006m050k.xml > image.page006m050k
...
shell$ wikix < enwiki-latest-pages-articles.page006m950k.xml > image.page006m950k
```

The first million pages produce a script (image.page000m is 653M) that is unwieldy. So, it too is split using 50k files:

```
shell$ wikix < enwiki-latest-pages-articles.page000m000k.xml > image.page000m000k
shell$ wikix < enwiki-latest-pages-articles.page000m050k.xml > image.page000m050k
...
shell$ wikix < enwiki-latest-pages-articles.page000m950k.xml > image.page000m950k
```

E.18.3.3 IMAGE (existing)

Since we have over 1 million images, let us insert the image table records first, so that we can see images right away during page table insertion below (time: one week):

```
root-shell# php /usr/share/mediawiki/maintenance/rebuildImages.php --missing
```

Due to apparent memory leak, this job had to be killed every 50,000 images and restarted (e.g. process 0/00 to 0/07, kill, then process 0/08 to 0/0f).

E.18.3.4 IMAGE (new)

Download any new images, then insert into image table (time: 3-4 weeks):

```
root-shell# cp image.page* /database0/images/
```

Since darkstar-5 has no internet connection, I use darkstar-4:

```
root-shell# ssh darkstar-4
darkstar-4:#  sshfs darkstar-5:/database0 /database0
darkstar-4:#  cd /database0/images/
darkstar-4:#  ./image.page000m
darkstar-4:#  ./image.page001m
darkstar-4:#  ...
darkstar-4:#  fusermount -u /database0
darkstar-4:#  exit
root-shell#
```

Then I return to darkstar-5 and insert any new images into wikidb:

```
root-shell# php /usr/share/mediawiki/maintenance/rebuildImages.php --missing
```

E.18.3.5 PAGE

Insert page table records 50,000 at a time, due to apparent memory leak (time: two months):

```
root-shell# php /usr/share/mediawiki/maintenance/importDump.php <
 enwiki-latest-pages-articles.page000m000k.xml
root-shell# php /usr/share/mediawiki/maintenance/importDump.php <
 enwiki-latest-pages-articles.page000m050k.xml
...
```

Actually, I prefer commands that can run for 20-24 hours (or even 2-3 days if I am away for the weekend), for example:

```
root-shell# php /usr/share/mediawiki/maintenance/importDump.php < \
latest/enwiki-latest-pages-articles.page000m000k.xml ; sleep 30m ; php \
 /usr/share/mediawiki/maintenance/importDump.php <  \
latest/enwiki-latest-pages-articles.page000m050k.xml ; sleep 30m ; php \
 /usr/share/mediawiki/maintenance/importDump.php <  \
latest/enwiki-latest-pages-articles.page000m100k.xml ; sleep 30m ; php \
 /usr/share/mediawiki/maintenance/importDump.php <  \
latest/enwiki-latest-pages-articles.page000m150k.xml
root-shell# rm latest/enwiki-latest-pages-articles.page000m[01]*xml
```

E.18.3.6 InnoDB Flushing

The sleep 30m commands gives InnoDB time to flush its buffer pool between each batch of inserts. This actually yeilds an overall speed increase (about 2x). Flushing the buffer pool usually takes about 20m:

```
shell$ dc
192000 0.75/p
144000
133/p
1082
60/pq
18
shell$
```

The estimation of time required to flush the buffer pool goes as follows: The InnoDB Plugin default innodb_max_dirty_pages_pct is 75. So

192,000 buffer pool size * 75% = 144,000 max modified pages

144,000 modified pages / 133 pages flushed per s = 1082s = 18m.

Actually, this may be an underestimate, because InnoDB reads in tens of thousands of pages during the page flushing process; perhaps in order to merge underfilled pages and split overfilled pages. When, during the flushing, we look at the engine status, we can see a great increase in the number of 'merged recs' and 'merges'.

```
mysql> SHOW ENGINE INNODB STATUS\G
...
-------------------------------------
INSERT BUFFER AND ADAPTIVE HASH INDEX
-------------------------------------
Ibuf: size 937, free list len 1475, seg size 2413,
952695 inserts, 824292 merged recs, 440497 merges    <-- increase during flush
Hash table size 6222817, node heap has 3850 buffer(s)
0.00 hash searches/s, 174.10 non-hash searches/s
...
```

E.18.3.7 Pipelining

To save time, I pipelined the IMAGE(new) and PAGE steps:

Timeslot	Image Download	rebuildImages.php	importDump.php
000	image.page000m		
001		image.page000m	
002	image.page001m		enwiki...page000m000k.xml enwiki...page000m050k.xml ...
003		image.page001m	
004	image.page002m		enwiki...page001m000k.xml enwiki...page001m050k.xml enwiki... ...
005		image.page002m	
006	image.page003m		enwiki...page002m000k.xml enwiki...page002m050k.xml ...

Note: Simultaneously running rebuildImages.php and importDump.php can cause InnoDB to deadlock, which in turn causes one or both scripts to abort, leaving an error message identifying the offending SQL statement.

```
mysql> SHOW ENGINE INNODB STATUS\G
```

also describes the deadlock in detail, identifying the SQL statements involved, and which one was rolled back to break the deadlock.

Note: There appears to be no conflict between running the image download scripts and the other two processes.

E.18.3.8 InnoDB Deleting

Sometimes importDump.php appears to just stop, with top showing no activity for php, and significant activity for mysql. This is usually because InnoDB sometimes stops taking queries to concentrate upon flushing tens of thousands of dirty pages to disk. The above mentioned sleep 30m is meant for that case.

Occasionally, however, the delay is hours long! Investigation revealed MediaWiki occasionally submits a query that must delete a million or so rows, such as this one:

```
DELETE /* mediawikiBagOStuff::_doquery 127.0.0.1 */ FROM `objectcache`
WHERE exptime < '20110517051938'
```

In this particular case, InnoDB showed high levels of read and delete:

```
mysql> SHOW ENGINE INNODB STATUS\G
...
Number of rows inserted 9536, updated 5830, deleted 1190986, read 1242549
0.00 inserts/s, 0.00 updates/s, 774.23 deletes/s, 774.23 reads/s
...
```

and the objectcache table showed millions of rows:

```
mysql> SHOW TABLE STATUS LIKE 'objectcache'\G
...
Rows: 2390302
...
```

Even after the deleting stops, InnoDB will still need an hour to merge all the underfilled 'database pages'.

The one way to deal with this situation is to preemptively delete the cache by running update.php:

```
root-shell# php /usr/share/mediawiki/maintenance/update.php
```

E.19 Messages

E.19.1 /bin/bash

Some image filenames confuse sh, even though they are in quotes. Control characters (ampersand, asterisk, backquote, brackets, braces, etc.) must be escaped. In some cases (file name contains a ampersand, percent) WP-MIRROR will not download the file at all.

Many file-names use Unicode. Most utilities are now able to handle Unicode.

E.19.2 Filename Issues

If you use polipo (a caching web proxy), then thousands of image file requests will be redirected. File names containing an apostrophe or a quote, seem to be the only ones effected.

```
root-shell# cat /var/lib/mediawiki/images/0/00/Dalida'.jpg
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<title>Proxy result: 302 Redirected by external redirector.</title>
</head><body>
<h1>302 Redirected by external redirector</h1>
<p>The following status was returned:<br><br>
<strong>302 Redirected by external redirector</strong></p>
<hr>Generated Sat, 05 Nov 2011 18:32:23 EDT by Polipo on
 <em>darkstar-7:8123</em>.
</body></html>
```

We can identify and remove these files after downloading.

```
root-shell# cd /var/lib/mediawiki/images/0/00/
root-shell# gm identify -verbose Chief_Scout's_Bronze_Award_(The_Scout_Association).png
gm identify: Improper image header (Chief_Scout's_Bronze_Award_(The_Scout_Association).png).
root-shell# gm identify -verbose Dalida'.jpg
gm identify: Not a JPEG file: starts with 0x3c 0x21 (Dalida'.jpg).
root-shell# gm identify -verbose Rock_'N_Roll_Loony_Party_-The-_1.gif
gm identify: Improper image header (Rock_'N_Roll_Loony_Party_-The-_1.gif).
root-shell# gm identify -verbose Transistor_Count_and_Moore's_Law_-_2011.svg
gm identify: Opening and ending tag mismatch: br line 0 and p.
```

Note that error messages are different for each image file format.

E.19.3 gm convert (GraphicsMagick)

gm convert warns of image file corruption

```
/usr/bin/gm convert: Corrupt image (/var/lib/mediawiki/images/5/5e/
Swainsley_Bridge.gif).

/usr/bin/gm convert: Corrupt JPEG data: 190 extraneous bytes before
marker 0xd9 (/var/lib/mediawiki/images/4/45/Silveira.jpg).

/usr/bin/gm convert: Corrupt JPEG data: bad Huffman code (/var/lib/
mediawiki/images/a/ad/CarolynSGriner3.jpg)

/usr/bin/gm convert: Corrupt JPEG data: found marker 0xd9 instead of
RST4 (/var/lib/mediawiki/images/1/11/Union_Station_Louisville.JPG).
```

```
/usr/bin/gm convert: Corrupt JPEG data: found marker 0xd9 instead of
RST6 (/var/lib/mediawiki/images/c/c5/BrownUniversity-JohnDRockefelle
rJrLibrary.jpg).

/usr/bin/gm convert: Corrupt JPEG data: premature end of data segme
nt (/var/lib/mediawiki/images/0/0c/Hurricane_Dennis_10_july_2005_16
15Z.jpg).

/usr/bin/gm convert: Ignoring attempt to set cHRM RGB triangle with
zero area (/var/lib/mediawiki/images/8/8a/DYK_star.png).

/usr/bin/gm convert: Ignoring gAMA chunk with gamma=0 (/var/lib/med
iawiki/images/6/62/TF3D_Portrait.png).
```

```
/usr/bin/gm convert: Incorrect sRGB chunk length (/var/lib/mediawik
i/images/f/f7/Bunny_Breckinridge_(cropped).png).

/usr/bin/gm convert: Invalid JPEG file structure: missing SOS marke
r (/var/lib/mediawiki/images/9/9b/Vw-baunatal-brand.jpg).

/usr/bin/gm convert: Invalid SOS parameters for sequential JPEG (/v
ar/lib/mediawiki/images/e/e2/Vf142ghost.jpg).

/usr/bin/gm convert: Missing PLTE before bKGD (/var/lib/mediawiki/i
mages/d/d4/SpartaDOS_X_menu.png).
```

```
/usr/bin/gm convert: Premature end of JPEG file (/var/lib/mediawiki
/images/f/fe/Bush-Howard_2001_review.jpg).

/usr/bin/gm convert: Profile size field missing from iCCP chunk (/v
ar/lib/mediawiki/images/1/13/Couturier.png).

/usr/bin/gm convert: Read Exception (/var/lib/mediawiki/images/2/2b
/J.F._de_la_Cerda_por_Claudio_Coello_01.png).

/usr/bin/gm convert: tRNS chunk has out-of-range samples for bit_de
pth (/var/lib/mediawiki/images/7/76/Communes_of_Luxembourg.png).

/usr/bin/gm convert: Warning: unknown JFIF revision number 2.01 (/v
ar/lib/mediawiki/images/c/c0/MASINT-AF-Helo-AN-TPQ-36.jpg).
```

E.19.4 convert (ImageMagick)

This program overcommits DRAM, causing the kernel to randomly kill other processes, eventually hanging the system.

Recommendation: Do not use.

E.19.5 graphicsmagick-imagemagick-compat

This package is supposed to replace /usr/bin/convert with a symbolic link to gm. However, experiments show that this approach results in a surprising number of segmentation faults. Worse, convert frequently hung without crashing, thereby stopping the importation process.

```
sh: line 1: 15126 Segmentation fault      convert -background white -thumbnail
 250x198 '/var/lib/mediawiki/images/f/f6/Aphaenogaster.lepida.-.wheeler.svg'
 PNG:'/var/lib/mediawiki/images/thumb/f/f6/Aphaenogaster.lepida.-.wheeler.svg/
250px-Aphaenogaster.lepida.-.wheeler.svg.png' 2>&1
```

A few hundred pages thereafter, convert crashed with the following error message.

```
*** glibc detected *** convert: double free or corruption (out): 0x00007f313150e010 ***
======= Backtrace: =========
/lib/libc.so.6(+0x71ad6)[0x7f3132ac1ad6]
/lib/libc.so.6(cfree+0x6c)[0x7f3132ac684c]
/usr/lib/libGraphicsMagick.so.3(DestroyCacheInfo+0xd1)[0x7f3135a31901]
/usr/lib/libGraphicsMagick.so.3(DestroyImagePixels+0x28)[0x7f3135a31a88]
/usr/lib/libGraphicsMagick.so.3(DestroyImage+0x76)[0x7f3135a1f5d6]
/usr/lib/libGraphicsMagick.so.3(MogrifyImage+0x14b5)[0x7f31359b38c5]
/usr/lib/libGraphicsMagick.so.3(MogrifyImages+0x147)[0x7f31359b6947]
/usr/lib/libGraphicsMagick.so.3(ConvertImageCommand+0xaba)[0x7f31359cc07a]
/usr/lib/libGraphicsMagick.so.3(MagickCommand+0x161)[0x7f31359c4e91]
/usr/lib/libGraphicsMagick.so.3(GMCommand+0xbc)[0x7f31359c4fbc]
/lib/libc.so.6(__libc_start_main+0xfd)[0x7f3132a6ec4d]
convert(+0x7f9)[0x7f31360777f9]
======= Memory map: ========
7f312c000000-7f312c024000 rw-p 00000000 00:00 0
7f312c024000-7f3130000000 ---p 00000000 00:00 0
...
```

convert frequently hung without crashing, thereby stopping the importation process. This could be seen by listing any image conversion processes

```
shell$ ps -wef | grep onv
```

Perhaps this is because /usr/bin/convert is a soft link to gm, so convert <params> becomes gm <params> instead of gm convert <params>.

Recommendation: Do not use.

E.19.6 dvips

On rare occasion, dvips can emit an error message:

```
dvips: Can't make it EPSF, sorry
Warning: no %%Page comments generated.
```

E.19.7 PHP Notice: Undefined index:

I got bursts of 'PHP Notice: Undefined index' messages. These arise when PHP reads an array and expects to find a certain index that turns out not to exist. In GlobalFunctions.php the offending code is trying to create an URL for email.

```
function wfParseUrl( $url )

PHP Notice:  Undefined index: scheme in /usr/share/mediawiki/includ
es/GlobalFunctions.php on line 2423

PHP Notice:  Undefined index: scheme in /usr/share/mediawiki/includ
es/GlobalFunctions.php on line 2425

function wfMakeUrlIndex( $url )

PHP Notice:  Undefined index: host in /usr/share/mediawiki/includes
/GlobalFunctions.php on line 2449
```

Fckeditor is a 'rich text format javascript web editor'. Debian has two packages fckeditor and mediawiki-extensions-fckeditor.

```
PHP Notice:  Undefined index: HTTP_USER_AGENT in /usr/share/fckedit
or/fckeditor_php5.php on line 37
```

You probably do not want to edit any pages of your mirror of wikipedia. These errors appear in /var/log/apache2/. They can be reduced by making sure your browsers (including cURL and wget) provide the user-agent field.

Update: For WP-MIRROR 0.4, the problem had gone away. As of MediaWiki 1.18, the Fckeditor extension is obsolete. According to the Wikimedia Foundation:

> This extension is no longer supported by the upstream developers.
> http://www.mediawiki.org/wiki/Extension:FCKeditor_(Official), accessed 2012-12-19

Recommendation: Do not use.

E.19.8 PHP Notice: xml_parse()

Some 50,000 files gave tens of thousands of warnings like these:

```
PHP Warning:  xml_parse(): Unable to call handler in_() in /usr/sha
re/mediawiki/includes/Import.php on line 437

PHP Warning:  xml_parse(): Unable to call handler out_() in /usr/sh
are/mediawiki/includes/Import.php on line 437
```

Importation of some 3 million pages was impeded by the above error. This is a bug in Import.php (1.15). To fix, download a patched version from
http:svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/includes/Import.php?view=co
and replace the original file in the /includes/ directory.

Update: For WP-MIRROR 0.4, the problem has gone away. MediaWiki 1.19 does not need the patch.

E.19.9 PHP Warning

I have seen this warning a couple dozen times, once as a blast of 40 or so messages.

```
PHP Warning:  preg_replace_callback(): Unknown modifier 'c' in /us
r/share/php-geshi/geshi.php on line 3300
```

GeSHi stands for Generic Syntax Highlighter. Debian packages are `php-geshi` and `mediawiki-extensions-geshi`.

Appendix F

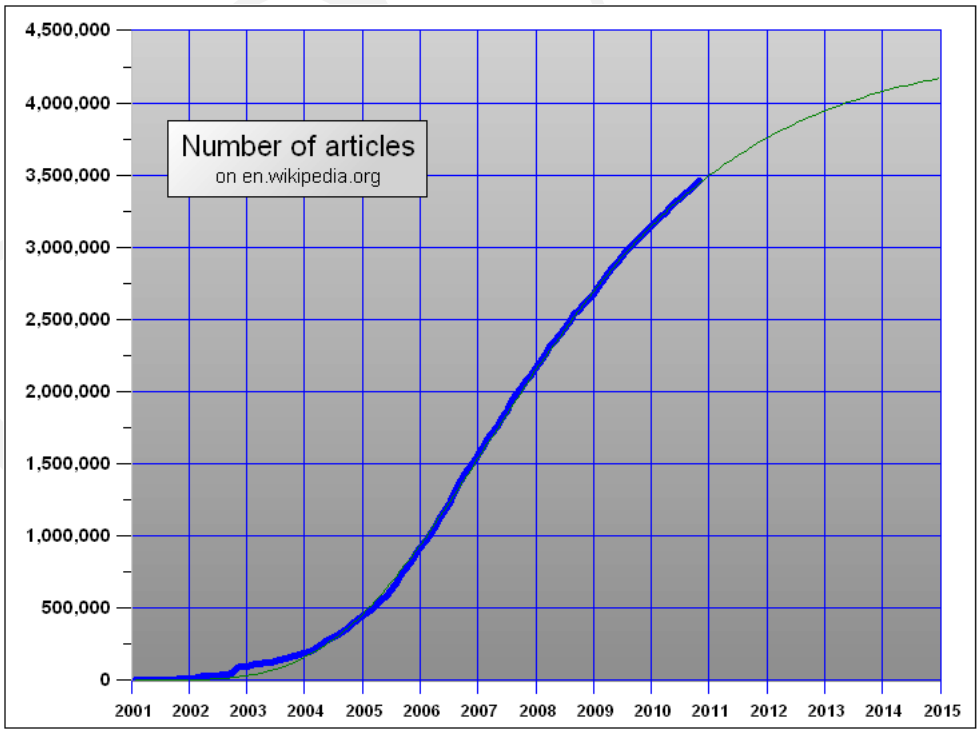
Trends (2011-Dec-21)

On 2011-Dec-21, the author made a presentation at the DC Linux Users Group days before the release of WP-MIRROR 0.2 on 2011-Dec-25. The slides, in PDF format, are posted at http://dclug.tux.org/. The tables and figures in this chapter are drawn from that presentation. This is also true of Chapter E, Experiments (Autumn 2010—Spring 2011).

F.1 History

We can get an estimate of the future growth and ultimate size of the en wikipedia. See Figure F.1, Number of Articles on en Wikipedia.

Figure F.1: Number of Articles on en Wikipedia



Source: http://en.wikipedia.org/wiki/History_of_Wikipedia.

Table F.1: Size Distribution by Language

No	Language	Wiki	Articles	Images	Comment
1	English	en	3,806,175	825,559	3T
2	German	de	1,318,393	175,658	
3	French	fr	1,176,662	44,516	
4	Dutch	nl	868,843	18	
5	Italian	it	863,119	96,656	
6	Polish	pl	845,685	2	
7	Spanish	es	845,675	0	
8	Russian	ru	793,753	124,541	
9	Japanese	ja	779,274	77,097	
10	Portugese	pt	705,058	12,920	
44	Simple English	simple	75,489	38	60G
50	Latin	la	61,065	1	
110	Yiddish	yi	9,094	1,434	
249	Zulu	zu	256	0	
275	Choctaw	cho	15	2	

Source: http://meta.wikimedia.org/wiki/List_of_Wikipedias on 2011-11-24.

F.2 Main Components

MediaWiki needs a LAMP stack (Linux, Apache, MySQL, PHP). MediaWiki is written in PHP. MediaWiki and all components of the LAMP stack are available in GNU/Linux distributions (e.g. Debian). The database management system (DBMS) used by the Wikimedia Foundation is MySQL; but, postgres, SQLite, MSsql, and IBM_DB2 are also supported.

Each month (more or less) the Wikimedia Foundation posts a dump file of the en wikipedia. We are interested in the latest revisions only. For example, the dump file enwiki-20111007-pages-articles.xml.bz2 contains 11.7 million pages and articles, and occupies:

- 7.8G - as a compressed dump file (compressed with bzip2),
- 34G - as an xml file (after decompressed with bunzip2),
- 150G - when stored in a InnoDB table space (using the antelope storage format).

The dump file contains no images. However, pages and articles do refer to a great number of image files. For example, the dump file enwiki-20111007-pages-articles.xml.bz2 refers to 1.6 million images, which after downloading occupy 1T when stored in a file system.

If one wanted to mirror everything, then I would guess (to the nearest power of 10) a disk storage requirement of:

- 10T - for the en wikipedia, with all revisions, user talk, etc.
- 100T - for all languages with everything.

Update (late 2012): Storage estimate for en wikipedia images is now 3T.

F.3 Size Distribution

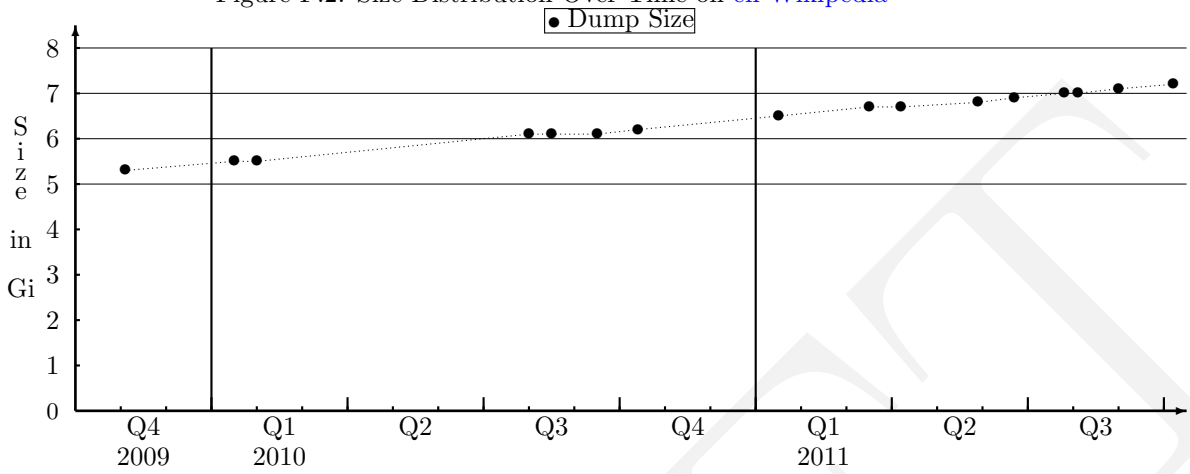
F.3.1 Size Distribution—by Language

The en wikipedia is the most challenging case. The simple wikipedia looks manageable for laptops. See Table F.1, Size Distribution by Language.

F.3.2 Size Distribution—over Time

The size of the en wikipedia dump file enwiki-*yyyymmdd*-pages-articles.xml.bz2 file has been growing. See Figure F.2, Size Distribution Over Time on en Wikipedia.

Figure F.2: Size Distribution Over Time on en Wikipedia



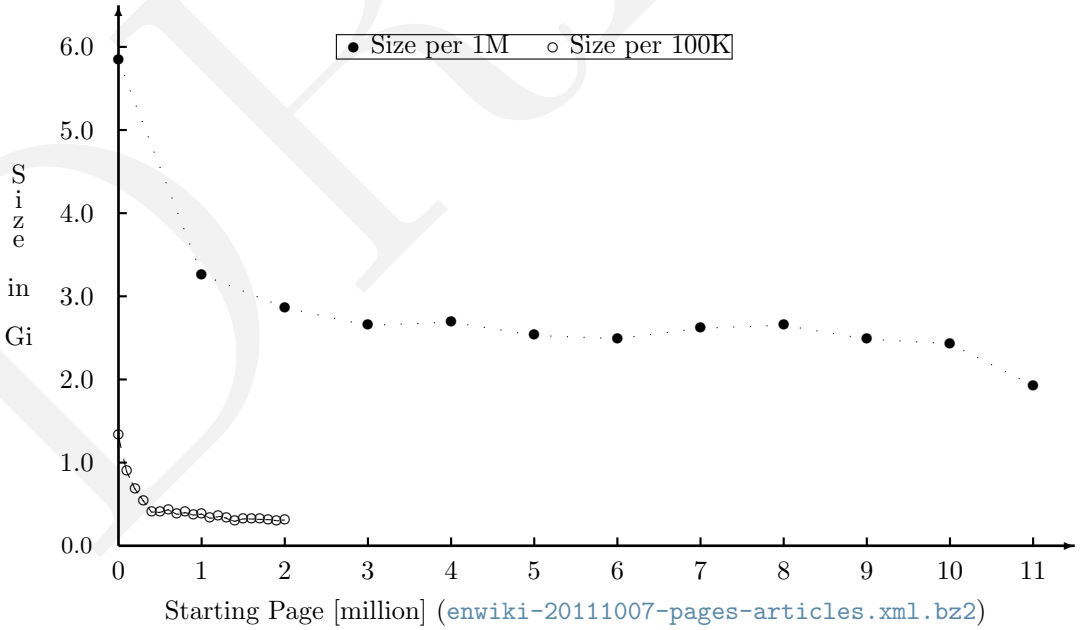
F.3.3 Size Distribution—by Age of Article

The oldest pages are the largest (especially the first 400k). See Figure F.3, Size Distribution by Age of Article on en Wikipedia. The oldest pages also have the most images (especially the first million). See Figure F.4, Size Distribution by `ichunk` on en Wikipedia.

Processing of the oldest pages consumes more resources (disk, cpu, bandwidth, and time) than newer pages. The image download rate (measured in `ichunk`s per day) accelerates, because the newer pages have fewer images. However, the page importation rate (measured in `xchunk`s per day):

- decelerates, for initial mirror building, because the `pagelinks` database table grows too large to fit in the `InnoDB buffer pool`, and hence disk I/O increases; but
- accelerates, for mirror *updating*, because newer pages are smaller.

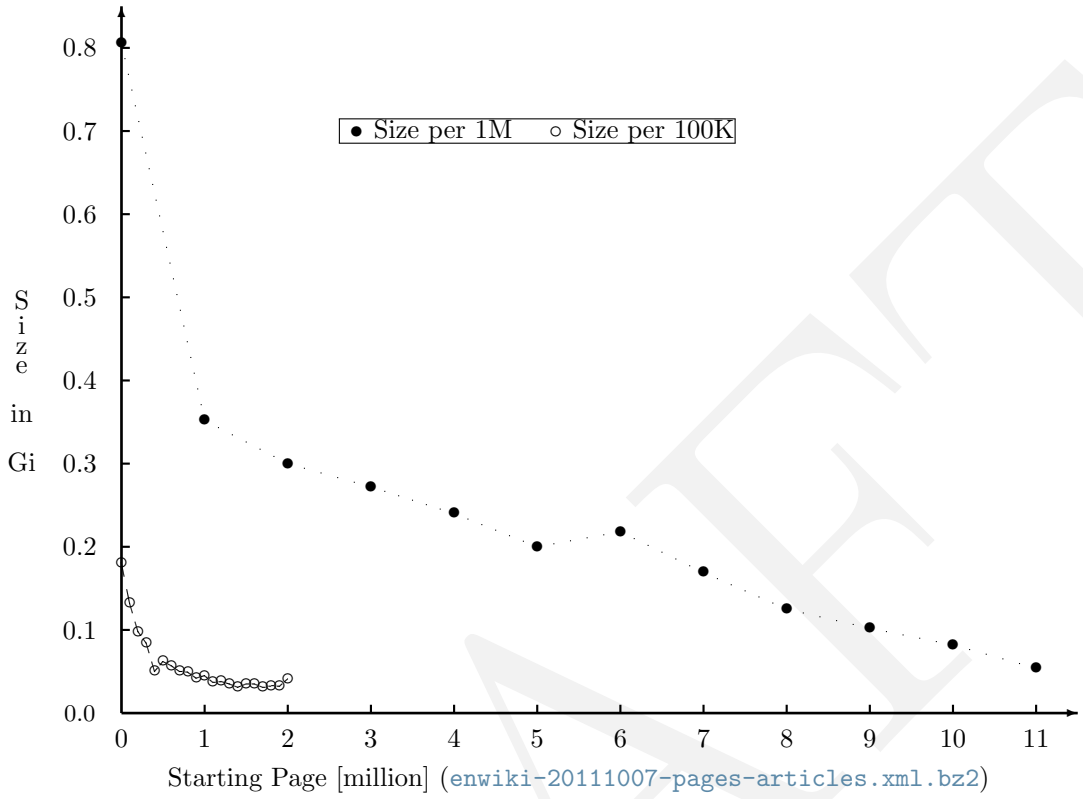
Figure F.3: Size Distribution by Age of Article on en Wikipedia



The data for the size distribution figures:

- Figure F.3, Size Distribution by Age of Article on en Wikipedia, and

Figure F.4: Size Distribution by ichunk on en Wikipedia



- Figure F.4, Size Distribution by ichunk on en Wikipedia

were produced by the following SQL commands.

```
mysql> SELECT FLOOR(page/1000000)*1000000 AS page_set,
    ->        SUM(size)/1000000000 AS 'size [Gi]'
    -> FROM file WHERE type='xchunk' GROUP BY page_set;

mysql> SELECT FLOOR(page/100000)*100000 AS page_set,
    ->        SUM(size)/1000000000 AS 'size [Gi]'
    -> FROM file WHERE type='xchunk' GROUP BY page_set LIMIT 20;

mysql> SELECT FLOOR(page/1000000)*1000000 AS page_set,
    ->        SUM(size)/1000000000 AS 'size [Gi]'
    -> FROM file WHERE type='ichunk' GROUP BY page_set;

mysql> SELECT FLOOR(page/100000)*100000 AS page_set,
    ->        SUM(size)/1000000000 AS 'size [Gi]'
    -> FROM file WHERE type='ichunk' GROUP BY page_set LIMIT 20;
```

F.4 Namespace Distribution

Each page belongs to a namespace. However, not all pages are found in the dump files. See Table F.2, Wikipedia Namespaces.

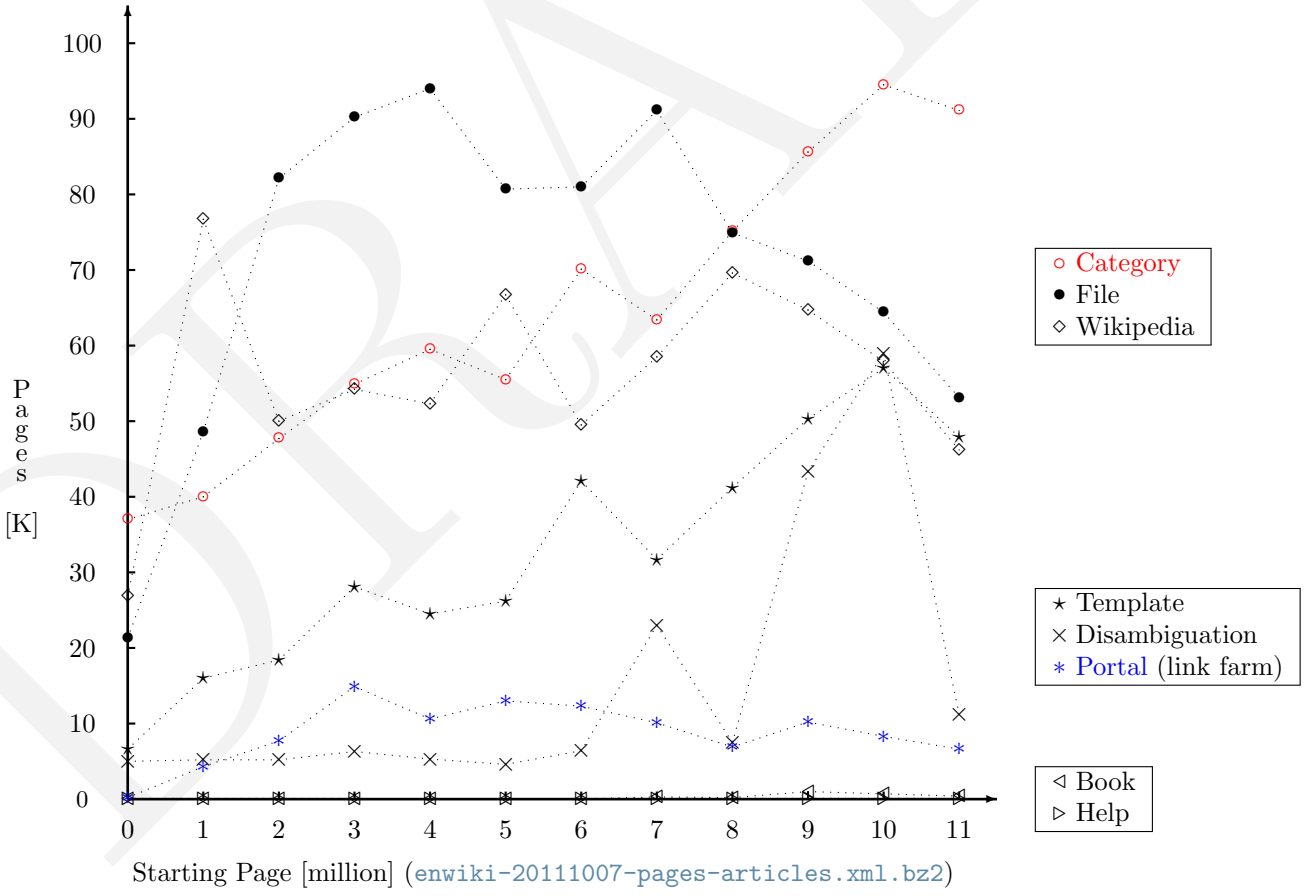
Some trends are apparent. The Category and Template namespaces have become more popular with time. Recently there was a burst of activity writing Disambiguation pages (between pages 9,000,000 and 11,000,000). The File namespace is gradually declining, perhaps due to the

Table F.2: Wikipedia Namespaces

Namespace	Purpose	In Dump
Main	encyclopedia article	Yes
Book	wikipedia books	Yes
Category	topical link farm	Yes
File	audio, image, video	Yes
Help	user and software manual	Yes
Portal	subject area main page	Yes
Template	page inside a page	Yes
Wikipedia	wikipedia project	Yes
Mediawiki	text for auto generated page	No
Summary	liquid thread summary	No
Thread	liquid thread (forum)	No
User	personal use	No

growing use of the Commons for storing images. See Figure F.5, Namespace Distribution by Age of Page on en Wikipedia for trends.

Figure F.5: Namespace Distribution by Age of Page on en Wikipedia



Screen shots of pages from different namespaces are shown below.

Figure F.6: Page in Category Namespace

Figure F.7: Page in Help Namespace

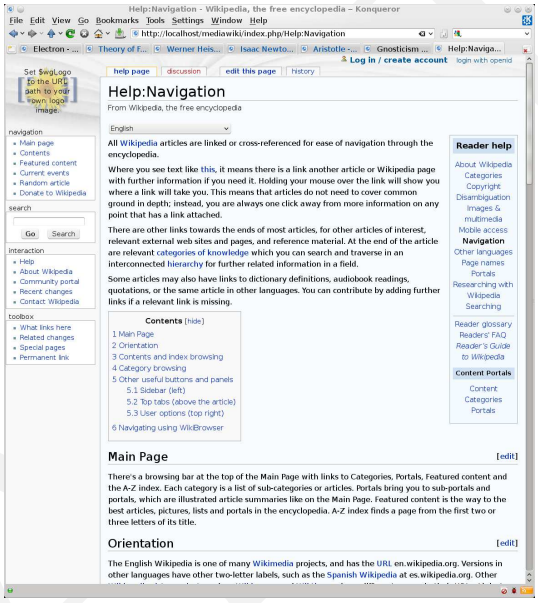


Figure F.8: Page in Main Namespace

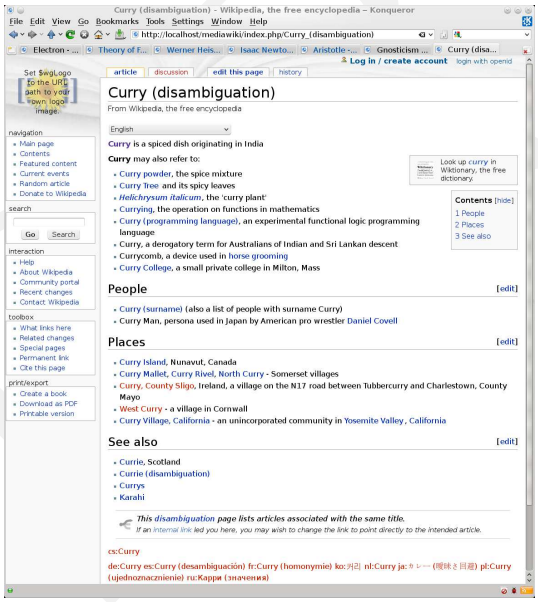


Figure F.9: Page in Portal Namespace

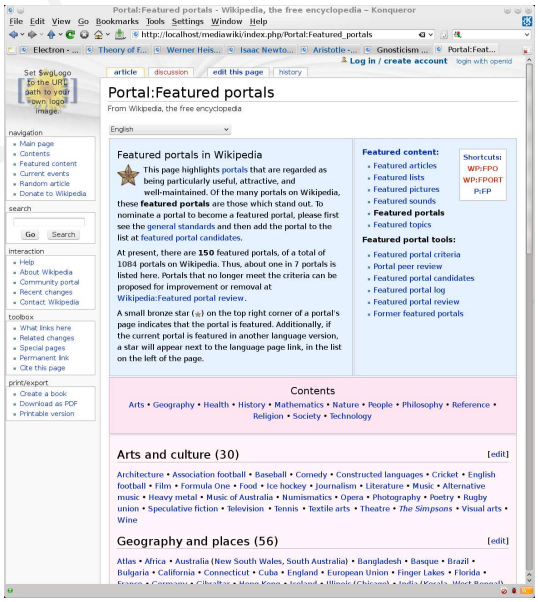


Figure F.10: Page with Small Template

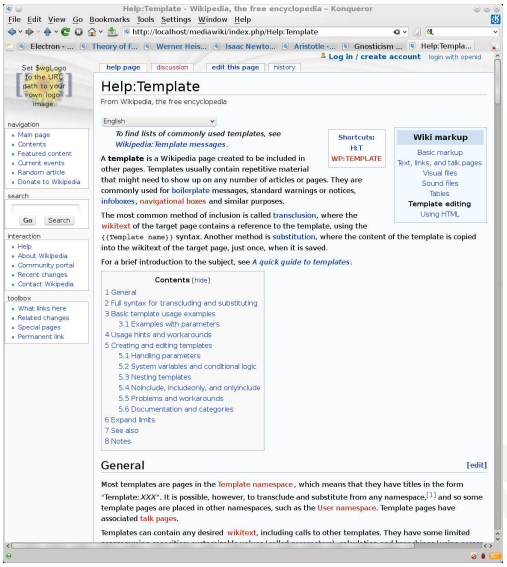


Figure F.11: Page with Large Template

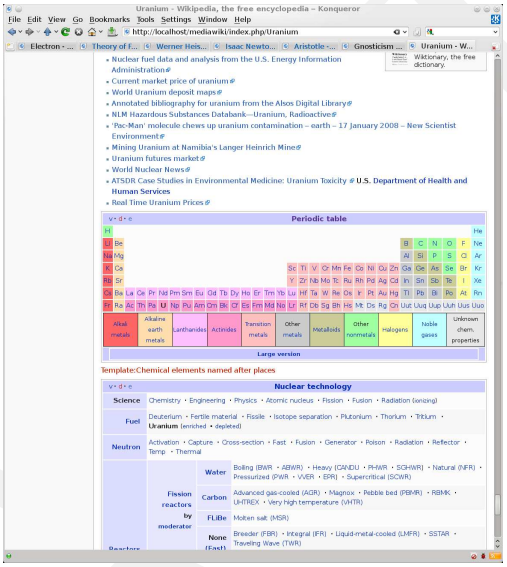
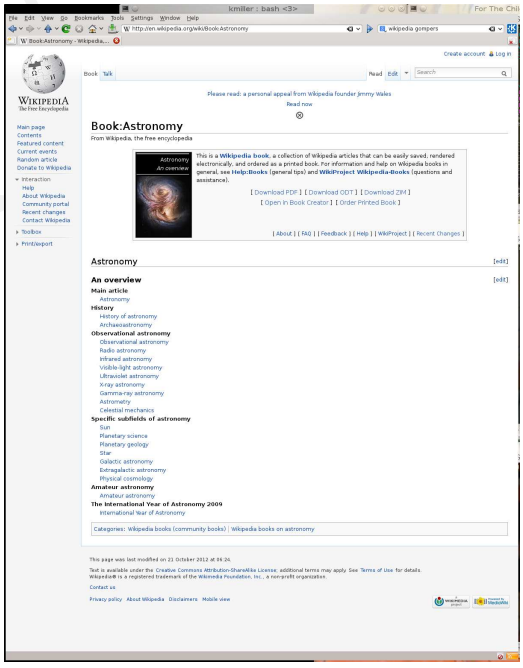


Figure F.12: Page in Wikipedia Namespace

Figure F.13: Page in Book Namespace



Appendix G

Experiments (Winter 2012–2013)

G.1 Introduction

Development for WP-MIRROR 0.6 focused on performance issues. This required profiling the performance of the Finite State Machine. The command-line option `--profile` was introduced to display data collected during a run.

The purpose of profiling is to identify the functions that are consuming most of the resources, and then to look for ways to improve the performance of those functions.

This set of experiments included:

- Baseline: profiling WP-MIRROR before any optimization in order to determine which functions needed optimization;
- Performance vs. durability: measuring the trade-off between Durability (the 'D' in ACID) and database performance;
- Performance and persistent connections: measuring the effect of HTTP/1.1 persistent connections for downloading image files;
- Performance and compilation: measuring the effect of compiling for maximum speed, a function containing a computationally expensive loop; and
- Performance vs. image processing: separately measuring the time spent on downloading and validating images.

Source: http://www.mysqlperformanceblog.com/2010/02/28/maximal-write-througput-in-mysql/, accessed 2012-12-28.

G.2 Baseline

Let us profile building a new mirror containing both the xh and zu wikipedias. These wikipedias are much smaller than the simple wikipedia, and are chosen in hopes of collecting profiling data adequate to the task of optimizing WP-MIRROR 0.6, without unduly burdening the servers at the Wikimedia Foundation.

First, since the author's system stands behind a caching web proxy, let us first `ssh` into the proxy and delete its cache of wikimedia images by executing:

```
shell$ ssh proxy
proxy$ su
Password:
proxy# /etc/init.d/polipo stop
proxy# rm -rf /var/cache/polipo/upload.wikimedia.org/
proxy# /etc/init.d/polipo start
proxy# exit
proxy$ exit
shell$
```

Second, let us choose our languages:

```
root-shell# emacs /etc/wp-mirror/local.conf
(defparameter *mirror-language-code-list* '("xh" "zu"))
```

Third, we make a series of time trials:

Run 1: Build the mirror and call this the 'baseline':

```
root-shell# wp-mirror --mirror
...
```

Run 2: Drop mirror, deleting local images, but keeping images cached by proxy, then rebuild mirror and call this the 'proxy baseline':

```
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
shell$ mysql --host=localhost --user=root --password --execute="DROP TABLE wpmirror.image"
Enter password:
root-shell# wp-mirror --mirror
```

Run 3: Drop mirror, keeping images, then rebuild mirror and call this the 'non-image baseline':

```
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# wp-mirror --mirror
```

Run 4: Rebuild mirror (this is the usual result of the weekly `cron` job when the Wikimedia Foundation has not posted a new dump).

Finally, let us display the profiles by executing:

```
root-shell# wp-mirror --profile
...
+------------------------+----+-------+--------+--------+
| Function               |  4 |     3 |      2 |      1 |
+------------------------+----+-------+--------+--------+
| fsm-boot               |  0 |     0 |      0 |      0 |
| fsm-database-checksum  |  0 |     0 |      0 |      0 |
| fsm-database-create    |  0 |    48 |     43 |     45 |
| fsm-database-grant     |  0 |     0 |      0 |      0 |
| fsm-database-interwiki |  0 |     0 |      0 |      0 |
| fsm-file-count         |  0 |     1 |      1 |      1 |
| fsm-file-decompress    |  0 |     1 |      0 |      0 |
| fsm-file-digest        |  0 |     0 |      1 |      0 |
| fsm-file-download      |  0 |     0 |      0 |      0 |
| fsm-file-import        |  0 | 1,414 |  1,350 |  1,387 |
| fsm-file-parse         |  0 |     0 |      0 |      0 |
| fsm-file-remove        |  0 |     0 |      0 |      0 |
| fsm-file-scrape        |  0 |   133 |    134 |    136 |
| fsm-file-shell         |  0 |   169 |  1,308 |  1,525 |
| fsm-file-split         |  0 |     1 |      1 |      1 |
| fsm-file-validate      |  0 |     0 |      0 |      0 |
| fsm-images-chown       |  0 |     9 |     10 |     10 |
| fsm-images-count       |  0 |    16 |     17 |     16 |
| fsm-images-rebuild     |  0 |   412 |    443 |    484 |
| fsm-images-validate    |  0 |    45 |  6,003 |  7,179 |
| fsm-no-op              |  0 |     0 |      0 |      0 |
| FSM-TOTAL              |  7 | 3,345 | 10,411 | 11,889 |
| GRAND-TOTAL            | 10 | 3,348 | 10,456 | 11,933 |
+------------------------+----+-------+--------+--------+
```

Most of the 'wall clock' time was consumed by three FSM functions:

- fsm-file-import which loads wikipedia pages into the database,
- fsm-file-shell which downloads image files, and
- fsm-images-validate which sequesters corrupt files.

and fsm-images-validate spent most of its time on one large (90M) image file, Hans_Holbein_the_Younger_-_The_Ambassadors_-_Google_Art_Project.jpg. Apparently, gm placed a 1.9G scratch file under /tmp/, and then spent the better part of 2 hours thrashing.

Let us examine the performance of the three functions:

- fsm-file-import consumes the same amount of time (runs 1, 2, and 3) independent of what we do with image files;
- fsm-file-shell now takes time for the initial download from the Wikimedia Foundation (run 1), shows some savings when downloading from a proxy (run 2), and consumes little time once the images have been downloaded (run 3);
- and fsm-images-validate is expensive (runs 1 and 2), unless the results from a previous run are stored in the wpmirror.image database table (run 3).

Let us now see what can be done to improve the performance of these three functions.

G.3 fsm-file-import and Durability

Motivation: fsm-file-import performance may be improved by tinkering with the database management system (DBMS) and the underlying hardware.

Concept: Full ACID compliance is achieved at a cost; and yet, full ACID compliance may not be necessary, in our case.

For full ACID compliance, each transaction must be flushed to disk right after it COMMITs. There is, however, a case to be made for backing away from full ACID compliance. WP-MIRROR was designed for robustness in the face of problematic dump and image files. To achieve robustness, the work was split up into small chunks, separated by checkpoints. If the processing of a chunk fails, the chunk is passed over. In the event of system failure, processing is resumed from the last checkpoint.

The use of checkpoints means that transactions lost due to system failure, are repeated when WP-MIRROR resumes. In other words, full ACID compliance is not a requirement, in our case.

Let us see if any performance gains can be had:

Run 1: The 'no-image baseline'.

Process: When a transaction COMMITs:

- transaction is written to log buffer,
- log buffer is written immediately to log file,
- log file is written immediately to physical disk.

Durability: Greatest possible durability.

```
root-shell# wp-mirror --profile 0           <-- drop old profiles
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# wp-mirror --mirror
```

Run 2: Enable HDD Write Cache

Process: When a transaction COMMITs:

- transaction is written to log buffer,
- log buffer is written immediately to log file,
- log file is flushed immediately to HDD write cache,
- HDD write cache is written *later* to physical disk.

Durability: System failure can cause loss of any transaction held in HDD write cache that is not yet written to the physical disk.

Configuration:

```
root-shell# emacs /etc/wp-mirror/local.conf
(defparameter *system-hdd-write-cache*           1)   ; 0=disabled, 1=enabled
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# wp-mirror --mirror
```

Run 3: Flush InnoDB log file once per second

Process: When a transaction COMMITs:

- transaction is written to log buffer,
- log buffer is written immediately to log file,
- log file is written *once per second* to HDD write cache,
- HDD write cache is *later* written to physical disk.

Durability: System failure can cause loss of up to one second of transactions.

Configuration:

```
root-shell# emacs /etc/mysql/conf.d/wp-mirror.cnf
innodb_flush_log_at_trx_commit  =    2 # default 1. (2=write log file once/sec)
sync_binlog                     =    0 # default 0.
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# wp-mirror --mirror
```

Run 4: Flush InnoDB log buffer once per second
Process: When a transaction COMMITs:

- transaction is written to log buffer,
- log buffer is written *once per second* to log file,
- log file is flushed immediately to HDD write cache,
- HDD write cache is written *later* to physical disk.

Durability: Any mysqld crash, and any system failure, can cause loss of up to one second of transactions.

Configuration:

```
root-shell# emacs /etc/mysql/conf.d/wp-mirror.cnf
innodb_flush_log_at_trx_commit  =    0 # default 1. (0=write log buffer once/sec)
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# wp-mirror --mirror
```

Finally, let us display the profiles by executing:

```
root-shell# wp-mirror --profile
...
+------------------------+-------+-------+-------+-------+
| Function               |     4 |     3 |     2 |     1 |
+------------------------+-------+-------+-------+-------+
| fsm-boot               |     0 |     0 |     0 |     0 |
| fsm-database-checksum  |     0 |     0 |     0 |     0 |
| fsm-database-create    |    30 |    29 |    28 |    47 |
| fsm-database-grant     |     0 |     0 |     0 |     0 |
| fsm-database-interwiki |     0 |     0 |     0 |     0 |
| fsm-file-count         |     2 |     1 |     1 |     1 |
| fsm-file-decompress    |     0 |     1 |     0 |     0 |
| fsm-file-digest        |     1 |     1 |     1 |     0 |
| fsm-file-download      |     5 |     1 |     3 |     1 |
| fsm-file-import        | 1,145 | 1,151 | 1,328 | 1,459 |
| fsm-file-parse         |     0 |     0 |     0 |     0 |
| fsm-file-remove        |     0 |     0 |     0 |     0 |
| fsm-file-scrape        |   133 |   133 |   136 |   132 |
| fsm-file-shell         |   170 |   170 |   169 |   168 |
| fsm-file-split         |     1 |     1 |     1 |     1 |
| fsm-file-validate      |     0 |     0 |     0 |     0 |
| fsm-images-chown       |    10 |    10 |    10 |     9 |
| fsm-images-count       |    19 |    19 |    18 |    16 |
| fsm-images-rebuild     |   216 |   202 |   639 |   428 |
| fsm-images-validate    |    50 |    49 |    50 |    45 |
| fsm-no-op              |     0 |     0 |     0 |     0 |
| FSM-TOTAL              | 2,749 | 2,742 | 3,470 | 3,405 |
| GRAND-TOTAL            | 2,751 | 2,745 | 3,473 | 3,407 |
+------------------------+-------+-------+-------+-------+
```

Analysis: Significant performance improvements are seen for fsm-file-import (22% less time) and fsm-images-rebuild (50% less time) when HDD write caching is enabled and transactions are flushed to disk once per second (runs 3 and 4). However there is negligible performance difference between: run 3 (innodb_flush_log_at_trx_commit = 2) which may lose transactions during system failure; and run 4 (innodb_flush_log_at_trx_commit = 0) which may lose transaction during system failure and during mysqld failure. Therefore, we should choose the former because it has fewer failure modes.

Recommendation: Default values for WP-MIRROR 0.6 should include:

```
shell$ cat /usr/bin/wp-mirror | grep defparameter | grep write-cache
(defparameter *system-hdd-write-cache*           1)   ; 0=disabled, 1=enabled
(defparameter *system-hdd-write-cache-flush*   nil)   ; t=flush at checkpoint
root-shell# cat /etc/mysql/conf.d/wp-mirror.cnf
[mysqld]
...
# for Durability (the 'D' in 'ACID compliance')
innodb_flush_log_at_trx_commit  =    2 # default 1. (2=write logfile once/sec)
sync_binlog                     =    0 # default 0.
...
```

G.4 fsm-file-shell and HTTP/1.1 Persistent Connections

Motivation: fsm-file-shell performance may be improved by using "persistent connections" as defined in RFC2616, HTTP/1.1.

Concept: Downloading several image files using a single TCP connection, reduces server workload and reduces latency between each download.

Implementation: Write shell scripts that invokes cURL with several image files specified on the command line. This permits downloading several files using a single TCP connection, thereby reducing latency for all but the first file.

Let us see if any performance gains can be had:

Run 1: The 'proxy baseline' with two changes:

- with HDD write caching enabled and InnoDB transactions flushed once per second (as recommended by §G.3, fsm-file-import and Durability) ; and
- keep the wpmirror.image database table to avoid repeating image validation (because validation takes time and is not relevant to measuring the use of persistent connections).

Set WP-MIRROR parameters to:

```
shell$ cat /usr/bin/wp-mirror | grep defparameter | grep connection
(defparameter *mirror-ichunk-connection-time-max*      3000)
(defparameter *mirror-ichunk-persistent-connection*       1)
(defparameter *mirror-ichunk-post-connection-sleep*       0)
```

which produces shell scripts that look like:

```
#!/bin/sh
COMMONSPATH=http://upload.wikimedia.org/wikipedia/commons/
IMAGEPATH=http://upload.wikimedia.org/wikipedia/xh/

sleep 0

curl -f -m 3000  \
$COMMONSPATH./6/60/ZICHYJFALU_-_Kastlypark.jpg -o ./6/60/ZICHYJFALU_-_Kastlypark.jpg

sleep 0

curl -f -m 3000  \
$COMMONSPATH./f/fd/IDSKI_BANOVCI.jpg -o ./f/fd/IDSKI_BANOVCI.jpg
...
```

Now execute:

```
root-shell# wp-mirror --profile 0          <-- drop old profiles
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 2: Two (2) image files per connection.
Configuration:

```
shell$ cat /usr/bin/wp-mirror | grep defparameter | grep persistent
(defparameter *mirror-ichunk-persistent-connection*       2)
```

which produces shell scripts that look like:

```
...
sleep 0

curl -f -m 3000 \
$COMMONSPATH./6/60/ZICHYJFALU_-_Kastlypark.jpg -o ./6/60/ZICHYJFALU_-_Kastlypark.jpg \
$COMMONSPATH./f/fd/IDSKI_BANOVCI.jpg -o ./f/fd/IDSKI_BANOVCI.jpg
...
```

Now execute:

```
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 3: Five (5) image files per connection.
Configuration:

```
shell$ emacs /usr/bin/wp-mirror
(defparameter *mirror-ichunk-persistent-connection*       5)
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 4: Ten (10) image files per connection.
Configuration:

```
shell$ emacs /usr/bin/wp-mirror
(defparameter *mirror-ichunk-persistent-connection*      10)
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 5: Twenty (20) image files per connection.
Configuration:

```
shell$ emacs /usr/bin/wp-mirror
(defparameter *mirror-ichunk-persistent-connection*      20)
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Finally, let us display the profiles by executing:

```
root-shell# wp-mirror --profile
...
```

Function	5	4	3	2	1
fsm-boot	0	0	0	0	0
fsm-database-checksum	0	0	0	0	0
fsm-database-create	32	30	28	31	37
fsm-database-grant	0	0	0	0	0
fsm-database-interwiki	0	0	0	0	0
fsm-file-count	1	1	1	1	1
fsm-file-decompress	1	0	1	1	0
fsm-file-digest	1	1	1	1	1
fsm-file-download	1	1	1	1	2
fsm-file-import	1,153	1,152	1,193	1,153	1,143
fsm-file-parse	0	0	0	0	0
fsm-file-remove	0	0	0	0	0
fsm-file-scrape	135	135	134	133	134
fsm-file-shell	487	521	479	747	1,308
fsm-file-split	1	1	1	1	1
fsm-file-validate	0	0	0	0	0
fsm-images-chown	9	10	10	11	11
fsm-images-count	19	19	19	19	19
fsm-images-rebuild	216	212	225	221	222
fsm-images-validate	50	49	50	49	49
fsm-no-op	0	0	0	0	0
FSM-TOTAL	3,074	3,111	3,121	3,331	3,883
GRAND-TOTAL	3,098	3,134	3,145	3,356	3,906

Analysis: Significant performance improvement for fsm-file-shell (64% less time) is seen by downloading five image files per TCP connection (run 3). No further improvement is seen by downloading 10 or 20 image files per TCP connection (runs 4 and 5). However, for this experiment, image files were downloaded from a nearby caching web proxy. Therefore, greater latency should be expected when downloading from the Wikimedia Foundation; and, the optimum number of image files per TCP connection should be greater, say 10.

Recommendation: Default values for WP-MIRROR 0.6 should include:

```
shell$ cat /usr/bin/wp-mirror | grep defparameter | grep connection
(defparameter *mirror-ichunk-connection-time-max*      3000)
(defparameter *mirror-ichunk-persistent-connection*      10)
(defparameter *mirror-ichunk-post-connection-sleep*       0)
```

G.5 fsm-file-scrape and compile

Motivation: fsm-file-scrape performance may be improved by using compiling the function for maximum speed.

Concept: common lisp offers the programmer great control over what *forms* are interpreted vs. compiled; and, if compiled, the degree to which they are optimized for an 'optimize quality'. The standard 'optimize qualities' are: compilation-speed, debug, safety, space, and speed.

Implementation: The function fsm-file-scrape calls parse-image-file-names which contains a rather expensive loop, which should be compiled for maximum speed.

It would be enough to compile just the loop, but the author chose to compile the whole function. This is done by placing an optimize declaration:

```
(defun parse-image-file-names ...
...
  (declare (optimize (speed 3)))
      (loop
...
```

above the loop within the body of the function. Then at the bottom of the code, we call the compile function:

```
(compile 'parse-image-file-names)
```

There are, in addition, a number of other (frequently called or expensive) functions that will be tested below.

Let us see if any performance gains can be had:

Run 1: The 'proxy baseline' with three changes:

- with HDD write caching enabled and InnoDB transactions flushed once per second (as recommended by §G.3, fsm-file-import and Durability) ;
- keep the wpmirror.image database table to avoid repeating image validation (because validation takes time and is not relevant to measuring parsing); and
- with HTTP/1.1 persistent connections for downloading image files (as recommended by §G.4, fsm-file-shell and HTTP/1.1 Persistent Connections).

Configuration:

```
root-shell# wp-mirror --profile 0          <-- drop old profiles
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 2: Optimize parse-image-file-names for speed, and compile.

Configuration: Edit code as described above, then

```
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 3: Optimize the following 'overhead' functions for speed, and compile:

- fsm-function, fsm-process-file, and fsm-transition;
- parse-image-file-names; and
- sql-select-next-file, and sql-select-next-file-type-state.

Configuration: Edit code as described above, then

```
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 4: Optimize the following functions for speed, and compile:

- fsm-function, fsm-images-validate, fsm-process-file, and fsm-transition; and
- parse-image-file-names, put-and-log-string;
- query-database-and-return-stream, query-database-and-return-list-of-strings, query-database-and-return-list-of-lists;

- shell-command;
- sql-select-next-file, sql-select-next-file-type-state, sql-select-numeric, and sql-update-file.

Configuration: Edit code as described above, then

```
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 5: Optimize the following functions for speed, and compile:

- _, filter-exist;
- fsm-file-shell, fsm-function, fsm-images-rebuild, fsm-images-validate, fsm-process-file, and fsm-transition;
- parse-image-file-names, put-and-log-string, and put-log-string;
- query-database-and-return-stream, query-database-and-return-list-of-strings, query-database-and-return-list-of-lists;
- shell-command;
- sql-select-next-file, sql-select-next-file-type-state, sql-select-numeric, and sql-update-file.

Configuration: Edit code as described above, then

```
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Finally, let us display the profiles by executing:

```
root-shell# wp-mirror --profile
...
+------------------------+-------+-------+-------+-------+-------+
| Function               |     5 |     4 |     3 |     2 |     1 |
+------------------------+-------+-------+-------+-------+-------+
| fsm-boot               |     0 |     0 |     0 |     0 |     0 |
| fsm-database-checksum  |     0 |     0 |     0 |     0 |     0 |
| fsm-database-create    |    34 |    34 |    41 |    29 |    30 |
| fsm-database-grant     |     0 |     0 |     0 |     0 |     0 |
| fsm-database-interwiki |     0 |     0 |     0 |     0 |     0 |
| fsm-file-count         |     2 |     1 |     1 |     1 |     1 |
| fsm-file-decompress    |     0 |     0 |     0 |     1 |     0 |
| fsm-file-digest        |     1 |     1 |     1 |     1 |     0 |
| fsm-file-download      |     1 |     1 |     1 |     1 |     1 |
| fsm-file-import        | 1,185 | 1,203 | 1,208 | 1,189 | 1,172 |
| fsm-file-parse         |     0 |     0 |     0 |     0 |     0 |
| fsm-file-remove        |     0 |     0 |     0 |     0 |     0 |
| fsm-file-scrape        |    24 |    25 |    25 |    25 |   137 |
| fsm-file-shell         |   444 |   423 |   440 |   457 |   544 |
| fsm-file-split         |     1 |     1 |     1 |     1 |     1 |
| fsm-file-validate      |     0 |     0 |     0 |     0 |     0 |
| fsm-images-chown       |    13 |    11 |    10 |    11 |     8 |
| fsm-images-count       |    18 |    19 |    19 |    18 |    19 |
| fsm-images-rebuild     |   207 |   209 |   218 |   209 |   211 |
| fsm-images-validate    |    44 |    43 |    50 |    49 |    50 |
| fsm-no-op              |     0 |     0 |     0 |     0 |     0 |
| FSM-TOTAL              | 2,921 | 2,932 | 2,983 | 2,968 | 3,150 |
| GRAND-TOTAL            | 2,944 | 2,954 | 3,007 | 2,991 | 3,172 |
+------------------------+-------+-------+-------+-------+-------+
  FSM-OVERHEAD           947     961     968     976     976
```

Analysis: Impressive performance gains for `fsm-file-scrape` (82% less time) are seen from compiling `parse-image-file-names` for speed (run 2). A collateral increase for `fsm-file-shell` (16% less time) is observed (run 2). An increase for `fsm-images-validate` (12% less time) is also seen (run 4). This last observation does not say much, because `gm` was not invoked for any of these runs.

Let us define 'FSM-OVERHEAD' as the difference between the sum of `fsm-*` times and the FSM-TOTAL time. Little performance increase for FSM Overhead (3% less time) is seen by compiling the other functions listed above (runs 3, 4, and 5).

Recommendation: The functions `parse-image-file-names` and `fsm-images-validate` should be compiled for speed in WP-MIRROR 0.6.

G.6 Performance and Images

Motivation: Mirror building might be quickened by either not downloading images or by not validating after downloading.

Implementation: WP-MIRROR has two variables `*mirror-image-download-p*` and `*mirror-image-validate-p*` that can be set to `nil` or `t`. This makes four combinations.

Run 1: The 'proxy baseline' with four changes:

- with HDD write caching enabled and `InnoDB` transactions flushed once per second (as recommended by §G.3, `fsm-file-import` and Durability) ;
- keep the `wpmirror.image` database table to avoid repeating image validation (because validation takes time and is not relevant to measuring parsing);
- with `HTTP`/1.1 persistent connections for downloading image files (as recommended by §G.4, `fsm-file-shell` and `HTTP`/1.1 Persistent Connections); and

- with parse-image-file-names compiled for maximum speed (as recommended by §G.6, Performance and Images).

Configuration:

```
root-shell# emacs /etc/wp-mirror/local.conf
(defparameter *mirror-image-download-p*  nil)
(defparameter *mirror-image-validate-p*  nil)
root-shell# wp-mirror --profile 0          <-- drop old profiles
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 2: Enable image validation (with no images to validate).
Configuration:

```
root-shell# emacs /etc/wp-mirror/local.conf
(defparameter *mirror-image-download-p*  nil)
(defparameter *mirror-image-validate-p*    t)
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 3: Enable image downloading.
Configuration:

```
root-shell# emacs /etc/wp-mirror/local.conf
(defparameter *mirror-image-download-p*    t)
(defparameter *mirror-image-validate-p*  nil)
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Run 4: Enable image downloading and validation.
Configuration:

```
root-shell# emacs /etc/wp-mirror/local.conf
(defparameter *mirror-image-download-p*    t)
(defparameter *mirror-image-validate-p*    t)
root-shell# wp-mirror --drop xh
root-shell# wp-mirror --drop zu
root-shell# rm -r /var/lib/mediawiki/images/
root-shell# wp-mirror --mirror
```

Finally, let us display the profiles by executing:

```
root-shell# wp-mirror --profile
...
+------------------------+-------+-------+-----+-----+
| Function               |     4 |     3 |   2 |   1 |
+------------------------+-------+-------+-----+-----+
| fsm-boot               |     0 |     0 |   0 |   0 |
| fsm-database-checksum  |     0 |     0 |   0 |   0 |
| fsm-database-create    |    35 |    33 |  32 |  35 |
| fsm-database-grant     |     0 |     0 |   0 |   0 |
| fsm-database-interwiki |    12 |    12 |  12 |  12 |
| fsm-file-count         |     1 |     1 |   1 |   1 |
| fsm-file-decompress    |     1 |     0 |   0 |   0 |
| fsm-file-digest        |     0 |     1 |   0 |   0 |
| fsm-file-download      |     1 |     6 |   0 |   0 |
| fsm-file-import        | 1,215 | 1,385 | 596 | 589 |
| fsm-file-parse         |     0 |     0 |   0 |   0 |
| fsm-file-remove        |     0 |     0 |   0 |   0 |
| fsm-file-scrape        |    25 |    25 |   0 |   0 |
| fsm-file-shell         |   461 |   466 |   0 |   0 |
| fsm-file-split         |     1 |     1 |   1 |   1 |
| fsm-file-validate      |     0 |     0 |   0 |   0 |
| fsm-images-chown       |    11 |    14 |   0 |   0 |
| fsm-images-count       |    19 |    19 |   0 |   0 |
| fsm-images-rebuild     |   337 |   353 |   0 |   0 |
| fsm-images-validate    | 6,345 |     7 |   0 |   0 |
| fsm-no-op              |     0 |     0 |   0 |   0 |
| FSM-TOTAL              | 9,435 | 3,292 | 662 | 657 |
| GRAND-TOTAL            | 9,458 | 3,315 | 684 | 679 |
+------------------------+-------+-------+-----+-----+
```

Analysis: Downloading images (run 3) triggers many image processing tasks. Interestingly, it also more than doubles the time spent by `fsm-file-import`. This is because the resizing of images to make thumbs is part of this process. Resizing is performed by `gm convert` and `inkscape`.

Validating images (run 4) adds an expensive task. The intention is to identify bad images and sequester them before attempting to resize the. Previous experiments with the en wikipedia revealed great numbers of corrupt image files, many of which hung the system when resized. See §E.14, Experiments with Corrupt Images and a collection of error messages in §E.19, Messages. That said, I have not seen this problem with the smaller wikipedias.

Recommendation: The default value for `*mirror-image-validate-p*` should be set to `nil` for WP-MIRROR 0.6. If the user encounters corrupt images, then the user can enable validation.

G.7 Miscellaneous

G.7.1 Query Cache

```
shell$ mysql --host=localhost --user=root --password
Password:
...
mysql> SHOW GLOBAL VARIABLES LIKE '%query_cache%';
+------------------------------+----------+
| Variable_name                | Value    |
+------------------------------+----------+
| have_query_cache             | YES      |
| query_cache_limit            | 1048576  |
| query_cache_min_res_unit     | 4096     |
| query_cache_size             | 16777216 |
| query_cache_type             | ON       |
| query_cache_wlock_invalidate | OFF      |
+------------------------------+----------+
6 rows in set (0.00 sec)
mysql> SHOW GLOBAL STATUS LIKE 'Qcache%';
+-------------------------+----------+
| Variable_name           | Value    |
+-------------------------+----------+
| Qcache_free_blocks      | 980      |
| Qcache_free_memory      | 10983744 |
| Qcache_hits             | 2191250  |
| Qcache_inserts          | 1371568  |
| Qcache_lowmem_prunes    | 0        |
| Qcache_not_cached       | 5259042  |
| Qcache_queries_in_cache | 4827     |
| Qcache_total_blocks     | 10656    |
+-------------------------+----------+
8 rows in set (0.00 sec)
```

Analysis: We see that the query_cache_size is 16M, of which 9%, 0.09 = 980/10656, is free. The default value for query_cache_size appears to be adequate.

Recommendation: Keep the default value.

G.7.2 Database Tuning

According to the MySQL 5.5 Reference Manual,

> When tuning a MySQL server, the two most important variables to configure are key_buffer_size and table_open_cache. You should first feel confident that you have these set appropriately before trying to change any other variables.
>
> MySQL 5.5 Reference Manual, Chap. 8 Optimization, p. 860

G.7.2.1 key_buffer_size Tuning

```
shell$ mysql --host=localhost --user=root --password
Password:
...
mysql> SHOW GLOBAL VARIABLES LIKE 'key%';
+-------------------------+----------+
| Variable_name           | Value    |
+-------------------------+----------+
| key_buffer_size         | 16777216 |
| key_cache_age_threshold | 300      |
| key_cache_block_size    | 1024     |
| key_cache_division_limit | 100     |
+-------------------------+----------+
4 rows in set (0.00 sec)
mysql> SHOW GLOBAL STATUS LIKE 'key%';
+------------------------+----------+
| Variable_name          | Value    |
+------------------------+----------+
| Key_blocks_not_flushed | 0        |
| Key_blocks_unused      | 4452     |
| Key_blocks_used        | 9239     |
| Key_read_requests      | 22646478 |
| Key_reads              | 113      |
| Key_write_requests     | 6238886  |
| Key_writes             | 4277027  |
+------------------------+----------+
7 rows in set (0.00 sec)
```

Analysis: We see that the key_buffer_size is 16M, of which 32%, $0.32 = 4452/(4452+9239)$, is free. The default value for key_buffer_size appears to be adequate.

Recommendation: Keep the default value.

G.7.2.2 table_open_cache Tuning

```
shell$ mysql --host=localhost --user=root --password
Password:
...
mysql> SHOW GLOBAL VARIABLES LIKE 'table_open%';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| table_open_cache | 400   |
+------------------+-------+
1 row in set (0.00 sec)
mysql> SHOW GLOBAL STATUS LIKE 'open_table%';
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| Com_show_open_tables  | 0     |
| Open_table_definitions | 46   |
| Open_tables           | 73    |
+-----------------------+-------+
3 rows in set (0.00 sec)
```

Analysis: The default value for table_open_cache appears to be more than adequate.

Recommendation: Keep the default value.

G.7.3 Optimizing Disk I/O

According to the MySQL 5.5 Reference Manual,

> On Linux, you can get much better performance by using hdparm to configure your disk's interface. (Up to 100% under load is not uncommon.) The following hdparm options should be quite good for MySQL, and probably for many other applications:
>
> ```
> hdparm -m 16 -d 1
> ```
>
> MySQL 5.5 Reference Manual, Chap. 8 Optimization, p. 862

G.7.3.1 Multiple Sector I/O

> Multiple sector mode (aka IDE Block Mode), is a feature of most modern IDE hard drives, permitting the transfer of multiple sectors per I/O interrupt. When this feature is enabled, it typically reduces operating system overhead for disk I/O by 30-50%.
>
> manpage for hdparm

```
root-shell# hdparm -m /dev/sda
/dev/sda:
 multcount     = 16 (on)
root-shell# hdparm -I /dev/sda | grep mult
        R/W multiple sector transfer: Max = 16  Current = 16
```

Recommendation: Keep default settings.

G.7.3.2 Direct Memory Access

```
root-shell# hdparm -d /dev/sda
/dev/sda:
 HDIO_GET_DMA failed: Inappropriate ioctl for device
root-shell# hdparm -I /dev/sda | grep DMA
       DMA: mdma0 mdma1 mdma2 udma0 udma1 udma2 udma3 udma4 udma5 *udma6
           *    WRITE_DMA|MULTIPLE_FUA_EXT
           *    READ,WRITE_DMA_EXT_GPL commands
           *    DMA Setup Auto-Activate optimization
```

Recommendation: Keep default settings.

G.8 Experiments with InnoDB data compression

Purpose: Determine which tables to compress with InnoDB Data Compression; and to determine the optimal page size for those table that compress well. By page size we mean setting KEY_BLOCK_SIZE=1,2,4,8, or 16 when ROW_FORMAT=COMPRESSED.

Intended benefit: Compressed tables should 1) reduce the time spent on disk I/O bound operations; and 2) ease the disk storage and memory constraints for users with laptops.

G.8.1 Theory

Some MediaWiki tables should compress well, other not.

> Character strings often compress well, whether defined in CHAR, VARCHAR, TEXT, or BLOB columns. On the other hand, tables containing mostly binary data (integers or floating point numbers) or data that is previously compressed (for example JPEG or PNG images) may not generally compress well, significantly or at all.
>
> MySQL 5.5 Reference Manual, Section 14.4.3 InnoDB Data Compression, p. 1628

The main benefit is improved performance of I/O bound operations. Economizing on storage and memory is a plus.

Unfortunately, improved performance does not happen for INSERT and UPDATE workloads.

> Compression and the InnoDB Log Files
>
> Before a compressed page is written to a database file, InnoDB writes a copy of the page to the redo log (if it has been recompressed since the last time it was written to the database). This is done to ensure that redo logs will always be usable, even if a future version of InnoDB uses a slightly different compression algorithm. Therefore, some increase in the size of log files, or a need for more frequent checkpoints, can be expected when using compression. The amount of increase in the log file size or checkpoint frequency depends on the number of times compressed pages are modified in a way that requires reorganization and recompression.
>
> MySQL 5.5 Reference Manual, Section 14.4.3 InnoDB Data Compression, p. 1633

G.8.2 Experimental method

Overall method: Measure the size of *.ibd files v. a variety of configurations.

Run: We use WP-MIRROR to build a mirror of simplewiki-20130112-pages-articles.xml.bz2. Then we identify the tables larger than 100M:

```
root-shell# cd /var/lib/mysql/
root-shell# ls -lh simplewiki/*ibd | grep [1-9][0-9][0-9]M
-rw-rw---- 1 mysql mysql 124M Jan 14 09:01 simplewiki/categorylinks.ibd
-rw-rw---- 1 mysql mysql 392M Jan 14 09:01 simplewiki/langlinks.ibd
-rw-rw---- 1 mysql mysql 420M Jan 14 09:01 simplewiki/pagelinks.ibd
-rw-rw---- 1 mysql mysql 108M Jan 14 09:00 simplewiki/templatelinks.ibd
-rw-rw---- 1 mysql mysql 476M Jan 14 09:01 simplewiki/text.ibd
```

Experimental method: For each of the above mentioned tables

1. **Setup:**
   - Reset the compression statistics:

     ```
     mysql> SELECT * FROM information_Schema.innodb_cmp_reset;
     mysql> SELECT * FROM information_Schema.innodb_cmpmem_reset;
     ```

   - Make empty copy in the test database:

     ```
     mysql> CREATE TABLE test.categorylinks LIKE simplewiki.categorylinks;
     ```

   - Set table options where KEY_BLOCK_SIZE = 1, 2, 4, 8, or 16:

     ```
     mysql> ALTER TABLE test.categorylinks ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=8;
     ```

2. **Run:**
   - Populate the copies with data:

     ```
     mysql> INSERT INTO test.categorylinks SELECT * FROM simplewiki.categorylinks;
     ```

   - Measure size of data file:

     ```
     mysql> SELECT * FROM information_schema.innodb_cmp;
     mysql> SELECT * FROM information_schema.innodb_cmpmem;
     root-shell# cd /var/lib/mysql/
     root-shell# ls -lh test/*ibd
     ```

3. **Tear-down:** Drop tables.

   ```
   mysql> DROP TABLE test.categorylinks;
   ...
   ```

G.8.3 EXPERIMENT.0 Baseline configuration

Code: Z-P16

Create copies of the tables:

```
mysql> INSERT INTO test.categorylinks SELECT * FROM simplewiki.categorylinks;
Query OK, 307658 rows affected (45.40 sec)
Records: 307658  Duplicates: 0  Warnings: 0
mysql> INSERT INTO test.langlinks     SELECT * FROM simplewiki.langlinks;
Query OK, 2890424 rows affected (8 min 47.26 sec)
Records: 2890424  Duplicates: 0  Warnings: 0
mysql> INSERT INTO test.pagelinks     SELECT * FROM simplewiki.pagelinks;
Query OK, 3006777 rows affected (5 min 16.12 sec)
Records: 3006777  Duplicates: 0  Warnings: 0
mysql> INSERT INTO test.templatelinks SELECT * FROM simplewiki.templatelinks;
Query OK, 791802 rows affected (38.08 sec)
Records: 791802  Duplicates: 0  Warnings: 0
mysql> INSERT INTO test.text          SELECT * FROM simplewiki.text;
Query OK, 161211 rows affected (1 min 36.74 sec)
Records: 161211  Duplicates: 0  Warnings: 0
```

Measurement:

```
root-shell# ls -lh test/*ibd
-rw-rw---- 1 mysql mysql  96M Jan 26 06:32 test/categorylinks.ibd
-rw-rw---- 1 mysql mysql 352M Jan 26 06:48 test/langlinks.ibd
-rw-rw---- 1 mysql mysql 296M Jan 26 06:53 test/pagelinks.ibd
-rw-rw---- 1 mysql mysql  76M Jan 26 06:54 test/templatelinks.ibd
-rw-rw---- 1 mysql mysql 476M Jan 26 06:56 test/text.ibd
```

Note the shrinkage for the first four tables. Both `data_length` and `index_length` shrink:

```
mysql> SHOW TABLE STATUS FROM simplewiki WHERE name='categorylinks'\G
*************************** 1. row ***************************
           Name: categorylinks
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 263136
 Avg_row_length: 185
    Data_length: 48889856
Max_data_length: 0
   Index_length: 72171520
      Data_free: 4194304
 Auto_increment: NULL
    Create_time: 2013-01-13 18:01:01
    Update_time: NULL
     Check_time: NULL
      Collation: binary
       Checksum: NULL
 Create_options:
        Comment:
1 row in set (0.38 sec)
```

```
mysql> SHOW TABLE STATUS FROM test WHERE name='categorylinks'\G
*************************** 1. row ***************************
           Name: categorylinks
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 292517
 Avg_row_length: 102
    Data_length: 29982720
Max_data_length: 0
   Index_length: 62701568
      Data_free: 4194304
 Auto_increment: NULL
    Create_time: 2013-01-26 11:23:02
    Update_time: NULL
     Check_time: NULL
      Collation: binary
       Checksum: NULL
 Create_options:
        Comment:
```

I would guess that the InnoDB pages for test.categorylinks are filled, whereas the pages for simplewiki.categorylinks have more spaces caused by repeated UPDATEs over many months.

The text table did not shrink when copied. This is because the text table only sees INSERTs at the high end of its primary key old_id, so it never fragments.

G.8.4 categorylinks

This looks like a good candidate for compression.

```
mysql> SHOW CREATE TABLE simplewiki.categorylinks\G
*************************** 1. row ***************************
       Table: categorylinks
Create Table: CREATE TABLE `categorylinks` (
  `cl_from` int(10) unsigned NOT NULL DEFAULT '0',
  `cl_to` varbinary(255) NOT NULL DEFAULT '',
  `cl_sortkey` varbinary(230) NOT NULL DEFAULT '',
  `cl_sortkey_prefix` varbinary(255) NOT NULL DEFAULT '',
  `cl_timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `cl_collation` varbinary(32) NOT NULL DEFAULT '',
  `cl_type` enum('page','subcat','file') NOT NULL DEFAULT 'page',
  UNIQUE KEY `cl_from` (`cl_from`,`cl_to`),
  KEY `cl_sortkey` (`cl_to`,`cl_type`,`cl_sortkey`,`cl_from`),
  KEY `cl_timestamp` (`cl_to`,`cl_timestamp`),
  KEY `cl_collation` (`cl_collation`)
) ENGINE=InnoDB DEFAULT CHARSET=binary
```

Runs with KEY_BLOCK_SIZE=16, 8, 4, and 2 were successful. However, the run with KEY_BLOCK_SIZE=1 failed.

```
mysql> ALTER TABLE test.categorylinks ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=1;
ERROR 1118 (42000): Row size too large. The maximum row size for the used table
type, not counting BLOBs, is 8126. You have to change some columns to TEXT or
BLOBs
```

Measurement:

```
mysql> SELECT * FROM  information_schema.innodb_cmp;
+-----------+--------------+-----------------+----------------+----------------+------------------+
| page_size | compress_ops | compress_ops_ok | compress_time | uncompress_ops | uncompress_time |
+-----------+--------------+-----------------+----------------+----------------+------------------+
|      1024 |            0 |               0 |              0 |              0 |                0 |
|      2048 |       334752 |          211889 |            140 |         100902 |               10 |
|      4096 |       158670 |          106536 |             96 |          40157 |                7 |
|      8192 |        21432 |           21378 |             11 |           8070 |                1 |
|     16384 |        12617 |           12617 |              5 |          19053 |                2 |
+-----------+--------------+-----------------+----------------+----------------+------------------+
```

```
mysql> SELECT * FROM  information_schema.innodb_cmpmem;
+-----------+----------------------+-----------+-----------+----------------+-----------------+
| page_size | buffer_pool_instance | pages_used | pages_free | relocation_ops | relocation_time |
+-----------+----------------------+-----------+-----------+----------------+-----------------+
|      1024 |                    0 |         0 |          0 |              0 |                0 |
|      2048 |                    0 |     14122 |          0 |              0 |                0 |
|      4096 |                    0 |      6597 |          0 |              0 |                0 |
|      8192 |                    0 |      5178 |          1 |              0 |                0 |
|     16384 |                    0 |      2313 |          0 |              0 |                0 |
+-----------+----------------------+-----------+-----------+----------------+-----------------+
```

Table Config	Size [M]	INSERT [sec]	Ops OK [%]	Comment
Z-	96	45	n/a	
Z+B16	96	52	100	
Z+B8	48	55	99	
Z+B4	31	208	67	
Z+B2	32	309	63	
Z+B1				Failed

Discussion: categorylinks is indeed a good candidate for compression. The Z+B8 configuration achieves 50% compression, at a cost of only 22% greater INSERT time and only 1% failure rate for compression operations. While Z+B4 achieves even greater compression, it comes at a cost of over four-fold greater INSERT time and a 33% failure rate for compression operations. This may be acceptable if INSERTion is done once a month.

Recommendation: Use ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4.

G.8.5 langlinks

This looks like a medium candidate for compression.

```
mysql> SHOW CREATE TABLE simplewiki.langlinks\G
*************************** 1. row ***************************
       Table: langlinks
Create Table: CREATE TABLE `langlinks` (
  `ll_from` int(10) unsigned NOT NULL DEFAULT '0',
  `ll_lang` varbinary(20) NOT NULL DEFAULT '',
  `ll_title` varbinary(255) NOT NULL DEFAULT '',
  UNIQUE KEY `ll_from` (`ll_from`,`ll_lang`),
  KEY `ll_lang` (`ll_lang`,`ll_title`)
) ENGINE=InnoDB DEFAULT CHARSET=binary
```

```
mysql> SELECT * FROM  information_schema.innodb_cmp;
+-----------+--------------+-----------------+----------------+------------------+-----------------+
| page_size | compress_ops | compress_ops_ok | compress_time | uncompress_ops | uncompress_time |
+-----------+--------------+-----------------+----------------+------------------+-----------------+
|      1024 |      3534262 |         1790669 |            941 |          3365249 |             175 |
|      2048 |      1696760 |          922050 |            692 |          2159232 |             174 |
|      4096 |       839084 |          483681 |            475 |          1618030 |             213 |
|      8192 |       348713 |          218493 |            277 |          1271866 |             287 |
|     16384 |        51301 |           51301 |             26 |          1165381 |             264 |
+-----------+--------------+-----------------+----------------+------------------+-----------------+
```

```
mysql> SELECT * FROM  information_schema.innodb_cmpmem;
+-----------+----------------------+------------+------------+----------------+-----------------+
| page_size | buffer_pool_instance | pages_used | pages_free | relocation_ops | relocation_time |
+-----------+----------------------+------------+------------+----------------+-----------------+
|      1024 |                    0 |      31287 |          4 |          91655 |               0 |
|      2048 |                    0 |      63637 |          3 |          35806 |               0 |
|      4096 |                    0 |      33517 |          1 |          14789 |               0 |
|      8192 |                    0 |      17027 |          1 |           5759 |               0 |
|     16384 |                    0 |      13936 |          0 |              0 |               0 |
+-----------+----------------------+------------+------------+----------------+-----------------+
```

Table Config	Size [M]	INSERT [sec]	Ops OK [%]	Comment
Z-	352	527	n/a	
Z+B16	352	1061	100	
Z+B8	184	2023	62	
Z+B4	180	2637	57	
Z+B2	204	3817	54	
Z+B1	244	6554	50	

Discussion: langlinks is indeed a medium candidate for compression. The Z-B8 configuration uses 48% less disk space, but this comes at a cost of nearly four-fold greater INSERT time, with a 38% failure rate for the compression operations. This may be acceptable if INSERTion is done monthly.

Recommendation: Use ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4.

G.8.6 pagelinks

This looks like a medium candidate for compression.

```
mysql> SHOW CREATE TABLE simplewiki.pagelinks\G
*************************** 1. row ***************************
       Table: pagelinks
Create Table: CREATE TABLE `pagelinks` (
  `pl_from` int(10) unsigned NOT NULL DEFAULT '0',
  `pl_namespace` int(11) NOT NULL DEFAULT '0',
  `pl_title` varbinary(255) NOT NULL DEFAULT '',
  UNIQUE KEY `pl_from` (`pl_from`,`pl_namespace`,`pl_title`),
  UNIQUE KEY `pl_namespace` (`pl_namespace`,`pl_title`,`pl_from`)
) ENGINE=InnoDB DEFAULT CHARSET=binary
```

```
mysql> SELECT * FROM  information_schema.innodb_cmp;
+-----------+--------------+-----------------+----------------+------------------+------------------+
| page_size | compress_ops | compress_ops_ok | compress_time | uncompress_ops | uncompress_time |
+-----------+--------------+-----------------+----------------+------------------+------------------+
|      1024 |      2602824 |         1502690 |            478 |          1468174 |               90 |
|      2048 |      1288533 |          794597 |            363 |           878152 |               90 |
|      4096 |       600328 |          388146 |            276 |           532090 |               92 |
|      8192 |       176814 |          120716 |             96 |           514791 |               93 |
|     16384 |        41137 |           41137 |             10 |           582956 |               95 |
+-----------+--------------+-----------------+----------------+------------------+------------------+
```

```
mysql> SELECT * FROM  information_schema.innodb_cmpmem;
+-----------+----------------------+-----------+-----------+----------------+-----------------+
| page_size | buffer_pool_instance | pages_used | pages_free | relocation_ops | relocation_time |
+-----------+----------------------+-----------+-----------+----------------+-----------------+
|      1024 |                    0 |      27220 |         16 |          29314 |               0 |
|      2048 |                    0 |      20760 |         16 |          13715 |               0 |
|      4096 |                    0 |       7934 |         16 |           7101 |               0 |
|      8192 |                    0 |      10059 |         15 |           4654 |               0 |
|     16384 |                    0 |         49 |          0 |              0 |               0 |
+-----------+----------------------+-----------+-----------+----------------+-----------------+
```

Table Config	Size [M]	INSERT [sec]	Ops OK [%]	Comment
Z-	296	316	n/a	
Z+B16	296	480	100	
Z+B8	156	742	68	
Z+B4	120	1178	64	
Z+B2	124	1617	61	
Z+B1	144	2533	57	

Discussion: This is a good candidate for compression.

Recommendation: Use ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4.

G.8.7 templatelinks

This looks like a medium candidate for compression.

```
mysql> SHOW CREATE TABLE simplewiki.templatelinks\G
*************************** 1. row ***************************
       Table: templatelinks
Create Table: CREATE TABLE `templatelinks` (
  `tl_from` int(10) unsigned NOT NULL DEFAULT '0',
  `tl_namespace` int(11) NOT NULL DEFAULT '0',
  `tl_title` varbinary(255) NOT NULL DEFAULT '',
  UNIQUE KEY `tl_from` (`tl_from`,`tl_namespace`,`tl_title`),
  UNIQUE KEY `tl_namespace` (`tl_namespace`,`tl_title`,`tl_from`)
) ENGINE=InnoDB DEFAULT CHARSET=binary
```

```
mysql> SELECT * FROM  information_schema.innodb_cmp;
+-----------+--------------+-----------------+---------------+----------------+------------------+
| page_size | compress_ops | compress_ops_ok | compress_time | uncompress_ops | uncompress_time |
+-----------+--------------+-----------------+---------------+----------------+------------------+
|      1024 |       636964 |          394930 |           114 |         120886 |                7 |
|      2048 |       312377 |          204010 |            91 |          64492 |                6 |
|      4096 |       107818 |           70436 |            40 |          25451 |                4 |
|      8192 |        29050 |           24158 |            14 |           7182 |                1 |
|     16384 |         9128 |            9128 |             1 |           4807 |                0 |
+-----------+--------------+-----------------+---------------+----------------+------------------+
```

```
mysql> SELECT * FROM  information_schema.innodb_cmpmem;
+-----------+----------------------+------------+------------+----------------+-----------------+
| page_size | buffer_pool_instance | pages_used | pages_free | relocation_ops | relocation_time |
+-----------+----------------------+------------+------------+----------------+-----------------+
|      1024 |                    0 |      24925 |         15 |           7562 |               0 |
|      2048 |                    0 |      13068 |          2 |              0 |               0 |
|      4096 |                    0 |       6447 |          1 |              0 |               0 |
|      8192 |                    0 |       4202 |          0 |              0 |               0 |
|     16384 |                    0 |       4166 |          0 |              0 |               0 |
+-----------+----------------------+------------+------------+----------------+-----------------+
```

Table Config	Size [M]	INSERT [sec]	Ops OK [%]	Comment
Z-	76	38	n/a	
Z+B16	80	68	100	
Z+B8	40	91	83	
Z+B4	29	125	65	
Z+B2	29	212	65	
Z+B1	36	279	62	

Discussion: This is a good candidate for compression.

Recommendation: Use ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4.

G.8.8 text

This looks like a good candidate for compression.

```
mysql> SHOW CREATE TABLE simplewiki.text\G
*************************** 1. row ***************************
       Table: text
Create Table: CREATE TABLE `text` (
  `old_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `old_text` mediumblob NOT NULL,
  `old_flags` tinyblob NOT NULL,
  PRIMARY KEY (`old_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=binary MAX_ROWS=10000000 AVG_ROW_LENGTH=10240;
```

```
mysql> SELECT * FROM  information_schema.innodb_cmp;
+-----------+--------------+-----------------+---------------+----------------+------------------+
| page_size | compress_ops | compress_ops_ok | compress_time | uncompress_ops | uncompress_time |
+-----------+--------------+-----------------+---------------+----------------+------------------+
|      1024 |       228554 |           99778 |            44 |          64659 |                3 |
|      2048 |       285813 |          124585 |           109 |          83294 |                8 |
|      4096 |       275680 |          127268 |           141 |          77459 |               10 |
|      8192 |        63064 |           62992 |            24 |           1851 |                0 |
|     16384 |        35393 |           35393 |             5 |           2833 |                0 |
+-----------+--------------+-----------------+---------------+----------------+------------------+
```

```
mysql> SELECT * FROM  information_schema.innodb_cmpmem;
+-----------+----------------------+------------+------------+----------------+-----------------+
| page_size | buffer_pool_instance | pages_used | pages_free | relocation_ops | relocation_time |
+-----------+----------------------+------------+------------+----------------+-----------------+
|      1024 |                    0 |      26434 |         16 |          18333 |               0 |
|      2048 |                    0 |      15346 |         16 |           7051 |               0 |
|      4096 |                    0 |       8596 |         15 |           3082 |               0 |
|      8192 |                    0 |       3121 |         15 |           2254 |               0 |
|     16384 |                    0 |       4333 |          0 |              0 |               0 |
+-----------+----------------------+------------+------------+----------------+-----------------+
```

Table Config	Size [M]	INSERT [sec]	Ops OK [%]	Comment
Z-	476	97	n/a	
Z+B16	456	262	100	
Z+B8	240	236	99	
Z+B4	220	397	46	
Z+B2	244	359	43	
Z+B1	252	406	43	

Discussion: `text` is indeed a good candidate for compression. The `Z+B8` configuration achieves 50% compression, at a cost of 2.4 times the `INSERT` time and only 1% failure rate for compression operations. While `Z+B4` achieves marginally better compression, it comes at a cost of 54% failure rate for compression operations. This may be acceptable if `INSERT`ion is done monthly.

Recommendation: Use `ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4`.

G.8.9 `enwiki.image`

Most wikis reference image files from the `commonswiki`. This means that most wikis have a very small `image` database table which is not worth compressing. The `commonswiki.image` table, on the other hand, is too large for quick experiments (but see §G.9, Experiments with `commonswiki.image`). The `enwiki` is however a good size for our purposes.

The `image` table looks like a good candidate for compression.

```
mysql> SHOW CREATE TABLE enwiki.image\G
*************************** 1. row ***************************
       Table: image
Create Table: CREATE TABLE `image` (
  `img_name` varbinary(255) NOT NULL DEFAULT '',
  `img_size` int(10) unsigned NOT NULL DEFAULT '0',
  `img_width` int(11) NOT NULL DEFAULT '0',
  `img_height` int(11) NOT NULL DEFAULT '0',
  `img_metadata` mediumblob NOT NULL,
  `img_bits` int(11) NOT NULL DEFAULT '0',
  `img_media_type` enum('UNKNOWN','BITMAP','DRAWING','AUDIO','VIDEO','MULTIMEDIA','OFFICE','TEXT','
  `img_major_mime` enum('unknown','application','audio','image','text','video','message','model','m
  `img_minor_mime` varbinary(100) NOT NULL DEFAULT 'unknown',
  `img_description` tinyblob NOT NULL,
  `img_user` int(10) unsigned NOT NULL DEFAULT '0',
  `img_user_text` varbinary(255) NOT NULL,
  `img_timestamp` varbinary(14) NOT NULL DEFAULT '',
  `img_sha1` varbinary(32) NOT NULL DEFAULT '',
  PRIMARY KEY (`img_name`),
  KEY `img_usertext_timestamp` (`img_user_text`,`img_timestamp`),
  KEY `img_size` (`img_size`),
  KEY `img_timestamp` (`img_timestamp`),
  KEY `img_sha1` (`img_sha1`)
) ENGINE=InnoDB DEFAULT CHARSET=binary;
```

Experimental method:

```
shell$ rsync rsync://ftpmirror.your.org/wikimedia-dumps/enwiki/20130102/enwiki-20130102-image.sql.g
shell$ gunzip enwiki-20130102-image.sql.gz
shell$ mysql --host=localhost --user=root --password --database=test < enwiki-20130102-image.sql
mysql> select * from  information_schema.innodb_cmp_reset;
mysql> select * from  information_schema.innodb_cmpmem_reset;
```

```
mysql> CREATE TABLE test.image0 LIKE test.image;
mysql> INSERT INTO test.image0 SELECT * FROM test.image;
Query OK, 807542 rows affected (15 min 20.25 sec)
Records: 807542  Duplicates: 0  Warnings: 0
root-shell# ls -lh /var/lib/mysql/test/*.ibd
mysql> DROP TABLE test.image0;
```

```
mysql> CREATE TABLE test.image0 LIKE test.image;
mysql> ALTER TABLE test.image0 ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=16;
mysql> INSERT INTO test.image0 SELECT * FROM test.image;
root-shell# ls -lh /var/lib/mysql/test/*.ibd
mysql> select * from  information_schema.innodb_cmp;
mysql> select * from  information_schema.innodb_cmpmem;
mysql> DROP TABLE test.image0;
```

and so on with KEY_BLOCK_SIZE=8, 4, 2, and 1.

Measurement:

Table G.1: Database table size [M] v. ROW_FORMAT and KEY_BLOCK_SIZE

Table	Z-	Z+B16	Z+B8	Z+B4	Z+B2	Z+B1
categorylinks	96	96	48	31	32	fail
externallinks						
image	748	744	392	348	412	fail
imagelinks						
langlinks	352	352	184	180	204	244
pagelinks	296	296	156	120	124	144
templatelinks	76	80	40	29	29	36
text	476	456	240	220	244	252

```
mysql> SELECT * FROM  information_schema.innodb_cmp;
+-----------+--------------+-----------------+---------------+----------------+------------------+
| page_size | compress_ops | compress_ops_ok | compress_time | uncompress_ops | uncompress_time |
+-----------+--------------+-----------------+---------------+----------------+------------------+
|      1024 |            0 |               0 |             0 |              0 |                0 |
|      2048 |      2862744 |         1411589 |           828 |        2948193 |              257 |
|      4096 |      1358140 |          736381 |           608 |        2228324 |              319 |
|      8192 |       319821 |          244692 |           203 |        1783801 |              410 |
|     16384 |        94043 |           94043 |            23 |        1916914 |              404 |
+-----------+--------------+-----------------+---------------+----------------+------------------+
```

```
mysql> SELECT * FROM  information_schema.innodb_cmpmem;
+-----------+----------------------+------------+------------+----------------+-----------------+
| page_size | buffer_pool_instance | pages_used | pages_free | relocation_ops | relocation_time |
+-----------+----------------------+------------+------------+----------------+-----------------+
|      1024 |                    0 |          0 |         16 |              0 |               0 |
|      2048 |                    0 |      62908 |         16 |          22401 |               0 |
|      4096 |                    0 |      35002 |          0 |           6722 |               0 |
|      8192 |                    0 |      19276 |         16 |           2558 |               0 |
|     16384 |                    0 |      13984 |          0 |              0 |               0 |
+-----------+----------------------+------------+------------+----------------+-----------------+
```

Table Config	Size [M]	INSERT [sec]	Ops OK [%]	Comment
Z-	748	920	n/a	
Z+B16	744	1337	100	
Z+B8	392	2284	76	
Z+B4	348	3907	54	
Z+B2	412	4916	49	
Z+B1				Failed

Discussion: image is indeed a medium candidate for compression. With the Z+B8 configuration, image.ibd is 52% the uncompressed size, at a cost of 2.5 times the INSERT time and only 24% failure rate for compression operations. While Z+B4 achieves marginally better compression, it comes at a cost of 54% failure rate for compression operations. This may be acceptable if INSERTion is done once a month.

Recommendation: Use ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4.

G.8.10 Conclusions

In each case shown above, best compression was achieved with page size 4K. See §G.1, Database table size [M] v. ROW_FORMAT and KEY_BLOCK_SIZE.

G.9 Experiments with commonswiki.image

Purpose: Determine how to speed up the INSERTion of the commonswiki.image database table.

Motivation: commonswiki.image contains image file metadata needed by every wiki that references image files from the commonswiki. The following is known:

- it is large
  - commonswiki-20130113-image.sql.gz is 7.4G;
  - commonswiki-20130113-image.sql is 31G;
  - commonswiki/image.ibd is 39G.
- INSERT INTO ‘image‘ VALUES ... is disk I/O bound;
- it should be a medium candidate for compression.

Intended benefit: Reduce the time required to build a mirror farm that makes use of shared images.

G.9.1 Theory

G.9.1.1 Disk Write Speed

Most disks in laptops use some version of SATA interface:

```
root-shell# hdparm -I /dev/sda | grep Transport
       Transport:          Serial, SATA 1.0a, SATA II Extensions, SATA Rev 2.5, SATA Rev 2.6
```

For such disks, we expect the electronic interface to transport data at 300M/s. However, the hardware (platters and read/write heads) are far less capable. Performance can vary widely. For cylinders near the edge of the platter, we may achieve 90M/s R/W. For cylinders near the spindle we may expect 30M/s R/W. Actually, this is the best case, because it assumes that the read/write heads stay on one cylinder as the platters rotate. If the filesystem is fragmented, we can expect 1-2M/s.

InnoDB tries to flush the log buffer and log files once per second. This is set with innodb_flush_log_at_trx_commit=0. Let us increase the size of the log buffer and log files to match the best case, and see if performance improves.

```
root-shell# cat /etc/mysql/conf.d/wp-mirror.cnf | grep log
innodb_log_buffer_size          =  80M # default   8M.
innodb_log_file_size            =  50M # default   5M.
# for Durability (the ‘D’ in ‘ACID compliance’)        (0=flush log buff 1/sec)
innodb_flush_log_at_trx_commit  =    0 # default    1. (2=write log file 1/sec)
sync_binlog                     =    0 # default    0.
```

G.9.1.2 Fast index creation

> Overview of Fast Index Creation
>
> With MySQL 5.5 and higher, or MySQL 5.1 with the InnoDB Plugin, creating and dropping secondary indexes for InnoDB tables is much faster than before. Historically, adding or dropping an index on a table with existing data could be very slow. The CREATE INDEX and DROP INDEX statements worked by creating a new, empty table defined with the requested set of indexes, then copying the existing rows to the new table one-by-one, updating the indexes as the rows are inserted. After all rows from the original table were copied, the old table was dropped and the copy was renamed with the name of the original table.
>
> The performance speedup for fast index creation applies to secondary indexes, not to the primary key index. The rows of an InnoDB table are stored in a clustered index organized based on the primary key, forming what some database systems call

> an "index-organized table". Because the table structure is so closely tied to the primary key, redefining the primary key still requires copying the data.
>
> This new mechanism also means that you can generally speed the overall process of creating and loading an indexed table by creating the table with only the clustered index, and adding the secondary indexes after the data is loaded.
>
> MySQL 5.5 Reference Manual, Section 14.4.2, Fast Index Creation in the InnoDB Storage Engine, p.1621

This presents an opportunity to experiment. Instead of INSERTing the data, the idea is this: first DROP the secondary indices, then INSERT the data, and finally ADD the secondary indices:

```
mysql> ALTER TABLE commonswiki.image
    -> DROP INDEX img_usertext_timestamp, DROP INDEX img_size,
    -> DROP INDEX img_timestamp, DROP INDEX img_sha1;
root-shell# wp-mirror --delete all
root-shell# wp-mirror --mirror
mysql> ALTER TABLE commonswiki.image
    -> ADD INDEX `img_usertext_timestamp` (`img_user_text`,`img_timestamp`),
    -> ADD INDEX `img_size` (`img_size`),
    -> ADD INDEX `img_timestamp` (`img_timestamp`),
    -> ADD INDEX `img_sha1` (`img_sha1`);
```

G.9.1.3 LOAD DATA INFILE

> Speed of INSERT Statements
>
> When loading a table from a text file, use LOAD DATA INFILE. This is usually 20 times faster than using INSERT statements.
>
> MySQL 5.5 Reference Manual, Section 8.2.2 Optimizing DML Statements

The possibility of such a speed increase must be tested.

G.9.2 Experimental Method

Overall method: Measure the time required to load 15m commonswiki.image records v. a variety of configurations.

Run: The file commonswiki-20130113-image.sql.gz was downloaded from WMF (or a mirror site). This file contains 27,570 INSERT statements, each with about 500 sets of values. This file, when decompressed, expands from 7G to 31G (4.5x). This was split these into 2,757 schunks of 10 INSERT statements each. These schunks were then loaded into the database table.

Measurement: A timestamp was recorded upon completion of each schunk. WP-MIRROR 0.6 automates both the run and measurement.

Experimental method:

1. **Setup:**
   - Delete working files and state information:
     ```
     root-shell# wp-mirror --delete all
     ```
   - Set table options where KEY_BLOCK_SIZE = 1, 2, 4, 8, or 16:
     ```
     mysql> TRUNCATE TABLE commonswiki.image;
     mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=8;
     ```
   - Reset compression statistics tables:
     ```
     mysql> SELECT * FROM information_schema.innodb_cmp_reset;
     mysql> SELECT * FROM information_schema.innodb_cmpmem_reset;
     ```
2. **Run:**
   - In one terminal run the test:

```
root-shell# wp-mirror --mirror
```

- In another terminal collect data:

```
mysql> SELECT timestamp, name FROM wpmirror.file
    -> WHERE type='schunk' AND state IN ('done','fail') AND name LIKE '%000-c%';
mysql> SELECT * FROM information_schema.innodb_cmp;
mysql> SELECT * FROM information_schema.innodb_cmpmem;
root-shell# ls /var/lib/mysql/commonswiki/image.ibd
```

3. **Tear-down:** None.

Measurement: The raw data looks like this:

```
mysql> SELECT timestamp, name FROM wpmirror.file WHERE type='schunk'
    -> AND state IN ('done','fail') AND name LIKE '%000-c%';
+---------------------+-------------------------------------------------------+
| timestamp           | name                                                  |
+---------------------+-------------------------------------------------------+
| 2013-01-21 21:55:25 | commonswiki-20130113-image-p000000000-c000000010.sql |
| 2013-01-21 22:43:41 | commonswiki-20130113-image-p000001000-c000000010.sql |
| 2013-01-21 23:40:32 | commonswiki-20130113-image-p000002000-c000000010.sql |
| 2013-01-22 00:55:51 | commonswiki-20130113-image-p000003000-c000000010.sql |
| 2013-01-22 02:18:43 | commonswiki-20130113-image-p000004000-c000000010.sql |
...
+---------------------+-------------------------------------------------------+
```

However the following information, which is derived from the raw data, is more useful for comparing one run with another:

```
mysql> SET @t0=(SELECT TO_SECONDS(MIN(timestamp)) FROM wpmirror.file
    ->          WHERE type='schunk' AND state IN ('done','fail'));
mysql> SELECT ROUND((TO_SECONDS(timestamp)-@t0)/1000,1) AS 'time [ks]',
    ->        ROUND(CAST(TRIM(LEADING 'p' FROM SUBSTRING_INDEX(SUBSTRING_INDEX(name,'-',-2),'-',1)
    ->                   AS UNSIGNED)/1000,1) AS 'inserts [k]'
    -> FROM wpmirror.file WHERE type='schunk' AND state IN ('done','fail') AND name LIKE '%000-c%'
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       2.9 |         1.0 |
|       6.3 |         2.0 |
|      10.8 |         3.0 |
|      15.8 |         4.0 |
...
+-----------+-------------+
```

Coding: We extend the coding used in §E.7, Experiments with InnoDB:

- **D1** /var/lib/mysql/[database]/[table].ibd and /var/lib/mysql/ib_logfile* are on the same disk,
- **F+** /var/lib/mysql/[database]/[table].ibd are files on a file system,
- **L10** /var/lib/mysql/ib_logfile* are 10M each,
- **P1024** InnoDB_buffer_pool is 1024M,
- **H-** hugepages are not used,
- **Z-** data is uncompressed (ROW_FORMAT=COMPACT),
- **Z+** data is compressed with zlib (ROW_FORMAT=COMPRESSED), and
- **B16** KEY_BLOCK_SIZE is 16K.

G.9.3 EXPERIMENT.0 Baseline configuration

Code: D1F+L10P1024H-Z-

Configuration files:

```
root-shell# /etc/mysql/conf.d/wp-mirror.cnf
[mysqld]
# time zone UTC
default-time-zone               = UTC
# transactional (ACID) storage engine.
default-storage-engine          = innodb
# UTF-8 character set and collation.
character-set-server            = utf8
collation-server                = utf8_general_ci

# InnoDB

# to save disk space (compress tables using 'zlib')
innodb_file_per_table           =    1 # default  OFF. Create .ibd file/table
innodb_file_format              = Barracuda # default Antelope.
innodb_file_format_check        =    1 # default   ON.
innodb_strict_mode              =    1 # default    0.
# increase size of buffer pool and log files for speed
innodb_buffer_pool_size         =1024M # default 128M.
innodb_log_buffer_size          =   8M # default   8M.
innodb_log_file_size            =  10M # default   5M.
innodb_additional_mem_pool_size =   8M # default   8M. (data dict, internals)
# for Durability (the 'D' in 'ACID compliance')
innodb_flush_log_at_trx_commit  =    2 # default    1. (2=write log file 1/sec)
sync_binlog                     =    0 # default    0.

# Other

lower_case_table_names          =    1 # default    0. (use 1 for InnoDB)
# <meta.wikimedia.org/wiki/Data_dumps> recommends 20M.
max_allowed_packet              =  64M # default  16M.
query_cache_size                =  16M # default  16M.
query_cache_type                =    1 # default   ON.

[mysql]
default-character-set           = utf8
# <meta.wikimedia.org/wiki/Data_dumps> recommends 20M.
max_allowed_packet              =  64M # default  16M.
```

The baseline image table has the Antelope format:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPACT;
```

Measurement: /var/lib/mysql/commonswiki.image.ibd is 39G.

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       2.9 |         1.0 |
|       6.3 |         2.0 |
|      10.8 |         3.0 |
|      15.8 |         4.0 |
|      22.5 |         5.0 |
|      29.7 |         6.0 |
|      34.7 |         7.0 |
|      41.9 |         8.0 |
|      48.6 |         9.0 |
|      59.3 |        10.0 |
|      67.9 |        11.0 |
|      77.0 |        12.0 |
|      85.2 |        13.0 |
|      91.5 |        14.0 |
|     100.1 |        15.0 |
|     108.1 |        16.0 |
|     117.4 |        17.0 |
|     126.4 |        18.0 |
|     136.8 |        19.0 |
|     144.1 |        20.0 |
|     153.8 |        21.0 |
|     163.4 |        22.0 |
|     173.3 |        23.0 |
|     180.5 |        24.0 |
|     189.5 |        25.0 |
|     198.5 |        26.0 |
|     213.0 |        27.0 |
+-----------+-------------+
```

G.9.4 EXPERIMENT.1 ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=16

Code: D1F+L10P1024H-Z+B16

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf is the same as the baseline. The image table is now compressed and has 16K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=16;
```

Measurement: This run was terminated after 10,020 INSERTs, when it became clear that this configuration had lower performance than the baseline. /var/lib/mysql/commonswiki/image.ibd is 12G.

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       4.6 |         1.0 |
|       8.9 |         2.0 |
|      14.0 |         3.0 |
|      19.2 |         4.0 |
|      26.1 |         5.0 |
|      33.3 |         6.0 |
|      40.7 |         7.0 |
|      50.7 |         8.0 |
|      59.3 |         9.0 |
|      68.9 |        10.0 |
+-----------+-------------+
```

```
mysql> SELECT * FROM information_schema.innodb_cmp WHERE page_size=16384\G
*************************** 1. row ***************************
      page_size: 16384
   compress_ops: 1231447
compress_ops_ok: 1231447
  compress_time: 237
 uncompress_ops: 10648260
uncompress_time: 1974
mysql> SELECT * FROM information_schema.innodb_cmpmem WHERE page_size=16384\G
*************************** 1. row ***************************
          page_size: 16384
buffer_pool_instance: 0
         pages_used: 56280
         pages_free: 0
     relocation_ops: 0
    relocation_time: 0
```

G.9.5 EXPERIMENT.2 ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=8

Code: D1F+L10P1024H-Z+B8

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf is the same as the baseline. The image table is compressed and has 8K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=8;
```

Measurement: This run was terminated after 10,000 INSERTs. /var/lib/mysql/commonswiki/image.ibd 6.7G.

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       5.1 |         1.0 |
|       9.9 |         2.0 |
|      15.2 |         3.0 |
|      20.9 |         4.0 |
|      28.0 |         5.0 |
|      36.2 |         6.0 |
|      43.8 |         7.0 |
|      54.1 |         8.0 |
|      64.4 |         9.0 |
|      79.7 |        10.0 |
+-----------+-------------+
```

```
mysql> SELECT * FROM information_schema.innodb_cmp WHERE page_size=8192\G
*************************** 1. row ***************************
      page_size: 8192
   compress_ops: 3226025
compress_ops_ok: 2576819
  compress_time: 1503
 uncompress_ops: 10822547
uncompress_time: 2096
mysql> SELECT * FROM information_schema.innodb_cmpmem WHERE page_size=8192\G
*************************** 1. row ***************************
           page_size: 8192
buffer_pool_instance: 0
          pages_used: 107942
          pages_free: 2
      relocation_ops: 3482
     relocation_time: 8
```

G.9.6 EXPERIMENT.3 ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4

Code: D1F+L10P1024H-Z+B4

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf is the same as the baseline. The image table is compressed and has 4K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4;
```

Measurement: This run was terminated after 10,000 INSERTs. /var/lib/mysql/commonswiki/image.ibd is 5.3G.

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       6.9 |         1.0 |
|      12.4 |         2.0 |
|      18.5 |         3.0 |
|      25.6 |         4.0 |
|      34.8 |         5.0 |
|      46.9 |         6.0 |
|      59.7 |         7.0 |
|      79.8 |         8.0 |
|      96.1 |         9.0 |
|     109.8 |        10.0 |
+-----------+-------------+
```

```
mysql> SELECT * FROM information_schema.innodb_cmp WHERE page_size=4096\G
mysql> SELECT * FROM information_schema.innodb_cmpmem WHERE page_size=4096\G
```

G.9.7 EXPERIMENT.4 innodb_log_buffer_size=80M, innodb_log_file_size=50M

Code: D1F+L50P1024H-Z-

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf differs from the baseline as follows:

```
innodb_log_buffer_size          =  80M # default   8M.
innodb_log_file_size            =  50M # default   5M.
innodb_flush_log_at_trx_commit  =    0 # default    1. (0=flush log buff 1/sec)
```

Measurement: This run was terminated after 10,000 INSERTs.

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       1.3 |         1.0 |
|       3.1 |         2.0 |
|       5.3 |         3.0 |
|      10.0 |         4.0 |
|      13.7 |         5.0 |
|      18.8 |         6.0 |
|      23.1 |         7.0 |
|      29.0 |         8.0 |
|      34.2 |         9.0 |
|      39.7 |        10.0 |
+-----------+-------------+
```

G.9.8 EXPERIMENT.5 innodb_log_buffer_size=80M, innodb_log_file_size=512M

Code: D1F+L512P1024H-Z-

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf differs from the baseline as follows:

```
innodb_log_buffer_size          =  80M # default   8M. (max disk write speed)
innodb_log_file_size            = 512M # default   5M. (1/2 buffer pool size)
innodb_flush_log_at_trx_commit  =    0 # default    1. (0=flush log buff 1/sec)
```

Measurement: This run was terminated after 10,000 INSERTs.

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       1.0 |         1.0 |
|       1.8 |         2.0 |
|       2.7 |         3.0 |
|       3.9 |         4.0 |
|       5.4 |         5.0 |
|       7.2 |         6.0 |
|       9.1 |         7.0 |
|      12.2 |         8.0 |
|      15.4 |         9.0 |
|      18.1 |        10.0 |
+-----------+-------------+
```

G.9.9 EXPERIMENT.6 innodb_log_buffer_size=80M, innodb_log_file_size=512M, ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4

Code: D1F+L512P1024H-Z+B4

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf differs from the baseline as follows:

```
innodb_log_buffer_size          =  80M # default   8M. (max disk write speed)
innodb_log_file_size            = 512M # default   5M. (1/2 buffer pool size)
innodb_flush_log_at_trx_commit  =    0 # default    1. (0=flush log buff 1/sec)
```

The image table is compressed and has 4K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4;
```

Some new parameters are introduced for this and the remaining experiments. The values for the baseline, and all runs prior to this one, are:

```
(defparameter *mirror-alter-table-timeout-sec-max*    15000)
(defparameter *mirror-dchunk-load-timeout-sec-max*     3000)
(defparameter *mirror-dchunk-page-count*                 10)
(defparameter *mirror-innodb-fast-index-creation*       nil)
(defparameter *mirror-innodb-table-key-block-size-list* nil)
(defparameter *mirror-schunk-page-count*                 10)
(defparameter *mirror-split-sql*                    "schunk")
```

The difference for this run is that several tables will use compression with 4K page size:

```
(defparameter *mirror-innodb-table-key-block-size-list*
  '(("categorylinks" . 4) ("externallinks" . 4) ("image"     . 4)
    ("imagelinks"    . 4) ("langlinks"     . 4) ("pagelinks" . 4)
    ("templatelinks" . 4) ("text"          . 4)))
```

Measurement: commonswiki-20130205-image.sql.gz of 7.7G resulted in /var/lib/mysql/commonswiki/image.ibd of 15G.

```
mysql> SELECT data_length,index_length FROM information_schema.tables
    -> WHERE table_schema='commonswiki' AND table_name='image';
+-------------+--------------+
| data_length | index_length |
+-------------+--------------+
| 11219763200 |   3454533632 |
+-------------+--------------+
```

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       3.4 |         1.0 |
|       5.1 |         2.0 |
|       6.5 |         3.0 |
|       8.1 |         4.0 |
|      10.1 |         5.0 |
|      12.1 |         6.0 |
|      14.3 |         7.0 |
|      17.4 |         8.0 |
|      20.4 |         9.0 |
|      23.8 |        10.0 |
|      27.5 |        11.0 |
|      32.0 |        12.0 |
|      42.3 |        13.0 |
|      46.4 |        14.0 |
|      52.2 |        15.0 |
|      56.5 |        16.0 |
|      60.9 |        17.0 |
|      67.8 |        18.0 |
|      75.1 |        19.0 |
|      81.4 |        20.0 |
|      87.7 |        21.0 |
|      95.6 |        22.0 |
|     101.8 |        23.0 |
|     108.0 |        24.0 |
|     112.7 |        25.0 |
|     118.5 |        26.0 |
|     128.2 |        27.0 |
|     140.1 |        28.0 |
+-----------+-------------+
```

At the end of the run, the InnoDB Change Buffer (formerly called the InnoDB Insert Buffer) occupied about 15% of the buffer pool, and took an additional 3ks to process.

```
mysql> TRUNCATE TABLE commonswiki.image;
Query OK, 0 rows affected (12 min 21.22 sec)
```

The time vs. inserts data, when plotted in Figure G.1, InnoDB Experiments—Importing commonswiki.image Table, exhibits kinks which are due to swapping when other processes content for limited memory, and thereby disturb the smooth functioning of InnoDB. For this reason, the remaining runs will employ hugepages so that memory occupied by the InnoDB buffer pool will not be swapped.

G.9.10 EXPERIMENT.7 innodb_log_buffer_size=80M, innodb_log_file_size=512M, ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4

Code: D1F+L512P1024H+Z+B4

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf differs from the baseline as follows:

```
innodb_log_buffer_size          =  80M # default   8M. (max disk write speed)
innodb_log_file_size            = 512M # default   5M. (1/2 buffer pool size)
innodb_flush_log_at_trx_commit  =    0 # default    1. (0=flush log buff 1/sec)
large_pages                     =    1 # default  OFF.
```

The image table is compressed and has 4K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4;
```

The new WP-MIRROR parameters are same as the previous run:

```
(defparameter *mirror-innodb-fast-index-creation*       nil)
(defparameter *mirror-split-sql*                    "schunk")
```

The only difference with the previous run is that we now use hugepages.

Measurement: commonswiki-20130205-image.sql.gz of 7.7G resulted in /var/lib/mysql/commonswiki/image.ibd of 15G.

```
mysql> SELECT data_length,index_length FROM information_schema.tables
    -> WHERE table_schema='commonswiki' AND table_name='image';
+-------------+--------------+
| data_length | index_length |
+-------------+--------------+
| 11219763200 |   3454533632 |
+-------------+--------------+
```

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       1.1 |         1.0 |
|       2.3 |         2.0 |
|       3.5 |         3.0 |
|       4.9 |         4.0 |
|       6.8 |         5.0 |
|       8.8 |         6.0 |
|      11.2 |         7.0 |
|      14.4 |         8.0 |
|      17.7 |         9.0 |
|      21.5 |        10.0 |
|      25.7 |        11.0 |
|      30.9 |        12.0 |
|      36.5 |        13.0 |
|      40.2 |        14.0 |
|      46.4 |        15.0 |
|      50.5 |        16.0 |
|      55.9 |        17.0 |
|      63.9 |        18.0 |
|      71.2 |        19.0 |
|      78.2 |        20.0 |
|      81.4 |        21.0 |
|      87.2 |        22.0 |
|      94.6 |        23.0 |
|     101.7 |        24.0 |
|     107.2 |        25.0 |
|     114.2 |        26.0 |
|     121.1 |        27.0 |
|     130.6 |        28.0 |
+-----------+-------------+
```

```
mysql> TRUNCATE TABLE commonswiki.image;
Query OK, 0 rows affected (1 min 28.10 sec)
```

We see that the use of hugepages did not result in a significant performance gain. This we already knew from Table E.2, InnoDB Experimental Results—Performance Ratios. However, the system did run more smoothly. Without hugepages, users frequently perceived poor system response (pauses on the order of 10-100s) which made the system all but unusable for them.

G.9.11 EXPERIMENT.8 innodb_log_buffer_size=80M, innodb_log_file_size=512M, ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4, LOAD DATA INFILE

Code: D1F+L512P1024H+Z+B4, LOAD DATA INFILE

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf differs from the baseline as follows:

```
innodb_log_buffer_size          =  80M # default   8M. (max disk write speed)
innodb_log_file_size            = 512M # default   5M. (1/2 buffer pool size)
innodb_flush_log_at_trx_commit  =    0 # default    1. (0=flush log buff 1/sec)
large_pages                     =    1 # default  OFF.
```

The image table is compressed and has 4K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4;
```

For this run, the values of the new parameters are:

```
(defparameter *mirror-innodb-fast-index-creation*       nil)
(defparameter *mirror-split-sql*                    "dchunk")
```

LOAD DATA INFILE is done by preparing files, called dchunks, that contain data fields only; then loading each dchunk by specifying its absolute path:

```
mysql> LOAD DATA INFILE
    -> /var/lib/mediawiki/images/wp-mirror/commonswiki-20130124-image-p000000000-c000000010.dat'
    -> IGNORE INTO TABLE `commonswiki`.`image`
    -> FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY "'"
    -> LINES TERMINATED BY '),(';
```

Unfortunately, this failed for the following reason: The transactions piled up waiting for a table lock:

```
...
---TRANSACTION 277CA2, ACTIVE 451 sec inserting
mysql tables in use 1, locked 1
242 lock struct(s), heap size 31160, 457 row lock(s), undo log entries 4327
MySQL thread id 169064, OS thread handle 0x7f02aafeb700, query id 589273 localhost debian-sys-maint
LOAD DATA INFILE '/var/lib/mediawiki/images/wp-mirror/commonswiki-20130124-image-p000000010-c000000
---TRANSACTION 277BDC, ACTIVE 454 sec inserting
mysql tables in use 1, locked 1
228 lock struct(s), heap size 31160, 455 row lock(s), undo log entries 4776
MySQL thread id 168756, OS thread handle 0x7f02ab0e0700, query id 588081 localhost debian-sys-maint
LOAD DATA INFILE '/var/lib/mediawiki/images/wp-mirror/commonswiki-20130124-image-p000000000-c000000
```

after which some of these transactions failed due to timeout:

```
[ ok ]fsm-no-op commonswiki-20130124-image-p000001120-c000000010.dat
ERROR 1205 (HY000) at line 1: Lock wait timeout exceeded; try restarting transaction
```

then the number of failed transactions increased, and finally came a flood of error messages:

```
ERROR 1040 (08004): Too many connections
ERROR 1040 (HY000): Too many connections
ERROR 1040 (08004): Too many connections
...
```

Even after WP-MIRROR exited, InnoDB continued processing the backlog of transactions, occasionally emitting additional timeout errors, for an hour. Finally, we see that just under 1M of the 15.8M records were successfully loaded.

```
mysql> SELECT COUNT(*) FROM commonswiki.image;
+----------+
| count(*) |
+----------+
|   943800 |
+----------+
```

One solution is to make LOAD DATA INFILE a single transaction, and this is done by setting:

```
(defparameter *mirror-dchunk-page-count*     100000)
```

This was tried once successfully. However, on a second attempt there was a mishap that caused a ROLLBACK; and, as we soon discovered, rolling back such a large transaction takes days.

A better solution is to poll information_schema.innodb_trx and only start a LOAD DATA INFILE transaction if no other is running. In this manner the dchunks are processed *ad seriatim*; and while the performance is a little worse compared to the single large transaction, the ROLLBACK time, in the event of a mishap, is tolerable.

Measurement: commonswiki-20130205-image.sql.gz of 7.7G resulted in /var/lib/mysql/commonswiki/image.ibd of 7.2G.

```
mysql> SELECT data_length,index_length FROM information_schema.tables
    -> WHERE table_schema='commonswiki' AND table_name='image';
+-------------+--------------+
| data_length | index_length |
+-------------+--------------+
|  4642832384 |   2792095744 |
+-------------+--------------+
mysql> SELECT COUNT(*) FROM commonswiki.image;
+----------+
| count(*) |
+----------+
| 14869582 |
+----------+
1 row in set (17 min 48.23 sec)               <-- full table scan!
```

```
mysql> SET @t0=(SELECT TO_SECONDS(MIN(timestamp)) FROM wpmirror.file
    ->         WHERE type='dchunk' AND state IN ('done','fail'));
mysql> SELECT ROUND((TO_SECONDS(timestamp)-@t0)/1000,1) AS 'time [ks]',
    ->        ROUND(CAST(TRIM(LEADING 'p' FROM SUBSTRING_INDEX(SUBSTRING_INDEX(name,'-',-2),'-',1)
    ->                   AS UNSIGNED)/1000,1) AS 'inserts [k]'
    -> FROM wpmirror.file WHERE type='dchunk' AND state IN ('done','fail') AND name LIKE '%000-c%'
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       1.3 |         1.0 |
|       2.3 |         2.0 |
|       3.5 |         3.0 |
|       4.8 |         4.0 |
|       6.4 |         5.0 |
|       7.9 |         6.0 |
|       9.5 |         7.0 |
|      11.4 |         8.0 |
|      13.3 |         9.0 |
|      15.8 |        10.0 |
|      18.4 |        11.0 |
|      21.3 |        12.0 |
|      25.0 |        13.0 |
|      26.8 |        14.0 |
|      30.6 |        15.0 |
|      33.2 |        16.0 |
|      36.1 |        17.0 |
|      39.8 |        18.0 |
|      44.1 |        19.0 |
|      48.1 |        20.0 |
|      49.7 |        21.0 |
|      53.3 |        22.0 |
|      57.6 |        23.0 |
|      62.0 |        24.0 |
|      65.1 |        25.0 |
|      71.2 |        26.0 |
|      74.9 |        27.0 |
|      80.5 |        28.0 |
+-----------+-------------+
```

At the end of the run, the InnoDB Change Buffer (formerly called the InnoDB Insert Buffer) occupied about 10% of the buffer pool, and took an additional 2ks to process.

G.9.12 EXPERIMENT.9 innodb_log_buffer_size=80M, innodb_log_file_size=512M, ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4, fast index creation

Code: D1F+L512P1024H+Z+B4, fast index creation

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf differs from the baseline as follows:

```
innodb_log_buffer_size          =  80M # default   8M. (max disk write speed)
innodb_log_file_size            = 512M # default   5M. (1/2 buffer pool size)
innodb_flush_log_at_trx_commit  =    0 # default    1. (0=flush log buff 1/sec)
large_pages                     =    1 # default  OFF.
```

The image table is compressed and has 4K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4;
```

The new parameters are set to:

```
(defparameter *mirror-innodb-fast-index-creation*         t)
(defparameter *mirror-split-sql*                    "schunk")
```

Fast index creation is done by DROPping the secondary indices, INSERTing the data, then ADDing the secondary indices.

```
mysql> ALTER TABLE commonswiki.image
    -> DROP INDEX img_usertext_timestamp, DROP INDEX img_size,
    -> DROP INDEX img_timestamp, DROP INDEX img_sha1;
root-shell# wp-mirror --delete all
root-shell# wp-mirror --mirror
mysql> ALTER TABLE commonswiki.image
    -> ADD INDEX `img_usertext_timestamp` (`img_user_text`,`img_timestamp`),
    -> ADD INDEX `img_size` (`img_size`),
    -> ADD INDEX `img_timestamp` (`img_timestamp`),
    -> ADD INDEX `img_sha1` (`img_sha1`);
ERROR 1114 (HY000): The table 'image' is full
```

Unfortunately, this failed for the following reason:

> **Limitations of Fast Index Creation**
> During index creation, files are written to the temporary directory ($TMPDIR on Unix, %TEMP% on Windows, or the value of the tmpdir [566] configuration variable). Each temporary file is large enough to hold one column that makes up the new index, and each one is removed as soon as it is merged into the final index.
> MySQL 5.5 Reference Manual, 14.4.2 Fast Index Creation in the InnoDB Storage Engine

My /tmp resides on its own partition which is only 1G. During fast index creation, the /tmp directory fills up, and then MySQL throws the "The table 'image' is full" error.

Work-around: Use larger partition for TMP.

```
root-shell# emacs /etc/mysql/conf.d/wp-mirror.cnf
# fast index creation requires large TMP directory
tmpdir                          = /var/tmp # default /tmp
root-shell# service mysql stop
root-shell# service mysql start
mysql> SHOW VARIABLES LIKE 'tmpdir';
+---------------+----------+
| Variable_name | Value    |
+---------------+----------+
| tmpdir        | /var/tmp |
+---------------+----------+
mysql> ALTER TABLE commonswiki.image
    -> ADD INDEX `img_usertext_timestamp` (`img_user_text`,`img_timestamp`),
    -> ADD INDEX `img_size` (`img_size`),
    -> ADD INDEX `img_timestamp` (`img_timestamp`),
    -> ADD INDEX `img_sha1` (`img_sha1`);
Query OK, 0 rows affected (2 hours 32 min 24.15 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Measurement: commonswiki-20130216-image.sql.gz is 8.3G, commonswiki-20130216-image.sql is 34G, /var/lib/mysql/commonswiki/image.ibd is 12G after INSERT and 14G after ADD INDEX.

```
mysql> SELECT data_length, index_length FROM information_schema.tables
    -> WHERE table_schema='commonswiki' AND table_name='image';
+-------------+--------------+
| data_length | index_length |
+-------------+--------------+
| 11947212800 |   2045509632 |
+-------------+--------------+
```

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       0.9 |         1.0 |
|       1.7 |         2.0 |
|       2.4 |         3.0 |
|       3.0 |         4.0 |
|       3.8 |         5.0 |
|       4.6 |         6.0 |
|       5.4 |         7.0 |
|       6.2 |         8.0 |
|       7.0 |         9.0 |
|       7.8 |        10.0 |
|       8.6 |        11.0 |
|       9.4 |        12.0 |
|      10.1 |        13.0 |
|      10.9 |        14.0 |
|      11.6 |        15.0 |
|      12.4 |        16.0 |
|      12.9 |        17.0 |
|      13.6 |        18.0 |
|      14.4 |        19.0 |
|      15.2 |        20.0 |
|      16.0 |        21.0 |
|      16.7 |        22.0 |
|      17.6 |        23.0 |
|      18.4 |        24.0 |
|      19.2 |        25.0 |
|      19.9 |        26.0 |
|      20.6 |        27.0 |
|      21.3 |        28.0 |
|      22.1 |        29.0 |
|      22.9 |        30.0 |
|      33.5 |        30.0 |   <-- +10.6 [ks] fsm-table-add-index
+-----------+-------------+
```

G.9.13 EXPERIMENT.10 innodb_log_buffer_size=80M, innodb_log_file_size=512M, ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4, fast index, LOAD DATA INFILE

Code: D1F+L512P1024H+Z+B4, fast index, LOAD DATA INFILE

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf differs from the baseline as follows:

```
innodb_log_buffer_size          =  80M # default   8M. (max disk write speed)
innodb_log_file_size            = 512M # default   5M. (1/2 buffer pool size)
innodb_flush_log_at_trx_commit  =    0 # default    1. (0=flush log buff 1/sec)
large_pages                     =    1 # default  OFF.
```

The image table is compressed and has 4K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4;
```

The new parameters are set to:

```
(defparameter *mirror-innodb-fast-index-creation*         t)
(defparameter *mirror-split-sql*                    "dchunk")
```

The fast index creation and LOAD DATA INFILE method involves: 1) DROPping all secondary indices for commonswiki.image, 2) preparing a file containing data fields only, 3) LOADing that file (specifying the absolute path), and finally 4) ADDing the secondary indices:

```
mysql> TRUNCATE TABLE commonswiki.image;
mysql> ALTER TABLE commonswiki.image
    -> DROP INDEX img_usertext_timestamp, DROP INDEX img_size,
    -> DROP INDEX img_timestamp, DROP INDEX img_sha1;
mysql> LOAD DATA INFILE
    -> /var/lib/mediawiki/images/wp-mirror/commonswiki-20130124-image-p000000000-c000100000.dat'
    -> IGNORE INTO TABLE `commonswiki`.`image`
    -> FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY "'"
    -> LINES TERMINATED BY '),(';
mysql> ALTER TABLE commonswiki.image
    -> ADD INDEX `img_usertext_timestamp` (`img_user_text`,`img_timestamp`),
    -> ADD INDEX `img_size` (`img_size`),
    -> ADD INDEX `img_timestamp` (`img_timestamp`),
    -> ADD INDEX `img_sha1` (`img_sha1`);
Query OK, 0 rows affected (2 hours 12 min 12.54 sec)
```

Once the data is LOADed, it is useful to measure the time it takes to reload the data using either the IGNORE or the REPLACE options:

```
mysql> LOAD DATA INFILE
    -> '/var/lib/mediawiki/images/wp-mirror/commonswiki-20130124-image-p000000000-c000100000.dat'
    -> IGNORE INTO TABLE `commonswiki`.`image`
    -> FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY "'"
    -> LINES TERMINATED BY '),(';

mysql> LOAD DATA INFILE
    -> '/var/lib/mediawiki/images/wp-mirror/commonswiki-20130124-image-p000000000-c000100000.dat'
    -> REPLACE INTO TABLE `commonswiki`.`image`
    -> FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY "'"
    -> LINES TERMINATED BY '),(';
```

Measurement: commonswiki-20130216-image.sql.gz is 8.3G, commonswiki-20130216-image.sql is 34G, /var/lib/mysql/commonswiki/image.ibd is 4.6G after LOAD DATA INFILE and 6.1G after ADD INDEX.

```
mysql> SELECT data_length, index_length FROM information_schema.tables
    -> WHERE table_schema='commonswiki' AND table_name='image';
+-------------+--------------+
| data_length | index_length |
+-------------+--------------+
|  4714135552 |   1664352256 |
+-------------+--------------+
```

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       0.9 |         1.0 |
|       1.8 |         2.0 |
|       2.6 |         3.0 |
|       3.4 |         4.0 |
|       4.2 |         5.0 |
|       5.0 |         6.0 |
|       5.8 |         7.0 |
|       6.6 |         8.0 |
|       7.4 |         9.0 |
|       8.2 |        10.0 |
|       9.1 |        11.0 |
|       9.9 |        12.0 |
|      10.8 |        13.0 |
|      11.5 |        14.0 |
|      12.3 |        15.0 |
|      13.2 |        16.0 |
|      13.8 |        17.0 |
|      14.4 |        18.0 |
|      15.1 |        19.0 |
|      16.0 |        20.0 |
|      16.7 |        21.0 |
|      17.5 |        22.0 |
|      18.2 |        23.0 |
|      19.0 |        24.0 |
|      19.9 |        25.0 |
|      20.7 |        26.0 |
|      21.4 |        27.0 |
|      22.3 |        28.0 |
|      23.0 |        29.0 |
|      23.9 |        30.0 |
|      32.0 |        30.0 |   <-- +8.1 [ks] fsm-table-add-index
+-----------+-------------+
```

G.9.14 EXPERIMENT.11a innodb_log_buffer_size=80M, innodb_log_file_size=512M, ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4, fast index creation, schunk-page-count=100

Code: D1F+L512P1024H+Z+B4, fast index creation, p100

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf differs from the baseline as follows:

```
innodb_log_buffer_size          =  80M # default   8M. (max disk write speed)
innodb_log_file_size            = 512M # default   5M. (1/2 buffer pool size)
innodb_flush_log_at_trx_commit  =    0 # default    1. (0=flush log buff 1/sec)
large_pages                     =    1 # default  OFF.
```

The image table is compressed and has 4K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4;
```

The new parameters are set to:

```
(defparameter *mirror-innodb-fast-index-creation*         t)
(defparameter *mirror-schunk-page-count*                100)
(defparameter *mirror-split-sql*                      "schunk")
```

Measurement: commonswiki-20130216-image.sql.gz is 8.3G, commonswiki-20130216-image.sql is 34G, /var/lib/mysql/commonswiki/image.ibd is 12G after INSERT and 14G after ADD INDEX.

```
mysql> SELECT data_length, index_length FROM information_schema.tables
    -> WHERE table_schema='commonswiki' AND table_name='image';
+-------------+--------------+
| data_length | index_length |
+-------------+--------------+
| 11925192704 |   2037907456 |
+-------------+--------------+
```

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       0.5 |         1.0 |
|       1.0 |         2.0 |
|       1.5 |         3.0 |
|       1.9 |         4.0 |
|       2.4 |         5.0 |
|       2.8 |         6.0 |
|       3.4 |         7.0 |
|       3.9 |         8.0 |
|       4.5 |         9.0 |
|       4.9 |        10.0 |
|       5.4 |        11.0 |
|       5.8 |        12.0 |
|       6.3 |        13.0 |
|       6.8 |        14.0 |
|       7.3 |        15.0 |
|       7.7 |        16.0 |
|       8.0 |        17.0 |
|       8.4 |        18.0 |
|       8.9 |        19.0 |
|       9.4 |        20.0 |
|       9.8 |        21.0 |
|      10.2 |        22.0 |
|      10.8 |        23.0 |
|      11.3 |        24.0 |
|      11.8 |        25.0 |
|      12.2 |        26.0 |
|      12.6 |        27.0 |
|      13.1 |        28.0 |
|      13.6 |        29.0 |
|      14.1 |        30.0 |
|      24.1 |        30.0 |   <-- +10.0 [ks] fsm-table-add-index
+-----------+-------------+
```

G.9.15 EXPERIMENT.11b innodb_log_buffer_size=80M, innodb_log_file_size=512M, ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4, fast index creation, schunk-page-count=100, reload into populated commonswiki.image

This experiment is a repeat of 11a, with one difference: commonswiki.image is still populated with the results from 11a, and we use neither DROP INDEX at the start of the run nor ADD INDEX at the end.

Measurement: /var/lib/mysql/commonswiki/image.ibd is 15G.

```
mysql> SELECT data_length, index_length FROM information_schema.tables
    -> WHERE table_schema='commonswiki' AND table_name='image';
+-------------+--------------+
| data_length | index_length |
+-------------+--------------+
| 11943542784 |   2712387584 |
+-------------+--------------+
```

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       0.7 |         1.0 |
|       1.2 |         2.0 |
|       1.7 |         3.0 |
|       2.1 |         4.0 |
|       2.7 |         5.0 |
|       3.1 |         6.0 |
|       3.6 |         7.0 |
|       4.1 |         8.0 |
|       4.5 |         9.0 |
|       5.0 |        10.0 |
|       5.5 |        11.0 |
|       6.0 |        12.0 |
|       6.6 |        13.0 |
|       7.0 |        14.0 |
|       7.4 |        15.0 |
|       7.9 |        16.0 |
|       8.0 |        17.0 |
|       8.3 |        18.0 |
|       8.7 |        19.0 |
|       9.2 |        20.0 |
|       9.6 |        21.0 |
|      10.0 |        22.0 |
|      10.4 |        23.0 |
|      10.8 |        24.0 |
|      11.3 |        25.0 |
|      11.8 |        26.0 |
|      12.1 |        27.0 |
|      12.6 |        28.0 |
|      13.1 |        29.0 |
|      13.7 |        30.0 |
|      15.9 |        30.0 |   <-- +2.2 [ks] fragmentation of secondary indices
+-----------+-------------+
```

G.9.16 EXPERIMENT.12a innodb_log_buffer_size=80M, innodb_log_file_size=512M, ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4, fast index, LOAD DATA INFILE, dchunk-page-count=100

Code: D1F+L512P1024H+Z+B4, fast index, LOAD DATA INFILE, p100

Configuration files: /etc/mysql/conf.d/wp-mirror.cnf differs from the baseline as follows:

```
innodb_log_buffer_size           =  80M # default   8M. (max disk write speed)
innodb_log_file_size             = 512M # default   5M. (1/2 buffer pool size)
innodb_flush_log_at_trx_commit   =    0 # default    1. (0=flush log buff 1/sec)
large_pages                      =    1 # default  OFF.
```

The image table is compressed and has 4K page size:

```
mysql> ALTER TABLE commonswiki.image ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4;
```

The new parameters are set to:

```
(defparameter *mirror-dchunk-page-count*                100)
(defparameter *mirror-innodb-fast-index-creation*         t)
(defparameter *mirror-split-sql*                    "dchunk")
```

Measurement: commonswiki-20130216-image.sql.gz is 8.3G, commonswiki-20130216-image.sql is 34G, /var/lib/mysql/commonswiki/image.ibd is 4.6G after LOAD DATA INFILE and 6.1G after ADD INDEX.

```
mysql> SELECT data_length, index_length FROM information_schema.tables
    -> WHERE table_schema='commonswiki' AND table_name='image';
+-------------+--------------+
| data_length | index_length |
+-------------+--------------+
|  4667994112 |   1644429312 |
+-------------+--------------+
```

```
+-----------+-------------+
| time [ks] | inserts [k] |
+-----------+-------------+
|       0.0 |         0.0 |
|       0.4 |         1.0 |
|       0.8 |         2.0 |
|       1.2 |         3.0 |
|       1.6 |         4.0 |
|       2.0 |         5.0 |
|       2.3 |         6.0 |
|       2.7 |         7.0 |
|       3.0 |         8.0 |
|       3.4 |         9.0 |
|       3.7 |        10.0 |
|       4.1 |        11.0 |
|       4.5 |        12.0 |
|       5.0 |        13.0 |
|       5.2 |        14.0 |
|       5.6 |        15.0 |
|       5.9 |        16.0 |
|       6.0 |        17.0 |
|       6.2 |        18.0 |
|       6.5 |        19.0 |
|       6.9 |        20.0 |
|       7.2 |        21.0 |
|       7.6 |        22.0 |
|       7.8 |        23.0 |
|       8.1 |        24.0 |
|       8.5 |        25.0 |
|       8.8 |        26.0 |
|       9.1 |        27.0 |
ERROR 2002 (HY000): Can't connect to local MySQL server through socket
'/var/run/mysqld/mysqld.sock' (2)
                              <-- this loss of connectivity is reproducable
|      10.6 |        30.0 |   <-- +1.5 [ks] rerun wp-mirror --mirror
|      19.3 |        30.0 |   <-- +8.7 [ks] fsm-table-add-index
+-----------+-------------+
```

G.9.17 EXPERIMENT.12b innodb_log_buffer_size=80M, innodb_log_file_size=512M, ROW_FORMAT=COMPRESSED, KEY_BLOCK_SIZE=4, fast index creation, dchunk-page-count=100, reload into populated commonswiki.image

This experiment is a repeat of 12a, with a difference: commonswiki.image is still populated with the results from 12a, and we use neither DROP INDEX at the start of the run nor ADD INDEX at the end.

Measurement: /var/lib/mysql/commonswiki/image.ibd is 6.4G.

```
mysql> SELECT data_length, index_length FROM information_schema.tables
    -> WHERE table_schema='commonswiki' AND table_name='image';
+-------------+--------------+
| data_length | index_length |
+-------------+--------------+
|  4668776448 |   1953759232 |
+-------------+--------------+
```

time [ks]	inserts [k]
0.0	0.0
0.7	1.0
1.5	2.0
2.2	3.0
2.9	4.0
4.1	5.0
5.0	6.0
5.8	7.0
6.6	8.0
7.4	9.0
8.2	10.0
9.1	11.0
10.1	12.0
11.1	13.0
11.6	14.0
12.4	15.0
13.2	16.0
13.4	17.0
13.8	18.0
14.5	19.0
15.4	20.0
16.1	21.0
16.9	22.0
17.3	23.0
18.0	24.0
18.9	25.0
19.7	26.0
20.3	27.0
21.0	28.0
21.6	29.0
22.6	30.0

We note that: 1) experiment 12b took longer than 12a, 2) /var/lib/mysql/commonswiki/image.ibd grew from 6.1G to 6.4G; all of the size increase was due to fragmentation of the pages that store the secondary indices.

G.9.18 Conclusions

We achieved a nine-fold increase in speed for loading the commonswiki.image database table. This was done by:

- increasing log buffer size from 8M to 80M to match best case disk performance;
- increasing log file size from 10M to 512M to match innodb_buffer_pool_size;
- using InnoDB Data Compression with page size 4K (which reduces INSERT speed a bit, but saves disk space);
- using fast index creation so that INSERT scales linearly rather than as a logarithm;
- using the LOAD DATA INFILE command to INSERT data at a lower level in the InnoDB storage engine; and
- using larger dchunks and schunks to reduce FSM overhead.

The results are charted in Figure G.1, InnoDB Experiments—Importing commonswiki.image Table.

The choice of using LOAD DATA INFILE versus using REPLACE INTO is interesting. When we populate a table for the first time, LOAD DATA INFILE is faster (compare experiment 12 with 11). However, if we update an already populated table, REPLACE INTO is faster (compare experiment 11a with 12a). The reason appears to be that LOAD DATA INFILE produces a *ibd file consisting of tightly packed data pages and index pages; and therefore any later update must entail page splitting, which in turn means writing two pages to disk for each page read. Since WP-MIRROR installs once and then updates monthly, using REPLACE INTO is the better choice.

Recommendation: Use REPLACE INTO rather than LOAD DATA INFILE.

G.10 Experiments with mwxml2sql

On 2013-Feb-21, Ariel T. Glenn of the WMF announced the development of new import tools:

```
Date: Thu, 21 Feb 2013 19:05:17 +0200
From: Ariel T. Glenn <ariel@wikimedia.org>
To: xmldatadumps-l@lists.wikimedia.org
Subject: [Xmldatadumps-l] toos for import (setting up a local mirror)

So partly due to recent work by folks like Kent on creating local WP
mirrors using the import process, and partly from helping walk someone
through the process for the zillionth time, I have come to the
realization that This Process Sucks (TM).  I am not taking on the whole
stack, but I am trying to make a dent in at least part of it.  To that
end:

1) mwdumper available from download.wikimedia.org is now the current
version and should run without a bunch of fancy tricks.  Thanks to Chad
for fixing up the jenkins build. I tried it on a recent en wikipedia
current pages dump and it seemed to work. though I did not test
importing the output.

2) I have a couple of tools for *nix users importing into a MySQL
database.

* Somewhat equivalent to mwdumper is 'mwxml2sql', name chosen before I
saw that there was a long abandonded xml2sql tool available in the wild.
Input: stubs and page content xml files, output: sql files for each of
page, revision, text table, reading 0.4 xsd through 0.7 and writing Mw
1.5 through 1.20 output, as specified by the user.  Many specific
combinations are untested (e.g. I spent most work on 0.7 xds to MW
1.20).

* Converting an sql dump file to a tab delimited format suitable for
'LOAD DATA INFILE' is now possible via 'sql2txt' (also *nix platforms).
```

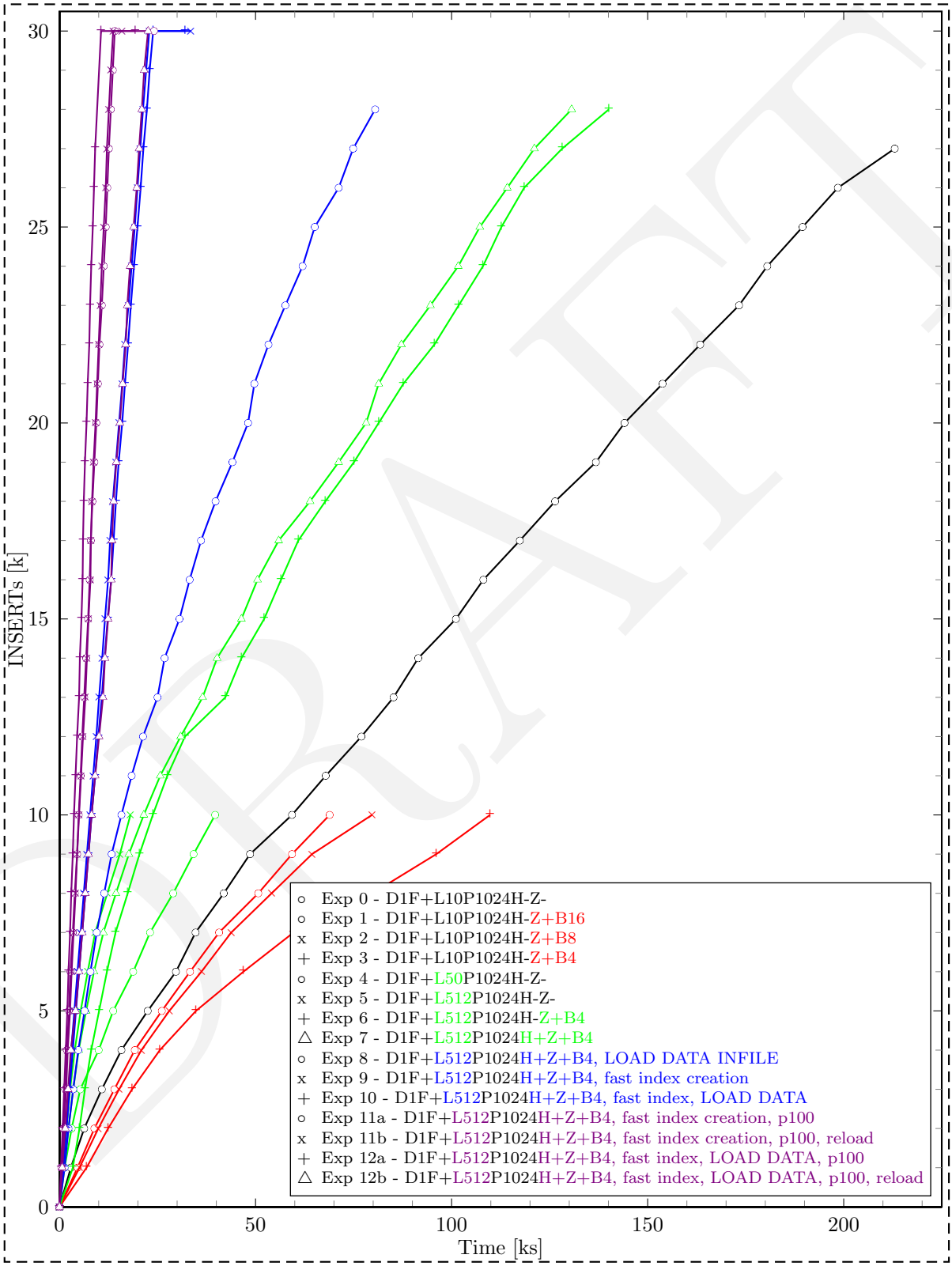
I would like to test wmxml2sql-0.0.2, the code for which can be found by chasing a link found at https://meta.wikimedia.org/wiki/Data_dumps/Tools_for_importing. An usage example can be found at https://meta.wikimedia.org/wiki/Data_dumps/Import_examples.

G.10.1 Testing mwxml2sql

A design choice had to be made:

1. split the XML dump file into xchunks, and then use mwxml2sql to convert xchunks into schunks; or

Figure G.1: InnoDB Experiments—Importing commonswiki.image Table



2. use mwxml2sql to convert the XML dump file into large SQL files, and then split the SQL files into schunks.

I decided that the first choice was more robust, and indeed trial runs with xchunks worked well, as failure was confined to very few xchunks.

mwxml2sql runs fast. It converts an xchunk into three schunks, one each for the page, revision, and text tables. Some schunks fail to import, throwing SQL syntax errors like:

```
[....]fsm-file-load-insert simplewiki-20130525-revision-p000035000-c000001000.sql
ERROR 1064 (42000) at line 10: You have an error in your SQL syntax;
check the manual that corresponds to your MySQL server version for the
right syntax to use near ' 'Fix typos, formatting, brackets and links,
typos fixed: etc\n
...
[....]fsm-file-load-insert simplewiki-20130525-revision-p000151000-c000001000.sql
ERROR 1064 (42000) at line 10: You have an error in your SQL syntax;
check the manual that corresponds to your MySQL server version for the
right syntax to use near ' 'Fix typos, formatting, brackets and links,
typos fixed: etc\n
...
[....]fsm-file-load-insert simplewiki-20130525-text-p000035000-c000001000.sql
ERROR 1064 (42000) at line 10: You have an error in your SQL syntax;
check the manual that corresponds to your MySQL server version for the
right syntax to use near ' 'This category is for Kings and Queens in the
various states within Germa
...
[....]fsm-file-load-insert simplewiki-20130525-text-p000151000-c000001000.sql
ERROR 1064 (42000) at line 10: You have an error in your SQL syntax;
check the manual that corresponds to your MySQL server version for the
right syntax to use near ' '{{documentation subpage}}\nThis template
takes 2 arguments:\n* <nowiki>{
```

In terms of performance, the real issue is importation of the schunks into the database, and this was studied in §G.9, Experiments with commonswiki.image. Coding and testing for the Finite State Machine took a couple of months.

The build for enwiki and enwiktionary without images completed in 800ks (nine days). This is 80% less time than WP-MIRROR 0.5. While most of the performance gain occurred in other areas of the code, it is clear that using mwxml2sql is a win.

G.10.2 Packaging mwxml2sql for Debian

For the WP-MIRROR DEB package to make use of mwxml2sql it was necessary to package mwxml2sql, and then edit WP-MIRROR's debian/control file to include it as a dependency.

However, to make a DEB package for mwxml2sql, the code and its Makefile had to be patched. For one thing, Debian requires man pages, and mwxml2sql had none. So I decided to modify the code so that man pages could be generated using help2man. For another thing, the Makefile had no build target, no deinstall target, and other targets needed work as well.

I produced a DEB package called mediawiki-mwxml2sql. The name was chosen to be consistent with other mediawiki packages that are part of the Debian distribution. mediawiki-mwxml2sql builds with debuild and pbuilder.

The next question was to determine if the patches would be welcome by the upstream author, Ariel T. Glenn <ariel@wikimedia.org>.

G.10.3 Submitting Patches Upstream to WMF

The people at the WMF do welcome patches. However, submitting code requires some initial setup.

- First, on your local platform, you must:
  - install git and git-review, and
  - generate an SSH key pair.
- Second, on the WMF side, you must
  - request a developer account from http://wikitech.wikimedia.org/ (Log in),
  - upload your SSH public key to http://wikitech.wikimedia.org/ (Log in), and
  - upload your SSH public key to http://gerrit.wikimedia.org/ (Sign in).
- Third, you must familiarize yourself with:
  - the version control system git, and
  - the software review system gerrit.

All the above is described by reading https://www.mediawiki.org/wiki/Gerrit, https://www.mediawiki.org/wiki/Gerrit/Tutorial, and related pages.

Using git may require some learning. Some useful links are:

- tutorials http://sixrevisions.com/resources/git-tutorials-beginners/,
- concepts http://www.sbf5.com/~cduan/technical/git/, and
- storage http://eagain.net/articles/git-for-computer-scientists/index.html.

Still, it took a day of trial and error to get my patches submitted (see https://gerrit.wikimedia.org/r/64343/). The git commands I used are given in the following Makefile.

```
wikimedia@darkstar-7:~$ cat Makefile
#-------------------------------------------------------------------------+
# Makefile for submitting patches to the Wikimedia Foundation             |
# Copyright (C) 2013 Dr. Kent L. Miller.  All rights reserved.            |
#                                                                         |
# This program is free software: you can redistribute it and/or modify    |
# it under the terms of the GNU General Public License as published by    |
# the Free Software Foundation, either version 3 of the License, or (at   |
# your option) any later version.                                         |
#                                                                         |
# This program is distributed in the hope that it will be useful, but     |
# WITHOUT ANY WARRANTY; without even the implied warranty of              |
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU        |
# General Public License for more details.                                |
#                                                                         |
# You should have received a copy of the GNU General Public License       |
# along with this program.  If not, see <http://www.gnu.org/licenses/>")  |
#-------------------------------------------------------------------------+
GERRIT   = wpmirrordev@gerrit.wikimedia.org
PORT     = 29418
WMFLABS  = wpmirrordev@bastion.wmflabs.org

all: clone hooks pull checkout branch edit diff add commit rebase review

clone:
        # create directory 'dumps' and initialize a repository in it
        # copy all commit objects and head references (from remote to local)
```

```
        # add 'remote repository reference' named 'origin' (saves typing)
        # add 'remote heads' named 'origin/[head-name]'
        # add 'HEAD' to track 'origin/master'
        git clone ssh://$(GERRIT):$(PORT)/operations/dumps

hooks:
        # get 'pre-commit-hook' to add 'change id' to commit summary
        scp -p -P $(PORT) $(GERRIT):hooks/commit-msg ~/dumps/.git/hooks/.
        cd dumps; git review -s

pull:
        # list 'remote heads'
        cd dumps; git branch -r
        # setup tracking branch 'ariel'
        cd dumps; git branch --track ariel origin/ariel
        # add new commit objects (if any)
        # update 'remote heads'
        cd dumps; git fetch origin
        # update 'local heads' ('master' and 'ariel') to 'remote-heads'
        # merge 'origin/HEAD' into 'HEAD'
        cd dumps; git pull origin

checkout:
        # point 'HEAD' to 'ariel's commit object
        cd dumps; git checkout ariel
        cd dumps; git status

branch:
        # create head 'wpmirrordev'
        # point 'wpmirrordev' to 'ariel's commit object
        cd dumps; git branch wpmirrordev ariel
        # point 'HEAD' to 'wpmirrordev's commit object
        cd dumps; git checkout wpmirrordev
        cd dumps; git status

edit:
        # apply patched files
        cp temp/* dumps/xmlfileutils/.

diff:
        # diff files (but not added files) against 'HEAD'
        cd dumps; git diff
        # list changed files against 'HEAD'
        cd dumps; git status

add:
        # stage the files to be committed
        cd dumps/xmlfileutils; git add mwxml2sql.c sql2txt.c sqlfilter.c
        cd dumps/xmlfileutils; git add Makefile
        # diff added files against 'HEAD'
        cd dumps; git diff --cached
        # list changed files against 'HEAD'
        cd dumps; git status

commit:
```

```
        # create ‘commit object’
        # point ‘HEAD’ to the new ‘commit object’
        cd dumps; git commit -m "Fix for compatibility with help2man and Debian Policy"
        # list all commits from ‘HEAD’ back to initial commit
        cd dumps; git log

rebase:
        # add new commit objects (if any)
        # update ‘remote head’ ‘origin/ariel’
        # merge ‘origin/ariel’ into ‘ariel’
        # point ‘ariel’ to ‘origin/ariel’s commit object
        cd dumps; git pull origin ariel
        # rebase ‘wpmirrordev’ branch on updated ‘ariel’ head
        cd dumps; git rebase ariel

review:
        # push changes to Gerrit
        cd dumps; git review -R ariel

#-------------------------------------------------------------------------------+

shell:
        ssh -A $(WMFLABS)

purge:
        rm -r dumps

clean:
        rm -f *~
```

GNU Free Documentation License

Version 1.3, 3 November 2008

<http://fsf.org/>

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to

be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts,

you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else an unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.