
WP-MIRROR 0.4 Reference Manual

Dr. Kent L. Miller

November 12, 2012

To Tylery

WP-MIRROR 0.4 Reference Manual

Legal Notices

Copyright (C) 2012 Dr. Kent L. Miller. All rights reserved.

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](#).

THIS PUBLICATION AND THE INFORMATION HEREIN ARE FURNISHED AS IS, ARE FURNISHED FOR INFORMATIONAL USE ONLY, ARE SUBJECT TO CHANGE WITHOUT NOTICE, AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY THE AUTHOR. THE AUTHOR ASSUMES NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR INACCURACIES THAT MAY APPEAR IN THE INFORMATIONAL CONTENT CONTAINED IN THIS MANUAL, MAKES NO WARRANTY OF ANY KIND (EXPRESS, IMPLIED, OR STATUTORY) WITH RESPECT TO THIS PUBLICATION, AND EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES OF MERCHANTABILITY, FITNESS FOR PARTICULAR PURPOSES, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

[Debian](#) is a registered United States trademark of Software in the Public Interest, Inc., managed by the Debian project. [Linux](#) is a trademark of Linus Torvalds. [InnoDB](#) and [MySQL](#) are trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Abstract

The [WikiMedia Foundation](#) offers wikipeidias in nearly [300 languages](#).

WP-MIRROR is a free utility for mirroring any desired set of these wikipeidias. That is, it builds a wikipedia 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 (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 40G of disk space, be served locally by a virtual host <http://simple.mediawiki.site/>, and update automatically every week. The default should be suitable for anyone who learned English as a second language (ESL).

The top ten wikipeidias are: [en](#), [de](#), [fr](#), [nl](#), [it](#), [es](#), [pl](#), [ru](#), [ja](#), and [pt](#). Because WP-MIRROR uses original size image files, these wikipeidias 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 5Ms (two months), occupy 2T of disk space, be served locally by a virtual host <http://en.mediawiki.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 November 12, 2012.

Features

WP-MIRROR has been designed for robustness. WP-MIRROR asserts hardware and software prerequisites, skips over unparsable articles and bad file names, validates image files (many are corrupt), waits for Internet access when needed, uses check-pointing to resume after interruption, and offers concurrency to accelerate mirroring of the largest wikipedias.

WP-MIRROR warns: if [MySQL](#) is insecure (e.g. has a root account with no password), and 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 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`. Wikipedias can be added to the mirror, or dropped from it. If a user makes a hash of the configuration files 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. A 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 two of the smaller wikipedias (e.g. the [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>](mailto:wp-mirror-list@nongnu.org).

Contents

WP-MIRROR 0.4 Reference Manual	ii
Contents	iv
List of Figures	xii
List of Tables	xiii
1 General Information	1
1.1 About This Manual	1
1.2 Typographical and Syntax Conventions	2
1.3 Overview of WP-MIRROR	2
1.3.1 What is WP-MIRROR?	2
1.3.2 History of WP-MIRROR	3
1.3.3 The Main Features of WP-MIRROR	4
1.4 WP-MIRROR Development History	5
1.5 What Is New in WP-MIRROR	5
1.5.1 What Is New in WP-MIRROR 0.4	6
1.5.2 What Is New in WP-MIRROR 0.3	7
1.5.3 What Is New in WP-MIRROR 0.2	7
1.5.4 What Is New in WP-MIRROR 0.1	8
1.6 WP-MIRROR Information Sources	9
1.6.1 Project Home Page for WP-MIRROR	9
1.6.2 Downloading WP-MIRROR	9
1.6.3 Documentation for WP-MIRROR	9
1.6.4 Mailing Lists for WP-MIRROR	9
1.7 How to Report Bugs or Problems	9
1.8 Credits	10
1.8.1 Contributors to WP-MIRROR	10
1.8.2 Documenters and Translators	10
2 Installing and Upgrading WP-MIRROR	11
2.1 General Installation Guidance	11
2.1.1 Operating Systems Supported by WP-MIRROR	11
2.1.2 How to Get WP-MIRROR	11
2.1.3 Choosing Which WP-MIRROR Distribution to Install	11
2.1.3.1 Choosing Which Version of WP-MIRROR to Install	11
2.1.3.2 Choosing a Distribution Format	11
2.1.3.3 Choosing When to Upgrade	12
2.1.4 Verifying Package Integrity Using Checksums or GnuPG	12
2.1.4.1 How to Get the Author's Public Build Key	12
2.1.4.1.1 Download the Public Key from Public Keyserver	13
2.1.4.1.2 Cut and Paste the Public Key from Public Keyserver	13
2.1.4.2 Checksums for a DEB Package	14

2.1.4.3	Checksums for a <code>RPM</code> Package	14
2.1.4.4	Signature Checking for a Tarball	15
2.2	Installing WP-MIRROR on Linux	15
2.2.1	Installing from a <code>DEB</code> Package	15
2.2.2	Installing from a <code>RPM</code> Package	16
2.3	Installing WP-MIRROR from Source	16
2.3.1	Installing Dependencies	16
2.3.1.1	Installing Build Dependencies	16
2.3.1.2	Verifying <code>clisp</code> Version	16
2.3.1.3	Configuring <code>common-lisp-controller</code>	17
2.3.1.4	Configuring <code>cl-asdf</code>	17
2.3.1.5	Installing Binary Dependencies	18
2.3.2	Installing WP-MIRROR from a Standard Source Distribution	18
2.3.3	WP-MIRROR Build Options	18
2.4	Postinstallation Configuration and Testing	19
2.4.1	Default Configuration	19
2.4.2	Working with <code>X</code>	20
2.4.3	Configuration of a Wikipedia Farm	20
3	System Planning and Configuration	22
3.1	General Information	22
3.2	Laptop Planning and Configuration	23
3.2.1	Plan Your Disk Space	23
3.2.1.1	Disable HDD Write Caching	23
3.2.1.2	Put <code>images</code> Directory on Partition with Adequate Free Space	24
3.2.1.3	Setup Thermal Management	25
3.2.1.4	Setup <code>smart</code> Disk Monitoring	25
3.2.2	Plan Your Database Management System	26
3.2.2.1	Secure the Database Management System	26
3.2.2.2	Load the DBMS <code>time_zone</code> Tables	27
3.2.2.3	Configure the <code>InnoDB</code> Storage Engine	28
3.2.3	Plan Your DRAM	29
3.2.4	Plan Your Internet Access	29
3.2.4.1	Configure <code>cURL</code>	29
3.2.5	Plan Your <code>MediaWiki</code>	30
3.2.5.1	Configure <code>MediaWiki</code>	30
3.2.5.2	Enable <code>MediaWiki</code> Extensions	31
3.2.5.3	<code>MediaWiki</code> Redux	32
3.2.6	Plan Your Image Processing	32
3.2.6.1	Replace <code>ImageMagick</code> with <code>GraphicsMagick</code>	32
3.2.6.2	Replace <code>SVG</code> Converter	33
3.2.7	Plan Your Virtual Host	34
3.2.7.1	Enable Virtual Host	34
3.2.8	Plan Your Caching Web Proxy	35
3.2.8.1	Configure <code>bash</code> for Use with Caching Web Proxy	35
3.2.8.2	Configure <code>cURL</code> for Use with Caching Web Proxy	35
3.2.8.3	Configure <code>wget</code> for Use with Caching Web Proxy	36
3.2.8.4	Configure Browser for Use with Caching Web Proxy	36
3.3	Desktop Planning and Configuration	36
3.3.1	Procure Hardware	37
3.3.2	Plan Your Disk Space	37
3.3.2.1	Disable HDD Write Caching	37
3.3.2.2	Setup Thermal Management	37
3.3.2.3	Setup <code>smart</code> Disk Monitoring	38
3.3.2.4	Allocate Disk Space for Articles	38

3.3.2.5	Allocate Disk Space for Image Files	38
3.3.2.6	Design Storage for Security and Robustness	38
3.3.2.7	Build Your Storage Array	39
3.3.2.7.1	Whole Disks	39
3.3.2.7.2	RAID	40
3.3.2.7.3	LUKS	40
3.3.2.7.4	LVM2	41
3.3.2.7.5	File System	41
3.3.2.7.6	Raw Partition	42
3.3.3	Plan Your Database Management System	42
3.3.3.1	Secure the Database Management System	42
3.3.3.2	Load the DBMS <code>time_zone</code> Tables	42
3.3.3.3	Customize the InnoDB Storage Engine	42
3.3.4	Plan Your DRAM	44
3.3.4.1	Hugepages	44
3.3.4.2	Permissions	44
3.3.4.3	Buffer Pool	44
3.3.4.4	Process Limits	45
3.3.5	Plan Your Internet Access	45
3.3.5.1	Configure <code>cURL</code>	45
3.3.6	Plan Your <code>MediaWiki</code>	45
3.3.6.1	Configure <code>MediaWiki</code>	45
3.3.6.2	Enable <code>MediaWiki</code> Extensions	45
3.3.6.3	<code>MediaWiki</code> Redux	45
3.3.7	Plan Your Image Processing	45
3.3.7.1	Replace <code>ImageMagick</code> with <code>GraphicsMagick</code>	45
3.3.7.2	Replace <code>SVG</code> Converter	45
3.3.8	Plan Your Virtual Host	45
3.3.8.1	Enable Virtual Host	45
3.3.9	Plan Your Caching Web Proxy	45
3.3.9.1	Download Large Dump Files	46
3.3.9.2	Download Millions of Image Files	46
3.3.9.3	Bypass Unreliable Caching Web Proxy	47
3.3.9.4	Configure <code>bash</code> for Use with Caching Web Proxy	47
3.3.9.5	Configure <code>cURL</code> for Use with Caching Web Proxy	47
3.3.9.6	Configure <code>wget</code> for Use with Caching Web Proxy	47
3.3.9.7	Configure Browser for Use with Proxy	47
4	WP-MIRROR Programs	48
4.1	WP-MIRROR in Mirror Mode	48
4.2	WP-MIRROR in Monitor Mode	49
4.2.1	WP-MIRROR in Monitor Mode with GUI Display	49
4.2.2	WP-MIRROR in Monitor Mode with Screen Display	49
4.2.3	WP-MIRROR in Monitor Mode with Text Display	49
5	How WP-MIRROR Works	52
5.1	How Mirror Mode Works	52
5.1.1	Start	52
5.1.1.1	<code>process-command-line-arguments-or-die</code>	52
5.1.2	Initializing	53
5.1.2.1	<code>clear-pidfile</code>	53
5.1.2.2	<code>set-pidfile</code>	53
5.1.2.3	set mode of operation to: <code>FIRST-MIRROR</code>	54
5.1.2.4	<code>assert-clisp-features-p</code>	54
5.1.2.5	<code>assert-utilities-p</code>	54

5.1.2.6	assert-images-directory-or-create-p	55
5.1.2.7	assert-images-bad-directory-or-create-p	55
5.1.2.8	assert-images-math-directory-or-create-p	55
5.1.2.9	assert-images-thumb-directory-or-create-p	55
5.1.2.10	assert-images-tmp-directory-or-create-p	55
5.1.2.11	assert-working-directory-or-create-p	56
5.1.2.12	assert-dbms-mysql-p	56
5.1.2.13	assert-dbms-mysql-config-debian-p	56
5.1.2.14	assert-dbms-credentials-debian-or-scrape-p	56
5.1.2.15	assert-dbms-connect-with-credentials-debian-p	56
5.1.2.16	assert-dbms-time-zone-or-load	57
5.1.2.17	assert-configuration-files-or-restore-default	57
5.1.2.18	process-configuration-files-or-die	57
5.1.2.19	put-parameters	58
5.1.3	Asserting Prerequisite Hardware	58
5.1.3.1	count-cpu	58
5.1.3.2	assert-disk-space-if-top-ten-p	58
5.1.3.3	assert-physical-memory-p	59
5.1.3.4	assert-partition-free-images	59
5.1.3.5	warn-if-disk-space-low-p	59
5.1.3.6	assert-internet-access-to-wikimedia-site-p	59
5.1.4	Asserting Prerequisite Software	60
5.1.4.1	assert-dbms-accounts-or-create-p	60
5.1.4.2	assert-dbms-credentials-or-scrape-p	60
5.1.4.3	assert-dbms-connect-with-credentials-wikiadmin-p	61
5.1.4.4	assert-dbms-connect-with-credentials-wikiuser-p	61
5.1.4.5	assert-dbms-grant-for-wikiadmin-p	61
5.1.4.6	assert-dbms-grant-for-wikiuser-p	61
5.1.4.7	warn-if-dbms-root-account-has-no-password	62
5.1.4.8	warn-if-dbms-has-anonymous-user-account	62
5.1.4.9	warn-if-dbms-has-root-accounts-accessible-from-outside-localhost	62
5.1.4.10	warn-if-dbms-has-test-database	62
5.1.4.11	assert-hdd-write-cache-disabled-p	62
5.1.4.12	assert-database-template-and-wikidb-p	63
5.1.4.13	assert-database-xxwiki-or-load	63
5.1.4.14	assert-database-wpmirror-or-create-p	63
5.1.4.15	assert-mediawiki-localsettings-p	64
5.1.4.16	assert-mediawiki-localsettings-account-p	64
5.1.4.17	assert-mediawiki-localsettings-wpmirror-p	64
5.1.4.18	assert-mediawiki-localsettings-image-p	64
5.1.4.19	assert-mediawiki-localsettings-tidy-p	65
5.1.4.20	assert-mediawiki-dbms-credentials-p	65
5.1.4.21	assert-concurrency-limit-xchunk-p	65
5.1.4.22	assert-virtual-host-p	65
5.1.4.23	assert-virtual-host-name-resolution-p	66
5.1.4.24	warn-if-detect-proxy	66
5.1.5	The Finite State Machine	66
5.1.6	The Finite State Machine—Step by Step	68
5.1.7	Check-pointing	70
5.1.8	Concurrency	70
5.1.9	The <code>fsm-*</code> Functions	72
5.1.9.1	<code>fsm-abort</code>	72
5.1.9.2	<code>fsm-boot</code>	72
5.1.9.3	<code>fsm-file-count</code>	72
5.1.9.4	<code>fsm-file-decompress</code>	73

5.1.9.5	fsm-file-download	73
5.1.9.6	fsm-file-import	74
5.1.9.7	fsm-file-md5sum	75
5.1.9.8	fsm-file-parse	75
5.1.9.9	fsm-file-remove	75
5.1.9.10	fsm-file-shell	76
5.1.9.11	fsm-file-split	77
5.1.9.12	fsm-file-validate	78
5.1.9.13	fsm-file-wikix	78
5.1.9.14	fsm-images-chown	79
5.1.9.15	fsm-images-count	79
5.1.9.16	fsm-images-rebuild	79
5.1.9.17	fsm-images-validate	80
5.1.9.18	fsm-no-op	80
5.1.10	Finalizing	80
5.1.10.1	clear-pidfile	81
5.1.10.2	log-stop	81
5.2	How Monitor Mode Works	81
5.2.1	Start	81
5.2.1.1	process-command-line-arguments-or-die	81
5.2.2	Initializing	81
5.2.2.1	assert-clisp-features-p	82
5.2.2.2	assert-utilities-p	82
5.2.2.3	assert-images-directory-or-create-p	82
5.2.2.4	assert-working-directory-or-create-p	82
5.2.2.5	assert-dbms-mysql-p	82
5.2.2.6	assert-dbms-mysql-config-debian-p	82
5.2.2.7	assert-dbms-credentials-debian-or-scrape-p	83
5.2.2.8	assert-dbms-connect-with-credentials-debian-p	83
5.2.2.9	assert-dbms-time-zone-or-load	83
5.2.2.10	assert-configuration-files-or-restore-default	83
5.2.2.11	process-configuration-files-or-die	84
5.2.3	Asserting Prerequisite Hardware	84
5.2.4	Asserting Prerequisite Software	85
5.2.4.1	assert-dbms-accounts-or-create-p	85
5.2.4.2	assert-dbms-credentials-or-scrape-p	85
5.2.4.3	assert-dbms-connect-with-credentials-wikiadmin-p	86
5.2.4.4	assert-dbms-connect-with-credentials-wikiuser-p	86
5.2.4.5	assert-dbms-grant-for-wikiadmin-p	86
5.2.4.6	assert-dbms-grant-for-wikiuser-p	86
5.2.4.7	assert-database-xxwiki-or-load	86
5.2.4.8	assert-database-wpmirror-or-create-p	87
5.2.4.9	assert-mediawiki-dbms-credentials-p	87
5.2.5	Compile Status Report	87
5.2.6	Display Progress Bars	87
5.3	How <code>--delete</code> Works	87
5.4	How <code>--drop</code> Works	88
5.5	How <code>--restore-default</code> Works	89
5.6	How Scheduled Jobs Work	90
5.6.1	cron	90
5.6.2	logrotate	90
A	Design Notes	91
A.1	Concurrency Limit	91
A.1.1	Why not Fork 20 <code>ichunk</code> Sub-processes?	91

A.1.2	Why not Fork 20 <code>xchunk</code> Sub-processes?	91
A.1.3	Why not Process <code>ichunks</code> in Parallel with <code>xchunks</code> ?	92
A.1.4	Design Decision	92
A.2	Mathematical Equations	92
A.2.1	Example: the Article on Algebra	92
A.2.2	<code>LaTeX</code>	96
A.2.3	<code>Math.php</code> and <code>texvc</code>	96
A.2.3.1	Update (2012)	96
A.2.4	Messages	97
A.3	Priority	97
B	Configuration File Parameter Reference	99
C	Error Messages (Work In Progress)	105
C.1	Error Codes	105
C.2	Error Codes (Obsolete)	106
C.3	Full Error Messages	107
C.4	Full Error Messages (Obsolete)	118
D	Change History	120
D.1	Changes in Release 0.x	120
D.1.1	Changes in WP-MIRROR 0.4 (2012-11-12)	120
D.1.1.1	Functionality Added or Changed	120
D.1.1.2	Bugs Fixed	121
D.1.2	Changes in WP-MIRROR 0.3 (2012-03-04)	122
D.1.2.1	Functionality Added or Changed	122
D.1.2.2	Bugs Fixed	122
D.1.3	Changes in WP-MIRROR 0.2 (2011-12-25)	122
D.1.3.1	Functionality Added or Changed	122
E	Experiments (Autumn 2010—Spring 2011)	124
E.1	Introduction	124
E.2	Hardware	126
E.2.1	PARTS	126
E.2.2	H/W Test	126
E.2.3	Partitions	126
E.2.4	RAID	126
E.2.5	(Optionally) Add Second Disk To Raid Array	128
E.2.6	LUKS	128
E.2.7	LVM2	130
E.2.8	ReiserFS	132
E.3	Configuring MySQL	133
E.4	Configuring <code>hugepages</code>	134
E.5	Configuring MediaWiki	138
E.6	Experiments with <code>MWdumper.jar</code> —Round 1	142
E.7	Experiments with InnoDB	143
E.7.1	Experimental Method	143
E.7.2	EXPERIMENT.0 Baseline configuration	144
E.7.3	EXPERIMENT.1 Put <code>ibdata1</code> on separate disk (but still on a file system)	145
E.7.4	EXPERIMENT.2 Store <code>ibdata1</code> on raw LVM2 partition (not a file system)	145
E.7.5	EXPERIMENT.3 Increase size of <code>buffer pool</code>	146
E.7.6	EXPERIMENT.4 Increase size of log file and log buffer	147
E.7.7	EXPERIMENT.5 Increase size of <code>buffer pool</code>	147
E.7.8	EXPERIMENT.6 Enable <code>hugepages</code>	148
E.7.9	EXPERIMENT.6b Repeat. Run to completion.	149

E.7.10 EXPERIMENT.6c Repeat. Do not disable keys.	149
E.7.11 EXPERIMENT.7 Increase <code>innodb_buffer_pool_size</code> with <code>filesystem,</code> <code>hugepage</code>	149
E.7.12 EXPERIMENT.8 Increase <code>innodb_buffer_pool_size</code> again with <code>filesys-</code> <code>tem, hugepage</code>	150
E.7.13 Conclusions	151
E.8 Experiments with <code>importDump.php</code> —Round 1	151
E.9 Experiments with <code>MWdumper.jar</code> —Round 2	154
E.10 Experiments with <code>importDump.php</code> —Round 2	157
E.11 Experiments with <code>wikix</code>	158
E.12 Experiments with Downloading Images	160
E.13 Experiments with <code>rebuildImages.php</code> —Round 1	161
E.14 Experiments with Corrupt Images	162
E.15 Experiments with the <code>objectcache</code>	163
E.16 Experiments with <code>rebuildImages.php</code> —Round 2	163
E.17 Experiments with Resource Contention	165
E.18 Upgrade from Debian Lenny to Squeeze	166
E.18.1 Fiasco with <code>pagelinks</code>	167
E.18.2 Post-Mortem Analysis and Lessons Learned	169
E.18.3 Starting Over	169
E.18.3.1 DUMP	170
E.18.3.2 SPLIT	170
E.18.3.3 IMAGE (existing)	170
E.18.3.4 IMAGE (new)	171
E.18.3.5 PAGE	171
E.18.3.6 InnoDB Flushing	171
E.18.3.7 Pipelining	172
E.18.3.8 InnoDB Deleting	173
E.19 Messages	173
E.19.1 <code>/bin/bash</code>	173
E.19.2 Filename Issues	173
E.19.3 <code>gm convert</code> (GraphicsMagick)	174
E.19.4 <code>convert</code> (ImageMagick)	175
E.19.5 <code>graphicsmagick-imagemagick-compat</code>	175
E.19.6 <code>dvips</code>	176
E.19.7 PHP Notice: Undefined index:	176
E.19.8 PHP Notice: <code>xml_parse()</code>	177
E.19.9 PHP Warning	177
F Trends (2011-Dec-21)	178
F.1 History	178
F.2 Main Components	179
F.3 Size Distribution	179
F.3.1 Size Distribution—by Language	179
F.3.2 Size Distribution—over Time	179
F.3.3 Size Distribution—by Age of Article	180
F.4 Namespace Distribution	181
GNU Free Documentation License	185
1. APPLICABILITY AND DEFINITIONS	185
2. VERBATIM COPYING	186
3. COPYING IN QUANTITY	186
4. MODIFICATIONS	187
5. COMBINING DOCUMENTS	188
6. COLLECTIONS OF DOCUMENTS	188

7. AGGREGATION WITH INDEPENDENT WORKS	189
8. TRANSLATION	189
9. TERMINATION	189
10. FUTURE REVISIONS OF THIS LICENSE	190
11. RELICENSING	190
ADDENDUM: How to use this License for your documents	190

List of Figures

2.1	WP-MIRROR Monitor Mode in X	20
3.1	Recommended Disk Configuration.	39
4.1	WP-MIRROR in Monitor Mode with GUI Display	50
4.2	WP-MIRROR in Monitor Mode with Screen Display	50
4.3	WP-MIRROR in Monitor Mode with Text Display	51
5.1	FSM State Transition Diagram	67
5.2	WP-MIRROR Instances Must Communicate State <i>Only</i> via the <code>wpmirror</code> Database	70
5.3	WP-MIRROR Monitor Mode Progress Bars	88
E.1	InnoDB Experiments—Importing <code>imagelinks</code> Table	153
F.1	Number of Articles on en Wikipedia	178
F.2	Size Distribution Over Time on en Wikipedia	180
F.3	Size Distribution by Age of Article on en Wikipedia	180
F.4	Size Distribution by <code>ichunk</code> on en Wikipedia	181
F.5	Namespace Distribution by Age of Page on en Wikipedia	182
F.6	Page in Category Namespace	183
F.7	Page in Help Namespace	183
F.8	Page in Main Namespace	183
F.9	Page in Portal Namespace	183
F.10	Page with Small Template	184
F.11	Page with Large Template	184
F.12	Page in Wikipedia Namespace	184
F.13	Page in Book Namespace	184

List of Tables

1.1	History of WP-MIRROR Features	5
2.1	History of WP-MIRROR Support for Operating Systems	11
2.2	History of WP-MIRROR Packages	15
2.3	History of <code>clisp</code> Version	16
2.4	History of <code>common-lisp-controller</code> Configuration	17
2.5	History of <code>cl-asdf</code> Configuration	17
2.6	WP-MIRROR Build Options Reference	18
2.7	WP-MIRROR Build Options Reference (Obsolete)	19
3.1	Automation of <code>laptop</code> Configuration	24
3.2	History of Configuring <code>hdparm</code>	24
3.3	History of Configuring <code>images</code> Directory	25
3.4	History of Configuring <code>hddtemp</code>	25
3.5	History of Configuring <code>smart</code> Disk Monitoring	25
3.6	History of Configuring MySQL Security	26
3.7	History of Configuring MySQL <code>time_zone</code> Tables	27
3.8	History of Configuring InnoDB Storage Engine	28
3.9	History of Configuring <code>cURL</code>	29
3.10	History of Configuring MediaWiki	30
3.11	History of Configuring <code>mediawiki-extensions</code>	31
3.12	History of Configuring Image Processing	32
3.13	History of Configuring SVG to PNG Conversion	33
3.14	History of Configuring <code>apache2</code> Virtual Host	34
3.15	History of Configuring <code>bash</code> for Caching Web Proxy	35
3.16	History of Configuring <code>cURL</code> for Caching Web Proxy	35
3.17	History of Configuring <code>wget</code> for Caching Web Proxy	36
3.18	History of Configuring Browsers for Caching Web Proxy	36
3.19	Security Policy	39
4.1	WP-MIRROR Mirror Mode Options Reference	48
4.2	WP-MIRROR Mirror Mode Options Reference (Obsolete)	48
4.3	WP-MIRROR Monitor Mode Options Reference	49
5.1	WP-MIRROR <code>PID</code> File Logic	54
5.2	FSM Tables <code>*state-transition*</code> and <code>*type-state-function*</code>	67
5.3	FSM Priority Table <code>*type-state-priority*</code>	68
B.1	WP-MIRROR Configuration File Parameter Reference	99
B.2	WP-MIRROR Configuration File Parameter Reference (Obsolete)	103
C.1	WP-MIRROR Error Codes	105
C.2	WP-MIRROR Error Codes (Obsolete)	106

E.1	InnoDB Experiments—Codes	151
E.2	InnoDB Experimental Results—Performance Ratios	152
E.3	InnoDB Experimental Results—Summary	152
E.4	Speed of <code>MWdumper.jar</code> v. <code>importDump.php</code>	153
E.5	Processes that Cause Resource Contention	165
F.1	Size Distribution by Language	179
F.2	Wikipedia Namespaces	182

Chapter 1

General Information

The [WikiMedia Foundation](#) offers wikipedias in nearly [300 languages](#).

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

WP-MIRROR 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.

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

- **Installation.** If you wish to build a wikipedia 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 (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 40G on a hard disk drive,
 - be served locally by a virtual host <http://simple.mediawiki.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 wikipedia farm.** Mirroring a set of wikipedias requires some configuration. This could be easy or hard depending on how large the selected wikipedias are. Therefore, before attempting to build your own wikipedia farm, please read [Chapter 3, System Planning and Configuration](#), as it may save you weeks or months of effort. Really.
- **Mirror a top ten wikipedia.** The ten largest wikipedias are judged to be too large to fit on a laptop PC equipped with a single 500G hard disk drive. The top ten languages are: the [en](#) wikipedia (the largest at about 2T), followed by the [de](#), [fr](#), [nl](#), [it](#), [es](#), [pl](#), [ru](#), [ja](#), and [pt](#) 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 0.4.

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 originally 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>](mailto: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](#). WP-MIRROR is a free utility for mirroring any desired set of these wikipedias. That is, it builds a wikipedia 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 [MySQL](#), [MediaWiki](#), [cURL](#), and [apache2](#) 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 (Simple English means shorter sentences). It should build in 200ks (two days), occupy 40G of disk space (which should fit on most laptops), be served locally by a virtual host <http://simple.mediawiki.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 wikipedias 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.mediawiki.site/>, one for each wikipedia 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>

- **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). 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.

1.3.3 The Main Features of WP-MIRROR

Functionality:

- **Access.** WP-MIRROR creates a virtual host, so that the user may access the mirrored wikipeidias locally using a web browser.
- **Mirror.** WP-MIRROR can run in two modes. In mirror mode, it builds a mirror of a one or more wikipeidias, 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.

Efficiency:

- **Concurrency.** Multiple instances of WP-MIRROR can run concurrently. This is done to accelerate the mirroring of the largest wikipeidias (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.
- **Image validation.** Many image files turn out to be corrupt, so WP-MIRROR validates all downloaded image files. The validation process is CPU intensive. Since this is a process that one should be loathe to repeat, WP-MIRROR keeps track of all image files that have been validated.

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](#) requires 40G, [en](#) requires 2T, as of year 2012).
- **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 [MySQL](#), [MediaWiki](#), [cURL](#), [apache2](#), and others.
- **Automated dependency configuration.** WP-MIRROR automatically configures dependencies such as [MySQL](#), [MediaWiki](#), [cURL](#), [apache2](#), and others. For the default case (a clean install of [simple](#) on a laptop) everything should ‘just work’.
- **Check-pointing.** WP-MIRROR may be interrupted (power failure, user closes laptop, cat walks across keyboard) and resumed later. This is important because building a mirror of one of the top ten wikipeidias could take days to 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.
- **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.
- **Warnings.** WP-MIRROR issues warnings if [MySQL](#) is insecure (e.g. has a root account with no password), if disk space may be inadequate, and if a caching web proxy is detected. These warnings are intended as a convenience for the system administrator.

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
0.4	Assert minimum disk space for images and InnoDB Automated default configuration for dependencies Command-line option <code>--restore-default</code> Shell commands and scripts now POSIX-compliant Warn if MySQL is insecure, disk space low, or proxy detected WP-MIRROR Reference Manual provided in TeX 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

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 0.4

Phase	alpha
Packaging	DEB, RPM, 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 [cURL](#). 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](#) wiki). 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 [\$\text{\LaTeX}\$](#) and PDF format.** WP-MIRROR 0.4 includes documentation in [PDF](#) format (this document), built from [\$\text{\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), if disk space is inadequate, or if a proxy is detected. 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.

1.5.2 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.

1.5.3 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 built-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.

1.5.4 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 [cURL](#), 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.

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>](mailto: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/manual/>. 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.

- **Benjamin Goldsmith.** Testing installation and operation on Debian GNU/Linux 6.0 (squeeze). Submitting bug reports.
- **Tylery Khai.** Recommending color scheme for monitor mode (`--gui` option).

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	Version	Architecture
0.4	Linux	3.2.0	amd64
0.3	Linux	2.6.32	amd64

WP-MIRROR has been reported to build successfully on the above operating systems. See §2.2, [Installing WP-MIRROR on Linux](#).

WP-MIRROR is written using `clisp` 2.48, so it should be possible to port WP-MIRROR to other systems that have `clisp` 2.48 or later.

2.1.2 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.

2.1.3 Choosing Which WP-MIRROR Distribution to Install

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

2.1.3.1 Choosing Which Version of WP-MIRROR to Install

Choosing the most recent version is recommended, as it offers more features with fewer bugs.

2.1.3.2 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 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.

2.1.3.3 Choosing When to Upgrade

If you are using a major distribution, such as Debian 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](#), [MySQL](#), [MediaWiki](#), 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.

2.1.4 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 against partial downloads and alteration. All WP-MIRROR packages are cryptographically signed using [GnuPG](#), the GNU Privacy Guard.

2.1.4.1 How to Get the Author’s Public Build Key

All tarballs and documentation are signed using [GnuPG](#). DEB packages may be signed as well (see §2.1.4.2, [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 §2.1.4.2, [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:

2.1.4.1.1 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).

2.1.4.1.2 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

mQENBE72h9MBCADqNjKAJs1jj90xqM6m7iTk211V7g2U+PiLujnEmvg/BokAtoFzLHIgBYw8
oemQgfp0AqOzYirQKnzBFTL/BaOSNv/TVGHXgK349GzJZ1/MY8rf9hvoSoNSV7RIGEB0U5a6
Bv47k4CYCDck++jAifTpk3yAgyW7JOZ3uzwEYrvIdEW/4JZuli4ZMmZIja+o6Q4lbXylwzoq
KKBtLWcofdpNKoS+/3Mq1wENN+d3c8SjqH8NL4tMCyF4LXC0CpSxfxUiMY1d8ifl+BjpguDl
jod+UqKEKigXep7kiwDE09t01RLFYlWbCRuw+qbfUXi0o6tTnQPiE06xqr3f6HywnTxrABEB
AAGOIVdQLU1JULJPuiA8d3BtaXJyb3JkZXZA21haWwuY29tPokBOAQTAAIAIgUCTvaH0wIb
AwYLCqgHAWIGFqGCCQoLBBYCAwECHgECF4AACGkQMGr8nTgvvQyK5QgAv62RuSJ/VB8V5ALj
1+J3Q4CRQjmwNncUbc7xvSJPbRjl3CeZSd7Mf1EjevA10japTGwrUboPeOyYPupxin2I4jMY
rWHTIfqtgh03YkxYaoK2MTrNKbYUBmzM5I9jQ49aNYfM3ikhaHMUIUFFzTV6nRmekEo1L2o2
wlVWmw2j/naF8KaHXL3X+dnNBZDz6kuBTo3MI3oIR5mRuVhj9ppbS6qYF3pmVmS2agj2186D
thSrD4JBBUhzqPriPS43JPEs1L696LcNgB0Qq70964ZiNrTyu/FBIkGGk160ly4zALApE8V
XmaYMwnJQwSVtt8WLD250S/Trb+C05WQfbvMybkBDQR09ofTAQgA6n4xWXt52PfWEYsXPuDb
La57M1KiyFPKHrViFR5ic9xNiY4H5P3Iryp1plKMBkUCI6TdKNXcMYR1X4tugYtyq/LyxYgt
82f2eJdVFq3wbFPt/eM0tMn05n+K+4J8ptu+qkwyr1VkaAPofbtQG1Zwb9wQmrNMZdJg6i9B
7vPvqjGaZArtrn9Gcqu8yttt/OMc/Pc9Y14iuCpkL4QeBMhAuKBBOAqGGYcovIP7w1RmvG05
ofKmtY1zATqGXknZpo0HSbaVcn3GxkrUNpcE2SZDFUOC82EJ7So3tzcCIRJvJU0yE+QP1IY
/XYV/uwpcxAHma36E9u/a+o3eIXft8AVxwARAQABiQEfBBgBAGAJBQJO9ofTAhsMAAoJEDIK
/JO4L70m81UIAJVHywQvU6EAZ62twjmnBfcwno0sL/VHrTsUeOoY+CjZj6p2EsGQio8wC97
wwol7fc9/8UUCqdqVxIFY30uyHsSjbsYqQ58o7e0P6hulBPc/zfw2jhpX3J4kTkyX7CgPWvd
C+WHHnmzdyBPuv+sPhsJTziVb/gr+Jzti3v85a+YtYwiGXV/7o2R373ULJFL158veV4gF5N8
IMrXP1U+CKTXvwS9qjULy7YrfrKiBiJ70kNkA2m+T79NXNbnUB4GHw8tS7T7bnA0c/E0Ajs
6JQCCEf3xSPS3Q8aEp4mxDUBZe7FWt1JJbvFlAV7PGmcMLZLj64j6zrbSikF+kxibtEI=
=wNKp
-----END PGP PUBLIC KEY BLOCK-----
```

Third, import the public key (now in `wpmirror_pubkey.asc`) into your `GPG` keyring as follows:

```
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).

2.1.4.2 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 §2.1.2, [How to Get WP-MIRROR](#).

First, verify the `.changes` file by executing:

```
shell$ gpg --verify wp-mirror_0.3-2.changes
gpg: Signature made Tue 06 Mar 2012 08:54:25 PM 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_0.3-2_all.deb
83411c3d750e85764545d7f86728febd639f7c38 wp-mirror_0.3-2_all.deb
shell$ cat wp-mirror_0.3-2_amd64.changes | grep all | grep deb | head -n 1
83411c3d750e85764545d7f86728febd639f7c38 112522 wp-mirror_0.3-2_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.

2.1.4.3 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 §2.1.2, [How to Get WP-MIRROR](#).

To verify the `RPM` 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).

2.1.4.4 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 §2.1.2, [How to Get WP-MIRROR](#). Be sure to put both files into the same directory, then execute:

```
shell$ gpg --verify wp-mirror-0.4.tar.gz
gpg: Signature made Fri 09 Mar 2012 03:17:24 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 2.2: History of WP-MIRROR Packages

Version	Package	Distributions
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 0.4 was packaged for use with the Debian GNU/Linux 7.0 (wheezy) distribution.

First, download the latest `DEB` package as described above in §2.1.2, [How to Get WP-MIRROR](#).

Second, install the `DEB` package. Beginning with version 0.4, this is done by executing:

```
root-shell# dpkg --install wp-mirror_0.4-1_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

The following packages must be installed before you can build WP-MIRROR 0.4.

```
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 2.3: History of [clisp](#) Version

Version	Configuration
≤ 0.4	clisp 2.48

Check that you have [clisp](#) 2.48 or higher.

```
shell$ clisp --version
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 2.4: 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 2.5: 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 0.4 to build a mirror.

Depends:

```
apache2 (>= 2.2.16), bzip2 (>= 1.0.5), cjk-latex (>= 4.8.3+git20120621-1),
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),
coreutils (>= 8.5), curl (>= 7.21.0),
gv (>= 1:3.7.1), graphicsmagick (>= 1.3.12),
hdparm (>= 9.32), inkscape (>= 0.48.3.1-1.1),
libsvg2-bin (>= 2.26.3), mediawiki (>= 1:1.19.1),
mediawiki-extensions (>= 2.6+wheezy1),
mediawiki-extensions-math (>= 2:1.0+git20120528-5),
mysql-client (>= 5.5.24+dfsg-9), mysql-server (>= 5.5.24+dfsg-9),
openssl (>= 0.9.8o),
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 2.6, WP-MIRROR Build Options Reference](#).

Table 2.6: WP-MIRROR Build Options Reference

Build Options	Description	Intr	Rem
<code>--copyright</code>	generate <code>copyright</code>	0.1	
<code>--help</code>	used by <code>help2man</code> to generate <code>wp-mirror.1</code>	0.1	
<code>--version</code>	used by <code>help2man</code> to generate <code>wp-mirror.1</code>	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 2.7, WP-MIRROR Build Options Reference \(Obsolete\)](#).

Table 2.7: WP-MIRROR Build Options Reference (Obsolete)

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

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](#).

2.4 Postinstallation Configuration and Testing

After installation, we note that WP-MIRROR relies on a great number of binary dependencies, such as [apache2](#), [cURL](#), [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 Wikipedia 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 Mode Works](#). If any message indicates a warning or a failure, please read that section.

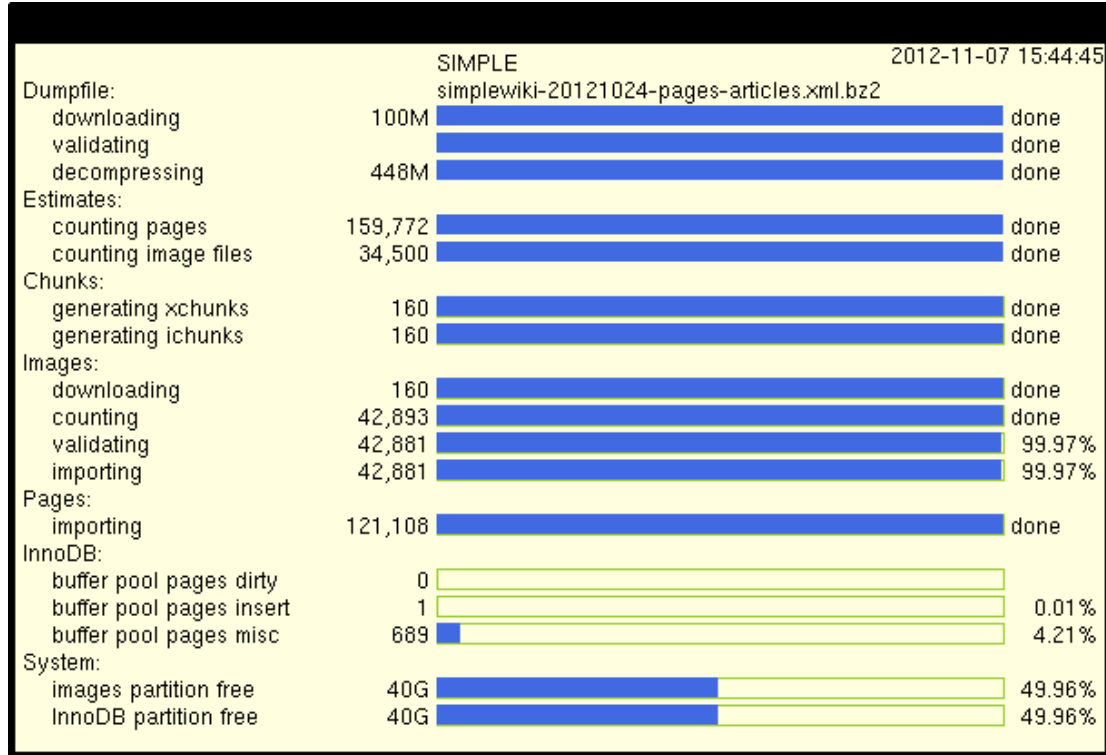
It is convenient to monitor the mirror by opening a second terminal and executing one of the following:

```
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 a 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.

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:

```
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 Wikipedia 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 to see which wikis are available.

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

Third, edit the configuration file `/etc/wp-mirror/local.conf` by uncommenting and editing the `*mirror-languages*` 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-languages* '("simple"))
;;;
;;; For a quick test. Expect 5ks (one hour) build.
;;(defparameter *mirror-languages* '("zh" "zu"))
;;;
;;; Major third-world languages.
;;(defparameter *mirror-languages* '("ar" "simple")) ;arabic
;;(defparameter *mirror-languages* '("hi" "simple")) ;hindi
;;(defparameter *mirror-languages* '("vi" "simple")) ;vietnamese
;;(defparameter *mirror-languages* '("zh" "simple")) ;chinese
;;;
;;; Classical languages.
;;(defparameter *mirror-languages* '("el" "la" "simple")) ;greek,latin
;;(defparameter *mirror-languages* '("he" "yi" "simple")) ;hebrew,yiddish
;;(defparameter *mirror-languages* '("zh-classical" "simple")) ;chinese
...
```

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, [en](#), [de](#), [fr](#), [nl](#), [it](#), [es](#), [pl](#), [ru](#), [ja](#), and [pt](#) (as of year 2012) then reading this chapter is required. Why?

- **Size.** Currently (year 2012), 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). 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 [MySQL](#), [MediaWiki](#), [cURL](#), and image processing. 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

The Wikimedia Foundation maintains about a thousand wiki's, including wikipedias for nearly 300 languages. The Foundation provides compressed dump files of these wiki's. These files have names resembling

[simplewiki-yyyyymmdd-pages-articles.xml.bz2](#), and
[enwiki-yyyyymmdd-pages-articles.xml.bz2](#).

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 string [yyyyymmdd](#) 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 with names something like

<http://dumps.wikimedia.org/simplewiki/yyyyymmdd/...>, and
<http://dumps.wikimedia.org/enwiki/yyyyymmdd/...>

These dump files contain the latest revisions of articles, templates, image descriptions, and the primary meta-pages. They do not contain talk, old revisions, or user pages.

Images are stored separately from everything else. Images that are used by more than one wiki are stored in a directory tree under the URL

<http://upload.wikimedia.org/wikipedia/commons/>.

Images that are unique to a given wiki have their own directory tree. You can expect URL's with names like

<http://upload.wikimedia.org/wikipedia/simple/>, and
<http://upload.wikimedia.org/wikipedia/en/>.

As for your computer: Articles, templates, image descriptions, etc. are stored in the form of a database. We use [MySQL](#) as the database management system. [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 wikipedia, 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.

The author recommends using a modern journalling file system (such as [ReiserFS](#)). The Wikimedia Foundation does not provide a dump file of images. Nor does it provide any kind of bulk download method (e.g. torrent files). Copyright and bandwidth issues have been cited. There is however a free software [wikix](#), a C language program written by Jeffrey Vernon Merkey (see <http://meta.wikimedia.org/wiki/Wikix>) that can read through a dump file, parse the names of image files, and generate a shell script for downloading said images. These images are then individually downloaded. WP-MIRROR provides a built-in alternative (default) to [wikix](#) that has a number of advantages. Again WP-MIRROR automates all these tasks for you.

When you wish to access your mirror, open a browser and enter the URL <http://mediawiki.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 resize images, respectively. Then the web server will serve the articles and resized images. Finally, your browser will display everything.

Currently (year 2012), 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](#) wiki (the largest at about 2T), [de](#), [fr](#), [nl](#), [it](#), [es](#), [pl](#), [ru](#), [ja](#), and the [pt](#) wiki. 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 40G (as of year 2012) 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](#) wiki (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](#) wiki. 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](#) wiki.

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

3.2.1 Plan Your Disk Space

3.2.1.1 Disable HDD Write Caching

Automatic disabling of HDD write caching was introduced with WP-MIRROR 0.1. No user configuration is necessary.

Table 3.1: Automation of `laptop` Configuration

Category	Configuration	Auto	Warn	Opt
Disk	Disable HDD Write Caching	Y		
	Assert Adequate Disk Space		Y	
	Setup Thermal Management			Y
	Setup <code>smart</code> Disk Monitoring			Y
DBMS	Secure Database Management System		Y	
	Load Time Zone Tables	Y		
	Configure <code>InnoDB</code> Storage Engine	Y		
DRAM	N/A			
Internet	Configure <code>cURL</code>	Y		
MediaWiki	Configure <code>MediaWiki</code>	Y		
	Enable <code>MediaWiki</code> extensions	Y		
Images	Replace <code>ImageMagick</code> with <code>GraphicsMagick</code>	Y		
	Replace <code>SVG Converter</code>	Y		
Host	Enable Virtual Host	Y		
Proxy	Configure <code>bash</code> for Use with Caching Web Proxy			Y
	Configure <code>cURL</code> for Use with Caching Web Proxy			Y
	Configure <code>wget</code> 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
≥ 0.1	automated, no user configuration is necessary

Write caching is disabled only for the disk(s) holding your `InnoDB` data. The disabling of HDD write caching is done to assure Durability (the ‘D’ in ‘ACID compliance’) of committed transactions.

You may manually disable write caching for a hard drive by configuring `/etc/rc.local` by inserting the following lines:

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

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

A mirror of the `simple` wikipedia (the default), requires about 40G (as of year 2012). Its image files are stored under `/var/lib/mediawiki/images/`.

Table 3.3: History of Configuring `images` Directory

Version	Configuration
≥ 0.4	warning issued if <code>/var/</code> has <50G free space; user configuration optional (see below)
≤ 0.3	user configuration optional (see below)

WP-MIRROR periodically checks the amount of free disk space. If `/var/` has less than 50G 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. It may not be necessary to repartition your HDD. If you mounted `/home/` on its own partition (which you should), 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
≥ 0.1	user configuration optional (see below)

Running WP-MIRROR puts quite a load on CPU, HDD, and memory. 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
```

If your disks are running hot (e.g. over 50°C), they will not last. 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 baked cookies cool). Put the cookie rack under your laptop PC.

3.2.1.4 Setup `smart` Disk Monitoring

Table 3.5: History of Configuring `smart` Disk Monitoring

Version	Configuration
≥ 0.1	user configuration optional (see below)

It is important to monitor the health of all your HDD's. Disks age, wear out, and fail. However, most disks 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/../../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)

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

Security testing of the database management system (DBMS) was introduced with WP-MIRROR 0.4. Warnings are issued as needed. User configuration is optional by highly recommended.

For WP-MIRROR 0.4 and earlier versions, 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)

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](#)

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.

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.

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 Configure the InnoDB Storage Engine

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 custom_laptop.cnf /etc/mysql/conf.d/.
root-shell# cd /etc/mysql/conf.d/
root-shell# mv custom_laptop.cnf custom.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/custom.cnf` to read:

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

# to save disk space (compress tables using 'zlib')
innodb_file_per_table          # default OFF.
innodb_file_format = Barracuda # default Antelope.
innodb_file_format_check

# increase buffer pool and log files for speed
innodb_buffer_pool_size = 256M # default 128M.
innodb_log_file_size = 10M     # default 5M.

# for Durability (the 'D' in 'ACID compliance')
innodb_flush_log_at_trx_commit = 1 # default 1.
sync_binlog = 1                  # default 0.
max_allowed_packet = 64M         # default 16M.

[mysql]
default-character-set = utf8
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. 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. The author recommends **Barracuda** for mirroring small wikipedias (but not the top ten).

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 journalling system. Storing the **table space** on top of a journalling file system (case 1) is unnecessary and slow—and it makes the HDD's run harder and hotter. **Antelope** also handles concurrency well. It is normal for a 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. The author recommends it (case 3) for mirroring the **en** wiki and other large wikipedias (i.e. the top ten).

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** wiki, 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.

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**

Table 3.9: History of Configuring **cURL**

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

Images are downloaded using the utility **cURL**, and, by default, **cURL** does not time-out when a file takes a long time to download. Some image files do fail to download completely, and **cURL** blocks when that happens.

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 uses 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)

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)

Without [MediaWiki](#) extensions enabled, most of the Wikipedia articles will look messy.

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

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.

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

By default, [MediaWiki](#) processes images using `convert` from [ImageMagick](#). 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/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

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](#). 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, [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](#).

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 Plan Your Virtual Host

3.2.7.1 Enable Virtual Host

WP-MIRROR sets up a virtual host named `mediawiki.site` on your own computer. This lets you access your mirror locally by giving your web browser the URL `http://simple.mediawiki.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.mediawiki.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/mediawiki.site.conf` with the following text:

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

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

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

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

```
:::1 mediawiki.site www.mediawiki.site
:::1 simple.mediawiki.site
```

Finally, for each language specified in `/etc/wp-mirror/local.conf`, append a corresponding line 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# a2disssite mediawiki.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://my-proxy:8123/
http_proxy  = http://my-proxy:8123/
ftp_proxy   = http://my-proxy:8123/
```

where `my-proxy` 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
≤ 0.4	user configuration required (see below)

Most web-access programs look for environment variables to determine whether or not traffic must go through a caching web proxy. These environment variables can be set globally by editing `/etc/bash.bashrc` by appending:

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

where `my-proxy` 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://my-proxy:8123/"
declare -x HTTPS_PROXY="http://my-proxy:8123/"
declare -x HTTP_PROXY="http://my-proxy: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

Table 3.16: History of Configuring `cURL` for Caching Web Proxy

Version	Configuration
≤ 0.4	user configuration required (see below)

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

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

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

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

`cURL` does not have a global configuration file. So you 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
≤ 0.4	user configuration required (see below)

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

`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://my-proxy:8123/
http_proxy  = http://my-proxy:8123/
ftp_proxy   = http://my-proxy:8123/
```

where *my-proxy* 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
≤ 0.4	user configuration required (see below)

If your system sits behind a caching web proxy, then you may find your self wondering why every attempt to browse <http://simple.wikimedia.site/> gets mis-routed. If this is the case, 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 months 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 months.

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

3.3.1 Procure Hardware

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 2T (or more) each,
- memory - qty 1 set, size 4G,
- 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,
- 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 Disable HDD Write Caching

Automatic disabling of HDD write caching was introduced with WP-MIRROR 0.1. No user configuration is necessary (see §3.2.1.1, [Disable HDD Write Caching](#) above).

You may manually turn off write caching for the disks that hold the `InnoDB table space` by configuring `/etc/rc.local` by appending the following lines:

```
hdparm -W0 /dev/sd[bc]      # table space stored here
exit 0
```

3.3.2.2 Setup Thermal Management

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 will be shifted an octave lower, which for many people is easier to bear.

3.3.2.3 Setup `smart` Disk Monitoring

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/../../../../03)
/dev/sdc -a -d ata -o on -S on -s (S/../../../../02|L/../../../../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/../../../../03)
/dev/sdc -a -d usbjmicron,1 -o on -S on -s (S/../../../../02|L/../../../../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` wiki require about 200G (as of 2012).

Articles are held in the database `enwiki`, which is stored by `InnoDB` in the `table space`, which is usually named `/var/lib/mysql/ibdata0`.

Put the `InnoDB table space` on a disk separate from that holding the `InnoDB` log files `/var/lib/mysql/ib_logfile[01]`. This can double disk access performance.

Use `Antelope` with the `table space` written directly onto a raw partition, as in case 3 described in the Design Note in [§3.2.2.3, Configure the `InnoDB` Storage Engine](#).

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` wiki require about 2T (as of 2012).

Images are stored under the directory `/var/lib/mediawiki/images/`. However, given its size, 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.

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

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

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

Table 3.19: Security Policy

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

and this should be done daily.

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.

3.3.2.7 Build Your Storage Array

We shall build a [RAID](#) array out of whole disks, and then encrypt that. The author recommends the following structure (note: `ibdata0` 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

To allow for future growth in the [en](#) wiki, the author recommends procuring a pair of HDDs no smaller than 2T. The following instructions assume 2T (0.3T for [table space](#), 1.7T for [images](#)).

Figure 3.1: Recommended Disk Configuration.

Data	<code>/var/lib/mysql/ibdata0</code>	<code>/var/lib/mediawiki/images/</code>
Filesystem	<code>none</code>	<code>ReiserFS</code>
LVM2	<code>/dev/mapper/vg0-ibdata0</code>	<code>/dev/mapper/vg0-images0</code>
LUKS	<code>/dev/mapper/xts_database0</code>	
RAID1	<code>/dev/md0</code>	
Whole disks	<code>/dev/sdb</code>	<code>/dev/sdc</code>

One builds this structure from the bottom up.

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 2T HDD's.

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]
```

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

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

Each of the above `badblocks` commands should take about 50ks (14 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`.

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
```

3.3.2.7.3 LUKS Encryption is unnecessary for public information like a set of wikipedias. However, **InnoDB Antelope** (default) stores all tables in a single `table space`. 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 disk. 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).

3.3.2.7.4 LVM2 Next we wish to partition the [LUKS](#) partition: 0.3T for `ibdata0`, and 1.7T (the rest) 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 401931 --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.

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,
- [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](#).

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

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

```
/dev/vg0/images0 /database0/images reiserfs defaults 0 2
```

Finally, test the mount procedure by executing:

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

3.3.2.7.6 Raw Partition 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

Automatic securing of the database management system (DBMS) was introduced with WP-MIRROR 0.4. No user configuration is necessary (see [§3.2.2.1, Secure the Database Management System](#) above).

3.3.3.2 Load the DBMS `time_zone` Tables

Automatic loading of the DBMS `time_zone` tables was introduced with WP-MIRROR 0.4. 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

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 = 1 # default 1.
sync_binlog = 1                    # 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 the OS swap algorithm could inadvertently break up `InnoDB` pages.

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 wikis without first finding adequate physical memory (at least 4G default).

3.3.4.1 Hugepages

Allocate say 4G of DRAM as `hugepages` by configuring `/etc/sysctl.conf` by adding the lines:

```
vm.nr_hugepages = 2048
vm.hugetlb_shm_group = 1001
kernel.shmmax = 4294967296
kernel.shmall = 1048576
```

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
```

3.3.4.3 Buffer Pool

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

This was done above in §3.3.3, [Customize the InnoDB Storage Engine](#).

3.3.4.4 Process Limits

Relax `mysqld` process limits by configuring `/usr/bin/mysqld_safe` by appending the line:

```
ulimit -l unlimited
```

3.3.5 Plan Your Internet Access

3.3.5.1 Configure `cURL`

Automatic configuration of `cURL` was introduced with WP-MIRROR 0.4. No user configuration is necessary (see §3.2.4.1, [Configure `cURL`](#) above).

3.3.6 Plan Your `MediaWiki`

3.3.6.1 Configure `MediaWiki`

Automatic configuration of `MediaWiki` was introduced with WP-MIRROR 0.4. No user configuration is necessary (see §3.2.5.1, [Configure `MediaWiki`](#) above).

3.3.6.2 Enable `MediaWiki` Extensions

Automatic enabling of the `mediawiki-extensions` was introduced with WP-MIRROR 0.4. No user configuration is necessary (see §3.2.5.2, [Enable `MediaWiki` Extensions](#) above).

3.3.6.3 `MediaWiki` Redux

Same as in §3.2.5.3, [MediaWiki Redux](#) above.

3.3.7 Plan Your Image Processing

3.3.7.1 Replace `ImageMagick` with `GraphicsMagick`

Automatic replacement of `ImageMagick` with `GraphicsMagick` in `MediaWiki`'s configuration was introduced with WP-MIRROR 0.1. No user configuration is necessary (see §3.2.6.1, [Replace `ImageMagick` with `GraphicsMagick`](#) above).

3.3.7.2 Replace `SVG` Converter

Automatic replacement of `ImageMagick` in `MediaWiki`'s configuration was introduced with WP-MIRROR 0.1. No user configuration is necessary (see §3.2.6.2, [Replace `SVG` Converter](#) above).

3.3.8 Plan Your Virtual Host

3.3.8.1 Enable Virtual Host

Automatic enabling of the virtual host was introduced with WP-MIRROR 0.2. No user configuration is necessary (see §3.2.7.1, [Enable 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](#) wiki dump file is 8G (as of 2012). 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-yyyyymmdd-pages-articles.xml.bz2 .
root-shell# rm *wip
```

Check that there is a `.head` file and no `.wip` file. Then restart WP-MIRROR. It will find the dump file, and handle matters from there.

Design note. The downloading of large dump files has been a problem.

1. To avoid downloading a dump that we already have, yet quickly recognize when a new dump file is available, WP-MIRROR stores the dump file’s HTTP header in a small file with a name like: `enwiki-yyyyymmdd-pages-articles.xml.bz2.head`. The `.head` file contains the ‘ETag’ of the dump file, which is then compared with the ‘ETag’ of the dump offered by the Wikimedia Foundation.
2. To recognize when a download is incomplete, WP-MIRROR first creates an empty file with a name like `enwiki-yyyyymmdd-pages-articles.xml.bz2.wip` and then begins downloading. If the download completes, the `.wip` file is deleted. (‘wip’ means ‘work-in-progress’). If an instance of WP-MIRROR sees a `.wip` file, it assumes that a partial download occurred. In this case, the file is deleted, and the download re-tried.

3.3.9.2 Download Millions of Image Files

Some caching proxies might not have disk space adequate for caching nearly 2T of images. Usually `/var/` is mounted on its own partition (and this is highly recommended). Now, a caching web proxy usually keeps its cache on the `/var/` 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. This includes the system log files under `/var/log/`, the loss of which is not too tragic. However, if the system administrator had the bad judgement to put a mail transfer agent (MTA) (e.g. `sendmail`, `exim`, etc.) on the same box, then stuffing `/var/` will hold up the mails. 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 `bash` for Use with Caching Web Proxy

Same as [§3.2.8.1, Configure `bash` for Use with Caching Web Proxy](#) above.

3.3.9.5 Configure `cURL` for Use with Caching Web Proxy

Same as [§3.2.8.2, Configure `cURL` for Use with Caching Web Proxy](#) above.

3.3.9.6 Configure `wget` for Use with Caching Web Proxy

Same as [§3.2.8.3, Configure `wget` for Use with Caching Web Proxy](#) above.

3.3.9.7 Configure Browser for Use with Proxy

Same as [§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](#).

Table 4.1: WP-MIRROR Mirror Mode Options Reference

Runtime Options	Description	Default	Intr	Rem
<code>--copyright</code>	display copyright, and exit		0.1	
<code>--debug</code>	verbose output		0.1	
<code>--delete language</code>	delete files and state information (but keep database and images), for given language, and exit		0.4	
<code>--drop language</code>	drop database, delete files and state info (but keep images), for given language, and exit		0.4	
<code>--help</code>	display help, and exit		0.1	
<code>--mirror</code>	force mirror mode (overrides any monitor mode options or defaults)		0.1	
<code>--restore-default</code>	drop all databases and files (except images), and start over (dangerous)		0.4	
<code>--version</code>	display version and copyright, and exit		0.1	

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

Runtime Options	Description	Default	Intr	Rem
<code>--delete-dump language</code>	delete dump file and state information for given language, and exit		0.1	0.4

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.

Table 4.3: WP-MIRROR Monitor Mode Options Reference

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

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](#).

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](#).

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

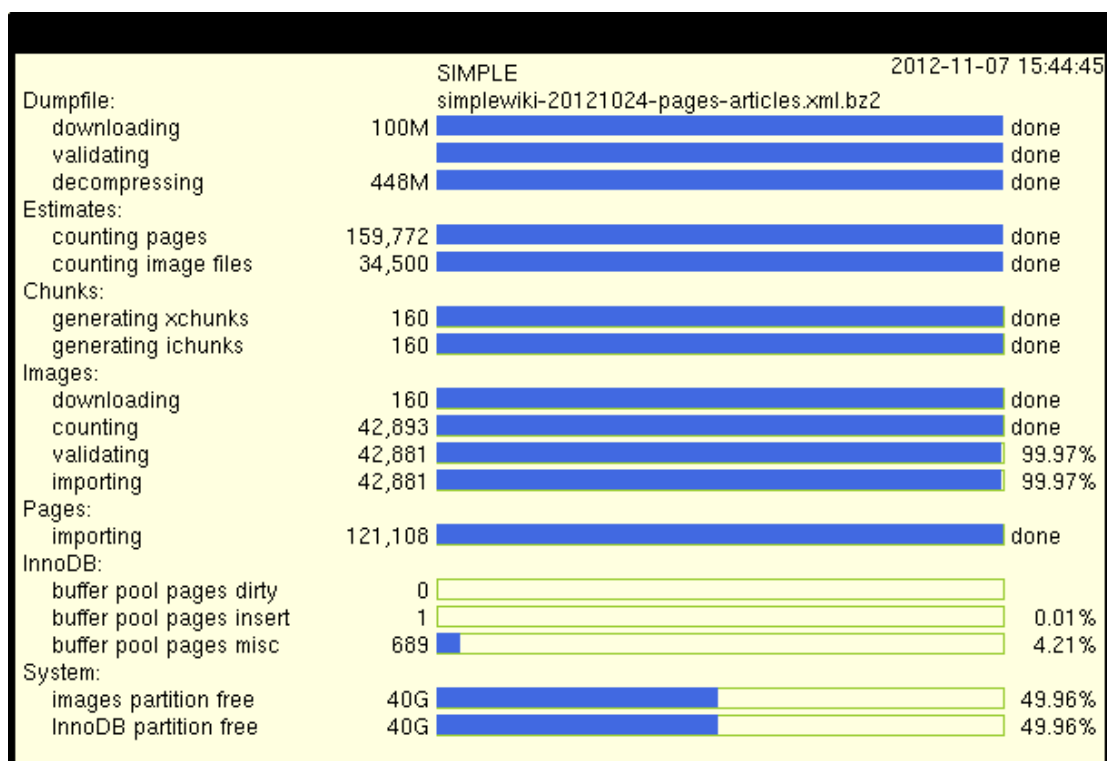


Figure 4.2: WP-MIRROR in Monitor Mode with Screen Display

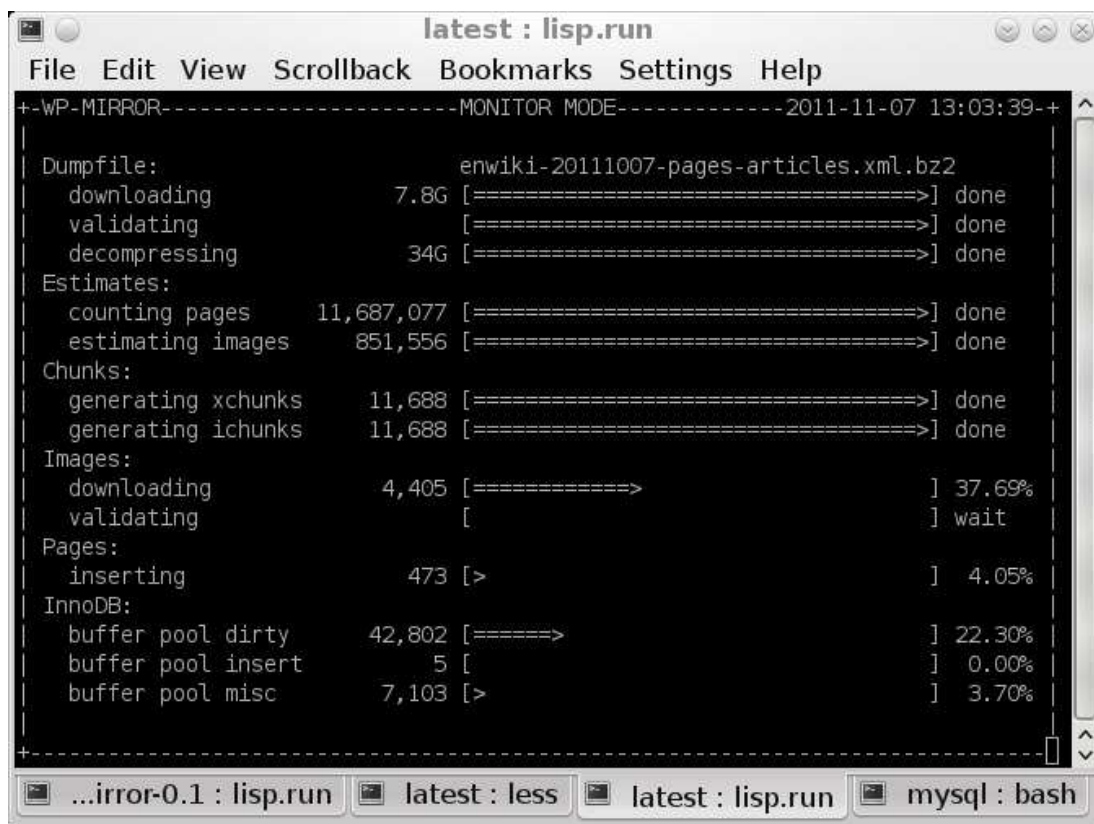
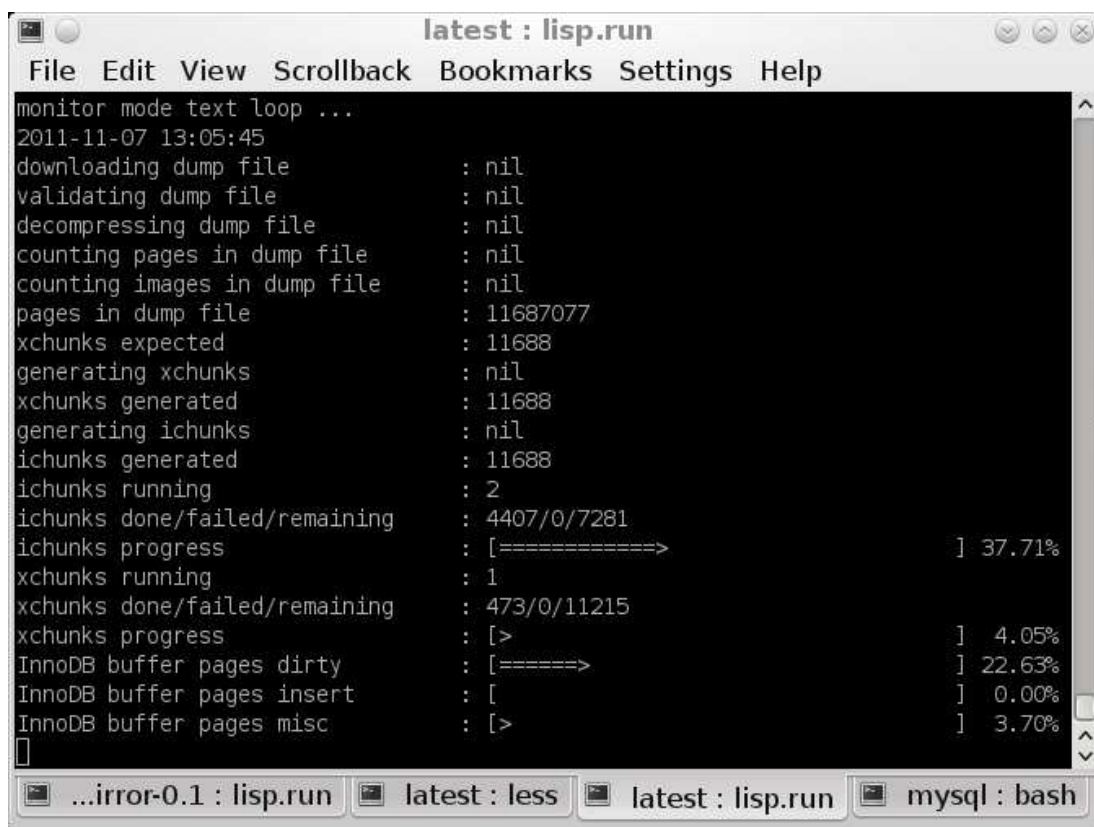


Figure 4.3: WP-MIRROR in Monitor Mode with Text Display



The screenshot shows a window titled "latest : lisp.run" with a menu bar containing "File", "Edit", "View", "Scrollback", "Bookmarks", "Settings", and "Help". The main text area displays the following output:

```
monitor mode text loop ...
2011-11-07 13:05:45
downloading dump file      : nil
validating dump file      : nil
decompressing dump file   : nil
counting pages in dump file : nil
counting images in dump file : nil
pages in dump file        : 11687077
xchunks expected         : 11688
generating xchunks        : nil
xchunks generated        : 11688
generating ichunks        : nil
ichunks generated        : 11688
ichunks running          : 2
ichunks done/failed/remaining : 4407/0/7281
ichunks progress          : [=====>] 37.71%
xchunks running          : 1
xchunks done/failed/remaining : 473/0/11215
xchunks progress          : [>] 4.05%
InnoDB buffer pages dirty : [=====>] 22.63%
InnoDB buffer pages insert : [ ] 0.00%
InnoDB buffer pages misc  : [>] 3.70%
```

The window has a scrollbar on the right side. At the bottom, there is a taskbar with four tabs: "...irror-0.1 : lisp.run", "latest : less", "latest : lisp.run", and "mysql : bash".

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 Mode 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

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

5.1.2 Initializing

```
root-shell# wp-mirror --mirror
...
-----initializing-begin-----
[ ok ]clear-pidfile
[ ok ]set-pidfile
[info]set mode of operation to: FIRST-MIRROR
[ ok ]assert-clisp-features-p
[ ok ]assert-utilities-p
[ ok ]assert-images-directory-or-create-p
[ ok ]assert-images-bad-directory-or-create-p
[ ok ]assert-images-math-directory-or-create-p
[ ok ]assert-images-thumb-directory-or-create-p
[ ok ]assert-images-tmp-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
[ 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: DELETE
[info]set mode of operation to: DROP
[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
```

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, release-all-semaphores	<code>:first-mirror</code>
2	N	<code>--mirror</code>	clear-pid, set-pidfile, release-all-semaphores	<code>:first-mirror</code>
3	N	<code>--monitor</code>	none	<code>:monitor</code>
4	Y	none	none	<code>:monitor</code>
5	Y	<code>--mirror</code>	none	<code>:next-mirror</code>
6	Y	<code>--monitor</code>	none	<code>:monitor</code>
7	N/A	<code>--delete</code>	none	<code>:delete</code>
8	N/A	<code>--drop</code>	none	<code>:drop</code>
9	N/A	<code>--restore-default</code>	none	<code>:restore-default</code>

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 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.

5.1.2.5 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.

5.1.2.6 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).

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

```
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 rather than delete them.

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).

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

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

WP-MIRROR creates a directory tree in which the [MediaWiki](#) extension `Math.php` stores the image files it creates, when it converts math equations from [\$\text{\LaTeX}\$](#) format into `PNG` image files. This directory is `/var/lib/mediawiki/images/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/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 [\$\text{\LaTeX}\$](#) format into `PNG` image files. This directory is `/var/lib/mediawiki/images/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`, `dump`, `xml`, `xchunk`, `ichunk`, 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-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.

5.1.2.14 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
```

5.1.2.15 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.

5.1.2.16 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
```

5.1.2.17 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/wp-mirror/default.conf
[info] restoring default      : /etc/wp-mirror/local.conf
[info] restoring default      : /etc/mysql/conf.d/wp-mirror.cnf
[info] restoring default      : /usr/share/mediawiki/maintenance/database_farm.sql
[info] restoring default      : /etc/mediawiki/LocalSettings.php
[info] restoring default      : /etc/mediawiki/LocalSettings_wpmirror.php
[ 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
...
```

5.1.2.18 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-languages* '("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.

5.1.2.19 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 Hardware

```
...
-----asserting-prerequisite-hardware-begin-----
[ ok ]count-cpu
[ ok ]assert-disk-space-if-top-ten-p
[ ok ]assert-physical-memory-p
[ ok ]assert-partition-free-images
[ ok ]warn-if-disk-space-low-p
[ ok ]assert-internet-access-to-wikimedia-site-p
...
```

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

5.1.3.1 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.

5.1.3.2 assert-disk-space-if-top-ten-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-disk-space-if-top-ten-p
...
```

WP-MIRROR initially (and periodically) determines the amount of free disk space available for storing images and the `InnoDB table space`.

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`, `dump`, etc.).

5.1.3.3 assert-physical-memory-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-physical-memory-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 ≥ 4 G 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](#), [dump](#), etc.).

5.1.3.4 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.

5.1.3.5 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 < 50 G of free disk space, in which case, the message looks like:

```
[...]warn-if-disk-space-low-p
[info]disk space below threshold ( 17G < 50G)
[warn]warn-if-disk-space-low-p
```

The value 50G was chosen because, WP-MIRROR in its default configuration, mirrors the [simple](#) wikipedia, which occupies 40G.

5.1.3.6 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.

5.1.4 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-connect-with-credentials-wikiuser-p
[ ok ] assert-dbms-grant-for-wikiadmin-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-hdd-write-cache-disabled-p
[ ok ] assert-database-template-and-wikidb-p
[ ok ] assert-database-xxwiki-or-load
[ ok ] assert-database-wpmirror-or-create-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-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
-----mirror-mode-begin-----
...

```

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

5.1.4.1 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`.

5.1.4.2 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';
```

5.1.4.3 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.

5.1.4.4 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.

5.1.4.5 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.

5.1.4.6 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).

5.1.4.7 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.

5.1.4.8 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.

5.1.4.9 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 a host outside localhost.

5.1.4.10 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.

5.1.4.11 assert-hdd-write-cache-disabled-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-hdd-write-cache-disabled-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 HDD’s `write cache`, where it may be lost during system failure. It is therefore important that the disk’s `write cache` be disabled.

WP-MIRROR first identifies the disk(s) that hold the `InnoDB table space`. Next WP-MIRROR attempts to disable the write cache for each such disk. Finally, WP-MIRROR asserts that the `write cache(s)` have been successfully disabled.

5.1.4.12 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 wikipeias 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.

5.1.4.13 assert-database-xxwiki-or-load

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-database-xxwiki-or-load
...
```

WP-MIRROR must be able to mirror wikipeias 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 [xxwiki](#), 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.

5.1.4.14 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.

5.1.4.15 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.

5.1.4.16 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.

5.1.4.17 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.

5.1.4.18 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
$wgEnableUploads      = true;
$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.

5.1.4.19 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;
```

5.1.4.20 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.

5.1.4.21 assert-concurrency-limit-xchunk-p

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-concurrency-limit-xchunk-p
...
```

`InnoDB` has two `table space` formats (as of 2012)

- `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.

5.1.4.22 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 a virtual host `mediawiki.site`. The virtual host container can be found in `/etc/apache2/sites-enabled/mediawiki.site.conf`, and looks like:

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

5.1.4.23 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.mediawiki.site`, where `simple` might be replaced with any other language code. For each missing virtual host name, WP-MIRROR appends to `/etc/hosts` a line like:

```
::1 simple.mediawiki.site
```

5.1.4.24 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.

5.1.5 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, scraped for information, 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 readily

grasps data in tabular format, and whereas the human mind struggles with large blocks of code, an experienced engineer looks for ways to replace the later with the former.

For WP-MIRROR 0.3 and later versions, to achieve clarity: 1) Every file is assigned a:

- **type** (one of `checksum`, `dump`, `xml`, `xchunk`, `ichunk`, and `images`), 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

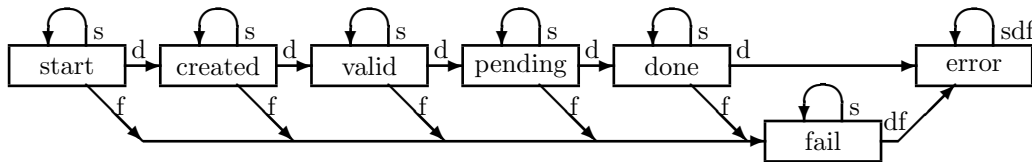


Table 5.2: FSM Tables ***state-transition*** and ***type-state-function***

Event	State				
	start	created	valid	pending	done
start	start	created	valid	pending	done
done	created	valid	pending	done	done
fail	fail	fail	fail	fail	error

File type	State				
	start	created	valid	pending	done
checksum	fsm-file-download	fsm-file-md5sum	fsm-file-parse	fsm-no-op	fsm-no-op
dump	fsm-file-download	fsm-file-validate	fsm-file-decompress	fsm-no-op	fsm-no-op
xml	fsm-file-count	fsm-file-md5sum	fsm-file-split	fsm-file-remove	fsm-no-op
xchunk	fsm-file-count	fms-file-md5sum	fsm-file-wikix	fsm-file-import	fsm-no-op
ichunk	fsm-file-count	fsm-file-md5sum	fsm-file-shell	fsm-file-remove	fsm-no-op
images	fsm-images-count	fsm-images-validate	fsm-images-rebuild	fsm-images-chown	fsm-no-op

Table 5.3: FSM Priority Table **type-state-priority**

Priority	Type	State	Concurrent	Non-image	Function
highest	checksum	pending	t	t	fsm-no-op
	checksum	valid	t	t	fsm-file-parse
	checksum	created	t	t	fsm-file-md5sum
	checksum	start	nil	t	fsm-file-download
	dump	pending	t	t	fsm-no-op
	dump	valid	t	t	fsm-file-decompress
	dump	created	t	t	fsm-file-validate
	dump	start	nil	t	fsm-file-download
	xml	pending	t	t	fsm-file-remove
	xml	valid	t	t	fsm-file-split
	xml	created	t	t	fsm-file-md5sum
	xml	start	t	t	fsm-file-count
	xchunk	valid	t	t	fsm-file-wikix
	xchunk	created	t	t	fsm-file-md5sum
	xchunk	start	t	t	fsm-file-count
	ichunk	pending	t	t	fsm-file-remove
	ichunk	valid	nil	nil	fsm-file-shell
	ichunk	created	t	nil	fsm-file-md5sum
	ichunk	start	t	nil	fsm-file-count
	images	start	t	nil	fsm-images-count
	images	created	t	nil	fsm-images-validate
	images	valid	nil	nil	fsm-images-rebuild
	xchunk	pending	t	t	fsm-file-import
lowest	images	pending	t	nil	fsm-images-chown

5.1.6 The Finite State Machine—Step by Step

```

root-shell# wp-mirror --mirror
...
-----mirror-mode-begin-----
[ ok ] fsm-boot
[ ok ] fsm-file-download simplewiki-latest-md5sums.txt
[ ok ] fsm-file-md5sum simplewiki-latest-md5sums.txt
[ ok ] fsm-file-parse simplewiki-latest-md5sums.txt
[ ok ] fsm-no-op simplewiki-latest-md5sums.txt
...

```

The FSM works as follows:

Step 0: Initialize 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 checksum file (the default is `simplewiki-latest-md5sums.txt`), its `type` (`checksum`), its `state` (`start`), and a semaphore (set to '1').

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

```

+-----+-----+-----+-----+
| name                | type    | state | semaphore |
+-----+-----+-----+-----+
| simplewiki-latest-md5sums.txt | checksum | start |          1 |
+-----+-----+-----+-----+

```

Step 1: Prioritize rows. The FSM orders these rows according to the **type-state-priority** table.

Caveat: If a user wants no images, by editing `/etc/wp-mirror/local.conf` to contain:

```
(defparameter *mirror-image-download-p* nil)
```

then the FSM ignores rows for which the `non-image` value in `*type-state-priority*` is 'nil'.

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-latest-md5sums.txt	checksum	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-file-download`.

Step 4: Apply function. The FSM applies the function to the file. In this case, `fsm-file-download` downloads the `checksum` file from the Wikimedia Foundation:

<http://dumps.wikimedia.org/simplewiki/latest/simplewiki-latest-md5sums.txt>

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`:

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-latest-md5sums.txt	checksum	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-latest-md5sums.txt	checksum	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-file-parse`, `fsm-file-decompress`, `fsm-file-split`, and `fsm-file-wikix`. 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 `xchunks`,
- 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.

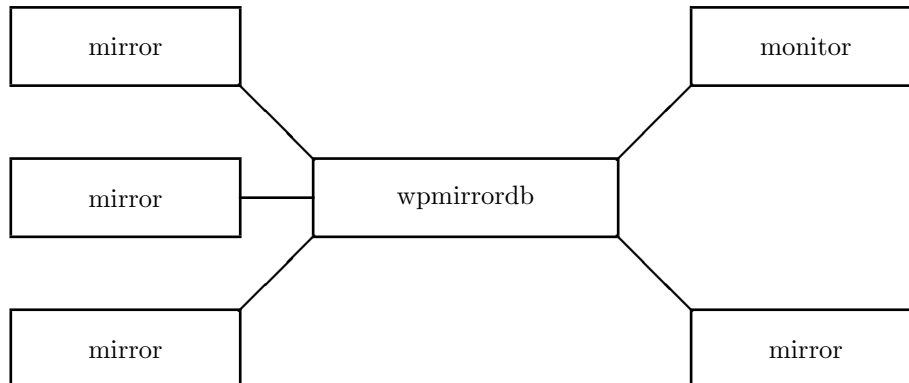
Here is how we enforce isolation:

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](#).

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:

```
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,
  'language'  varchar(20) NOT NULL DEFAULT 'error',      -- usually 'simplewiki'
  '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('checksum','dump','xml','xchunk','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',      -- [0-255]
  '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
FROM 'wpmirror'.'file'
(format nil "WHERE type=~a" " file-type)
(format nil "AND state=~a" " file-state)
AND semaphore=1                                    -- semaphore is free
ORDER BY name,state
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 ] fsm-file-download simplewiki-latest-md5sums.txt
[ ok ] fsm-file-md5sum simplewiki-latest-md5sums.txt
[ ok ] fsm-file-parse simplewiki-latest-md5sums.txt
[ ok ] fsm-no-op simplewiki-latest-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 a checksum file (the default is `simplewiki-latest-md5sums.txt`), its `type` (`checksum`), 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-latest-md5sums.txt       | checksum | start |          1 |
+-----+-----+-----+-----+
```

5.1.9.3 `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-20121106-pages-articles.xml | xml     | start |          1 |
| simplewiki-20121106-pages-articles.xml.bz2 | dump    | pending |          1 |
| simplewiki-latest-md5sums.txt           | checksum | done  |          1 |
+-----+-----+-----+-----+
```


FSM function: `fsm-file-count` scans the decompressed dump 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-20121106-pages-articles.xml	xml	created	1
simplewiki-20121106-pages-articles.xml.bz2	dump	pending	1
simplewiki-latest-md5sums.txt	checksum	done	1

5.1.9.4 fsm-file-decompress

Purpose: To decompress a `dump` file using `bunzip2` and thereby generate an `xml` file.

FSM before: The first row below will be processed with the `fsm-file-decompress` function.

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

name	type	state	semaphore
simplewiki-20121106-pages-articles.xml.bz2	dump	valid	1
simplewiki-latest-md5sums.txt	checksum	done	1

FSM function: The `fsm-file-decompress` function decompresses the `dump` 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-20121106-pages-articles.xml	xml	start	1
simplewiki-20121106-pages-articles.xml.bz2	dump	pending	1
simplewiki-latest-md5sums.txt	checksum	done	1

5.1.9.5 fsm-file-download

Purpose: Used to download both `checksum` files and `dump` files.

FSM before: Here is the `wpmirror.file` database table, after `fsm-boot` inserts the first entry.

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

name	type	state	semaphore
simplewiki-latest-md5sums.txt	checksum	start	1

FSM function: `fsm-file-download` invokes `cURL` to download the `checksum` file.

FSM after:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

name	type	state	semaphore
simplewiki-latest-md5sums.txt	checksum	created	1

5.1.9.6 fsm-file-import

Purpose: To have [MediaWiki](#) import the articles into the [xxwiki](#) database.

Motivation: The dumps 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.

Therefore, for robustness and simplicity, both for mirroring and monitoring, WP-MIRROR splits dump 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: The second row below will be processed by [fsm-file-import](#)

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

name	type	state	semaphore
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
...	xchunk	pending	1
simplewiki-20121106-pages-articles.xml	xml	done	1
simplewiki-20121106-pages-articles.xml.bz2	dump	done	1
simplewiki-latest-md5sums.txt	checksum	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-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
...	xchunk	pending	1
simplewiki-20121106-pages-articles.xml	xml	done	1
simplewiki-20121106-pages-articles.xml.bz2	dump	done	1
simplewiki-latest-md5sums.txt	checksum	done	1

5.1.9.7 fsm-file-md5sum

Purpose: Compute the `md5sum` of a file.

FSM before: The following row will be processed with the `fsm-file-md5sum` function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----+-----+-----+-----+
| name                | type    | state  | semaphore |
+-----+-----+-----+-----+
| simplewiki-latest-md5sums.txt | checksum | created |          1 |
+-----+-----+-----+-----+
```

FSM function: The `fsm-file-md5sum` function simply computes the `md5sum` of the file. This 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-latest-md5sums.txt | checksum | valid  |          1 |
+-----+-----+-----+-----+
```

5.1.9.8 fsm-file-parse

Purpose: Scrape a `checksum` file for the name of a `dump` file.

FSM before: The following row will be processed with the `fsm-file-parse` function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
+-----+-----+-----+-----+
| name                | type    | state  | semaphore |
+-----+-----+-----+-----+
| simplewiki-latest-md5sums.txt | checksum | valid  |          1 |
+-----+-----+-----+-----+
```

FSM function: The `fsm-file-parse` function reads the `checksum` file and looks up the name of the `dump` 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-20121106-pages-articles.xml.bz2 | dump    | start  |          1 |
| simplewiki-latest-md5sums.txt              | checksum | pending |          1 |
+-----+-----+-----+-----+
```

5.1.9.9 fsm-file-remove

Purpose: Delete a file from the file system.

Motivation: Some files are deleted to save disk space.

Type	State	Comment
<code>checksum</code>		Kept to avoid re-download until newer file posted
<code>dump</code>		Kept to avoid re-download during testing
<code>xml</code>	<code>pending</code>	Deleted to save disk space
<code>xchunk</code>		Not implemented
<code>ichunk</code>	<code>pending</code>	Deleted to save disk space

FSM before: The row with the `xml` file will be processed with the `fsm-file-remove` function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

name	type	state	semaphore
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	dump	done	1
simplewiki-latest-md5sums.txt	checksum	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;
```

name	type	state	semaphore
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	done	1
simplewiki-20121106-pages-articles.xml.bz2	dump	done	1
simplewiki-latest-md5sums.txt	checksum	done	1

5.1.9.10 fsm-file-shell

Purpose: Execute `ichunks` to download image files from the Wikimedia Foundation.

Motivation: The dumps for the largest wikipedias contain 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: The first row will be processed with the `fsm-file-shell` function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

name	type	state	semaphore
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
...	xchunk	pending	1
simplewiki-20121106-pages-articles.xml	xml	done	1
simplewiki-20121106-pages-articles.xml.bz2	dump	done	1
simplewiki-latest-md5sums.txt	checksum	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-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
...	xchunk	pending	1
simplewiki-20121106-pages-articles.xml	xml	done	1
simplewiki-20121106-pages-articles.xml.bz2	dump	done	1
simplewiki-latest-md5sums.txt	checksum	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 dump 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
```

5.1.9.11 fsm-file-split

Purpose: Divide a large [dump](#) file into smaller [xchunk](#) files (of 1000 pages each).

Motivation: The dumps for the largest wikipeidias 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.

Therefore, for robustness and simplicity, both for mirroring and monitoring, WP-MIRROR splits dump 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: The first row below will be processed with the [fsm-file-split](#) function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

name	type	state	semaphore
simplewiki-20121106-pages-articles.xml	xml	valid	1
simplewiki-20121106-pages-articles.xml.bz2	dump	done	1
simplewiki-latest-md5sums.txt	checksum	done	1

FSM function: The `fsm-file-split` 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-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	dump	done	1
simplewiki-latest-md5sums.txt	checksum	done	1

When mirroring the `en`, the `fsm-file-split` function will insert over ten thousand rows, which the FSM will process *seriatim*.

5.1.9.12 fsm-file-validate

Purpose: Compute the `md5sum` of the `dump` file and compare it to the `md5sum` listed in the `checksum` file.

Motivation: The largest `dump` files are several gigabytes. It is important to verify the integrity of the download process.

FSM before: The first row will be processed with the `fsm-file-validate` function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

name	type	state	semaphore
simplewiki-20121106-pages-articles.xml.bz2	dump	created	1
simplewiki-latest-md5sums.txt	checksum	done	1

FSM function: The `fsm-file-validate` function computes the `md5sum` of the `dump` 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-20121106-pages-articles.xml.bz2	dump	valid	1
simplewiki-latest-md5sums.txt	checksum	done	1

5.1.9.13 fsm-file-wikix

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-wikix` function:

```
mysql> SELECT name,type,state,semaphore FROM wpmirror.file;
```

name	type	state	semaphore
simplewiki-20121106-pages-articles-p000000000-c000001000.xml	xchunk	valid	1
simplewiki-20121106-pages-articles-p000001000-c000001000.xml	xchunk	start	1
...	xchunk	start	1
simplewiki-20121106-pages-articles.xml	xml	done	1
simplewiki-20121106-pages-articles.xml.bz2	dump	done	1
simplewiki-latest-md5sums.txt	checksum	done	1

FSM function: The `fsm-file-wikix` function reads the `xchunk` file and scrapes out the names of any image files. `fsm-file-wikix` 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-20121106-pages-articles-p000000000-c000001000.sh	ichunk	start	1
simplewiki-20121106-pages-articles-p000000000-c000001000.xml	xchunk	pending	1
simplewiki-20121106-pages-articles-p000001000-c000001000.xml	xchunk	start	1
...	xchunk	start	1
simplewiki-20121106-pages-articles.xml	xml	done	1
simplewiki-20121106-pages-articles.xml.bz2	dump	done	1
simplewiki-latest-md5sums.txt	checksum	done	1

5.1.9.14 `fsm-images-chown`

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.

5.1.9.15 `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.

5.1.9.16 `fsm-images-rebuild`

Purpose: To have MediaWiki rebuild the `xxwiki.image` database table (where `xx` stands for a language code).

Motivation: When images are downloaded, the `xxwiki.image` database table will not be populated automatically. A maintenance script,

```
root-shell# /usr/bin/php /usr/share/mediawiki/maintenance/rebuildImages_farm.php \
simple.mediawiki.site
```

must be run for that purpose.

Design note. There are two choices about when to apply `fsm-images-rebuild`:

- If the `xxwiki.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).
-

5.1.9.17 `fsm-images-validate`

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/`.

5.1.9.18 `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`.

5.1.10 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`.

5.2 How Monitor Mode 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 ([checksum](#), [dump](#), [xml](#), [xchunk](#), [ichunk](#), etc.). 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/wp-mirror/default.conf
[info]restoring default      : /etc/wp-mirror/local.conf
[info]restoring default      : /etc/mysql/conf.d/wp-mirror.cnf
[info]restoring default      : /usr/share/mediawiki/maintenance/database_farm.sql
[info]restoring default      : /etc/mediawiki/LocalSettings.php
[info]restoring default      : /etc/mediawiki/LocalSettings_wpmirror.php
[ 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-languages* '("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 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.

5.2.4 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-xxwiki-or-load
[ 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:

5.2.4.1 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`.

5.2.4.2 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';
```

5.2.4.3 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.

5.2.4.4 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.

5.2.4.5 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.

5.2.4.6 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).

5.2.4.7 assert-database-xxwiki-or-load

```
root-shell# wp-mirror --mirror
...
[ ok ]assert-database-xxwiki-or-load
...
```

WP-MIRROR must be able to mirror wikipeias 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 `xxwiki`, 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.

5.2.4.8 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.

5.2.4.9 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.

5.2.5 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.

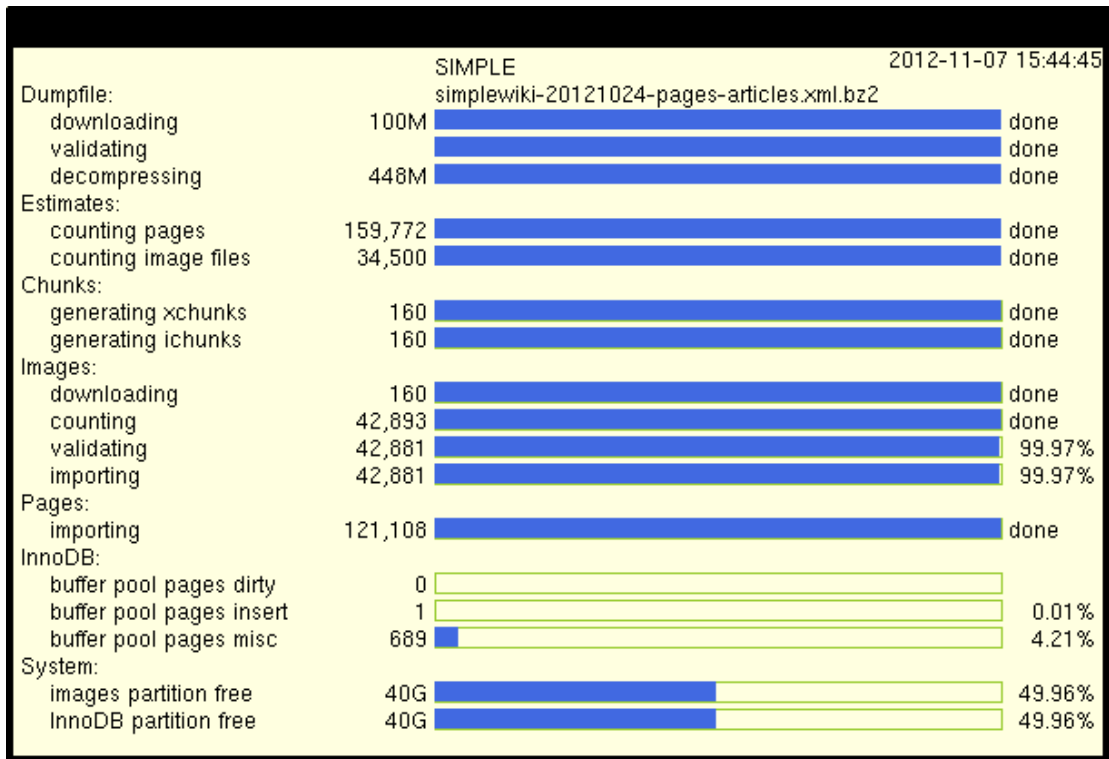
5.2.6 Display Progress Bars

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](#).

5.3 How `--delete` Works

```
root-shell# wp-mirror --delete xh
[ ok ]process-command-line-arguments-or-die
-----initializing-begin-----
[info]set mode of operation to: DELETE
[ 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
-----asserting-prerequisite-hardware-begin-----
-----asserting-prerequisite-software-begin-----
-----delete-mode-begin-----
[ ok ]delete-working-files-for-language-code xh
[ ok ]delete-state-for-language-code xh
-----finalizing-begin-----
[ ok ]log-stop
```

Figure 5.3: WP-MIRROR Monitor Mode Progress Bars



5.4 How `--drop` Works

```

root-shell# wp-mirror --drop xh
[ ok ]process-command-line-arguments-or-die
-----initializing-begin-----
[info]set mode of operation to: DROP
[ 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
-----asserting-prerequisite-hardware-begin----
-----asserting-prerequisite-software-begin----
-----drop-mode-begin-----
[ ok ]drop-database-for-language-code-p xh          <---
[ ok ]delete-working-files-for-language-code xh
[ ok ]delete-state-for-language-code xh
-----finalizing-begin-----
[ ok ]log-stop

```


5.5 How `--restore-default` Works

```

root-shell# wp-mirror --restore-default
[ ok ]process-command-line-arguments-or-die
-----initializing-begin-----
[info]set mode of operation to: RESTORE-DEFAULT
[ 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
-----asserting-prerequisite-hardware-begin----
-----asserting-prerequisite-software-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-images-bad
[ ok ]delete-directory-images-math
[ ok ]delete-directory-images-thumb
[ ok ]delete-directory-working
[info]deleting config file : /etc/wp-mirror/default.conf
[info]deleting config file : /etc/wp-mirror/local.conf
[info]deleting config file : /etc/mysql/conf.d/wp-mirror.cnf
[info]deleting config file : /usr/share/mediawiki/maintenance/database_farm.sql
[info]deleting config file : /etc/mediawiki/LocalSettings.php
[info]deleting config file : /etc/mediawiki/LocalSettings_wpmirror.php
[ 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
```

5.6 How Scheduled Jobs Work

5.6.1 `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.

5.6.2 `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 a 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

Design Notes

A.1 Concurrency Limit

WP-MIRROR imposes limits on concurrency:

- If the `InnoDB table space` uses the `Antelope` format, then concurrency is limited to the lesser of the number of CPUs and three; and
- If the `InnoDB table space` uses the `Barracuda` format, then concurrency is limited one.

Yet, as has been noted, `apt-mirror` forks 20 sub-processes (default), each of which run `wget`, to build a mirror of the Debian archives.

A.1.1 Why not Fork 20 `ichunk` Sub-processes?

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 `ichunks` one or two at a time works acceptably well.

A.1.2 Why not Fork 20 `xchunk` Sub-processes?

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.

A.1.3 Why not Process `ichunks` in Parallel with `xchunks`?

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.

Experiments also show that, when care is exercised to ensure that corresponding chunks are not processed concurrently, then significant performance gains are achieved.

A.1.4 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`, `dump`, `dump-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.

A.2 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`.

A.2.1 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\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>...\lt;/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  $y = mx + b$  where 'b' is the [[y-intercept]] of the graph
and 'm' is the [[slope]]. This formula applies to the coordinates of the
graph or  $(x, y)$ .
```

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)
```

Finally, when the page is rendered by a browser, the equation is replaced by the [PNG](#) image file. That is the math tags:

```
<math>y = mx + b</math>
```

are replaced with an image tag:

```

```

A.2.2 LaTeX

[L^AT_EX](#) is a macro package built on top of [T_EX](#), which is a typesetting system. [L^AT_EX](#) does a professional job of typesetting mathematical equations, chemical formulae, etc. The American Mathematical Society uses [L^AT_EX](#) for its publications.

A.2.3 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 [T_EX](#) 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/`.

A.2.3.1 Update (2012)

The code that converts math equations from [T_EX](#) 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 `xxwiki.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.mediawiki.site
```

WP-MIRROR 0.4 automates the configuration required by the above two caveats.

A.2.4 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.mediawiki.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.mediawiki.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.mediawiki.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)
...
```

A.3 Priority

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.

Appendix B

Configuration File Parameter Reference

Table B.1: WP-MIRROR Configuration File Parameter Reference

Name	Default	Range	Intr	Rem
<code>*ichunk-curl-retry*</code>	0	0, 1	0.1	
<code>*mirror-image-download-p*</code>	t	nil, t	0.1	
<code>*mirror-image-use-wikix-p*</code>	nil	nil, t	0.1	
<code>*mirror-image-validate-p*</code>	t	nil, t	0.1	
<code>*mirror-importdump-timeout-sec-max*</code>	nil	nil, 500-5000	0.4	
<code>*mirror-languages*</code>	'("simple")		0.2	
<code>*mirror-objectcache-delete-limit*</code>	1000		0.1	
<code>*mirror-objectcache-threshold*</code>	10000		0.1	
<code>*mirror-rebuildimages-threshold*</code>	20000		0.2	
<code>*mirror-rebuildimages-timeout-sec-inc*</code>	200		0.2	
<code>*mirror-rebuildimages-timeout-sec-min*</code>	2000		0.2	
<code>*mirror-rebuildimages-timeout-sec-max*</code>	20000		0.2	
<code>*mirrors*</code>	1	1-3	0.1	
<code>*monitor-gui-font-preference-list*</code>	'("-*-helvetica-medium-r-*-*12-*-*-*-*0.1" "-*-lucida-medium-r-*-*12-*-*-*-*" "-*-lucida-medium-r-*-*14-*-*-*-*")		0.1	
<code>*monitor-gui-one-window-per-language*</code>	t	nil, t	0.1	
<code>*monitor-mode*</code>	:auto	:auto, :gui, :screen, :text	0.1	
<code>*monitor-poll-sec*</code>	10		0.1	
<code>*monitor-poll-sec-min*</code>	10		0.1	
<code>*monitor-poll-sec-max*</code>	1000		0.1	

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
system-cpu	1	1-3	0.1	
system-cpu-min	1		0.4	
system-cpu-max	3		0.4	
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-configuration	/etc/wp-mirror/		0.1	
whereis-directory-images	/var/lib/mediawiki/images/		0.1	
whereis-directory-images-bad	/var/lib/mediawiki/images/bad-images/		0.1	
whereis-directory-images-math	/var/lib/mediawiki/images/math/		0.4	
whereis-directory-images-thumb	/var/lib/mediawiki/images/thumb/		0.4	
whereis-directory-mediawiki-config	/etc/mediawiki/		0.4	
whereis-directory-mediawiki-extensions-math	/var/lib/mediawiki/extensions/Math/math/		0.4	
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	

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
whereis-directory-tmp	/tmp/		0.2	
whereis-directory-working	/var/lib/mediawiki/images/ wp-mirror/		0.1	
whereis-directory-wpmirror-config	/etc/wp-mirror/		0.4	
whereis-env	/usr/bin/env		0.4	
whereis-file-checksums-template	xxwiki-latest-md5sums.txt		0.2	
whereis-file-config-default	default.conf		0.1	
whereis-file-config-local	local.conf		0.1	
whereis-file-dump-template	xxwiki-yyyyymmdd-pages-articles.xml.bz2		0.2	
whereis-file-dev-null	/dev/null		0.2	
whereis-file-etc-hosts	/etc/hosts		0.2	
whereis-file-log	/var/log/wp-mirror.log		0.1	
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-importdump	importDump.php		0.2	
whereis-file-mediawiki-rebuildimages	rebuildImages.php		0.2	
whereis-file-mediawiki-update	update.php		0.2	
whereis-file-mediawiki-farm-database	database_farm.sql		0.4	
whereis-file-mediawiki-farm-importdump	importDump_farm.php		0.2	
whereis-file-mediawiki-farm-rebuildimages	rebuildImages_farm.php		0.2	
whereis-file-mediawiki-farm-update	update_farm.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-pid	/var/run/wp-mirror.pid		0.4	
whereis-file-tmp-http	/tmp/http		0.4	
whereis-file-virtual-host	mediawiki.site.conf		0.2	

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
whereis-file-wikidb-template	template.sql		0.2	
whereis-file-wpmirror-config-default	default.conf		0.4	
whereis-file-wpmirror-config-local	local.conf		0.4	
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/lis		0.1	
whereis-md5sum	/usr/bin/md5sum		0.1	
whereis-mv	/bin/mv		0.1	
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-rm	/bin/rm		0.1	
whereis-rsvg	/usr/bin/rsvg		0.1	
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-wikix	/usr/bin/wikix		0.1	
whereis-x	/usr/bin/X		0.1	
whereis-zoneinfo	/usr/share/zoneinfo		0.4	
wikimedia-checksums-template	xxwiki/latest/xxwiki-latest-md5sums.txt		0.2	
wikimedia-dump-template	xxwiki/yyyymmdd/xxwiki-yyyyymmdd-pages-articles.xml.bz2		0.2	
wikimedia-large-wiki-list	'("en" "de" "fa" "nl" "it" "es" "pl" "ru" "ja" "pt")		0.1	
wikimedia-site	http://dumps.wikimedia.org/		0.1	
wikimedia-wikix	http://wikix.ngen-cast.de/wikix.tar.gz		0.1	
wpmirror-directory-restore	/usr/share/doc/wp-mirror/restore/		0.4	
wpmirror-config-delete-list			0.4	
wpmirror-config-restore-list			0.4	
xchunk-page-count	1000		0.1	
virtual-host-name	mediawiki.site		0.4	

Table B.2: WP-MIRROR Configuration File Parameter Reference
(Obsolete)

Name	Default	Range	Intr	Rem
<code>*mirror-chunk-diff-threshold*</code>	20		0.1	0.2
<code>*mirror-image-rebuild-threshold*</code>	20000		0.1	0.2
<code>*mirror-image-then-page-p*</code>	t	nil, t	0.1	0.2
<code>*system-disk-space-min*</code>	(<code>* 250 *gigabyte*</code>)		0.1	0.4
<code>*whereis-directory-mediawiki*</code>	/etc/mediawiki/		0.2	0.4
<code>*whereis-directory-mediawiki-extensions*</code>	/etc/mediawiki-extensions/		0.4	0.4
<code>*whereis-directory-mediawiki-extensions-available*</code>	/etc/mediawiki-extensions/extension-extensions-available/		0.4	0.4
<code>*whereis-directory-mediawiki-extensions-enabled*</code>	/etc/mediawiki-extensions/extension-extensions-enabled/		0.4	0.4
<code>*whereis-directory-mediawiki-includes*</code>	/usr/share/mediawiki/includes/		0.2	0.4
<code>*whereis-directory-mysql*</code>	/var/lib/mysql/		0.1	0.4
<code>*whereis-directory-mysql*</code>	/etc/mysql/		0.4	0.4
<code>*whereis-echo*</code>	/bin/echo/		0.2	0.4
<code>*whereis-file-mediawiki-import*</code>	Import.php		0.2	0.4
<code>*whereis-mediawiki-adminsettings*</code>	/etc/mediawiki/AdminSettings.php		0.1	0.2
<code>*whereis-mediawiki-localsettings*</code>	/etc/mediawiki/LocalSettings.php		0.1	0.2
<code>*whereis-file-mediawiki-adminsettings*</code>	AdminSettings.php		0.2	0.4
<code>*whereis-file-mediawiki-localsettings*</code>	LocalSettings.php		0.2	0.4
<code>*whereis-file-mediawiki-localsettings-wpmirror*</code>	LocalSettings_wpmirror.php		0.2	0.4
<code>*whereis-file-mysql-custom*</code>	/etc/mysql/conf.d/custom.cnf		0.4	0.4
<code>*whereis-file-mysql-debian*</code>	/etc/mysql/debian.cnf		0.4	0.4
<code>*whereis-mediawiki-importdump*</code>	/usr/share/mediawiki/maintenance/importDump.php		0.1	0.2
<code>*whereis-mediawiki-rebuildimages*</code>	/usr/share/mediawiki/maintenance/rebuildImages.php		0.1	0.2
<code>*whereis-mediawiki-update*</code>	/usr/share/mediawiki/maintenance/update.php		0.1	0.2
<code>*whereis-mysql-custom*</code>	/etc/mysql/conf.d/custom.cnf		0.1	0.4
<code>*whereis-php5-suhosin*</code>	/etc/php5/conf.d/suhosin.ini		0.1	0.4

continued on next page

continued from previous page

Name	Default	Range	Intr	Rem
whereis-pidfile	/var/run/wp-mirror.pid		0.1	0.4
wikimedia-checksums	xxwiki/latest/xxwiki-latest- md5sums.txt		0.1	0.2
wikimedia-dump-pattern	pages-articles.xml.bz2		0.1	0.2
wikimedia-import	http://svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/includes/Import.php		0.1	0.4
wikimedia-languages-to-mirror	'("simple")		0.1	0.2

Appendix C

Error Messages (Work In Progress)

C.1 Error Codes

Table C.1: WP-MIRROR Error Codes

Code	Intr	Rem
1000 (assert-asdf)	0.3	
1001 (assert-clisp-features-p)	0.1	
1002 (assert-concurrency-limit-xchunk-p)	0.1	
1004 (assert-database-administrator-credentials-p)	0.1	
1005 (assert-database-administrator-account-p)	0.1	
1006 (assert-database-administrator-template-or-create)	0.2	
1007 (assert-database-wikidb-p)	0.1	
1009 (assert-database-wpmirror-or-create)	0.2	
1010 (assert-database-wpmirror-table-p)	0.2	
1011 (assert-database-xxwiki-or-create)	0.2	
1012 (assert-dbms-mysql-p)	0.1	
1013 (assert-disk-space-p)	0.1	
1014 (assert-hdd-write-cache-disabled-p)	0.1	
1015 (assert-images-directory-p)	0.1	
1016 (assert-images-bad-directory-or-create)	0.1	
1017 (assert-internet-access-to-wikimedia-site-p)	0.1	
1021 (assert-mediawiki-localsettings-p)	0.1	
1022 (assert-mediawiki-localsettings-image)	0.1	
1024 (assert-mediawiki-localsettings-tidy)	0.1	
1026 (assert-physical-memory-p)	0.1	
1027 (assert-utilities-p)	0.1	
1028 (assert-virtual-host-p)	0.2	
1029 (assert-virtual-host-name-resolution-p)	0.2	
1030 (assert-working-directory-or-create)	0.1	
1031 (warn-if-detect-proxy)	0.1	

- 1000 ([assert-asdf](#))
- 1001 ([assert-clisp-features-p](#))
- 1002 ([assert-concurrency-limit-xchunk-p](#))
- 1004 ([assert-database-administrator-credentials-p](#))
- 1005 ([assert-database-administrator-account-p](#))
- 1006 ([assert-database-administrator-template-or-create](#))

- 1007 (`assert-database-wikidb-p`)
- 1009 (`assert-database-wpmirror-or-create`)
- 1010 (`assert-database-wpmirror-table-p`)
- 1011 (`assert-database-xxwiki-or-create`)
- 1012 (`assert-dbms-mysql-p`)
- 1013 (`assert-disk-space-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`)
- 1020 (`assert-mediawiki-import`)
- 1021 (`assert-mediawiki-localsettings-p`)
- 1022 (`assert-mediawiki-localsettings-image`)
- 1024 (`assert-mediawiki-localsettings-tidy`)
- 1026 (`assert-physical-memory-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`)

C.2 Error Codes (Obsolete)

Table C.2: WP-MIRROR Error Codes (Obsolete)

Code	Intr	Rem
1003 (<code>assert-curl-max-time-p</code>)	0.1	0.4
1008 (<code>assert-database-wpmirrordb-p</code>)	0.1	0.2
1008 (<code>assert-database-wpmirror-p</code>)	0.2	0.3
1009 (<code>assert-database-wpmirrordb-or-create</code>)	0.1	0.2
1010 (<code>assert-database-wpmirrordb-table-p</code>)	0.1	0.2
1018 (<code>assert-mediawiki-adminsettings-p</code>)	0.1	0.4
1019 (<code>assert-mediawiki-extensions</code>)	0.1	0.4
1020 (<code>assert-mediawiki-import</code>)	0.2	0.4
1023 (<code>assert-mediawiki-localsettings-tex</code>)	0.1	0.4
1025 (<code>assert-php5-suhosin-p</code>)	0.1	0.4

- 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.

- 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`.
- 1009 (`assert-database-wpmirrordb-or-create`)
Beginning with WP-MIRROR 0.2, this function was renamed `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`.
- 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`.
- 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.
- 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).

C.3 Full Error Messages

```
+-----+
| 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:	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:      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:	<p>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.</p> <p>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.</p> <p>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.</p> <ul style="list-style-type: none"> 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) <p>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.</p> <p>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.</p>
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:      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:  A 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:      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 credentials (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 better 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:      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:	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:      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 mediawiki.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.mediawiki.site/. |
|               The virtual host container should be found in |
|               /etc/apache2/sites-available/mediawiki.site.conf. |
|               There should also be a symbol link to it found in |
|               /etc/apache2/sites-enabled/mediawiki.site.conf. |
|               And the virtual host configuration should be loaded into |
|               Apache2. |
|               |
|               root-shell# apache2ctl -S 2>&1 | grep mediawiki |
|               port 80 namevhost mediawiki.site (/etc/apache2/sites-e |
|               nabled/mediawiki.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 mediawiki.site not found in /etc/hosts. |
| DESCRIPTION: Apache2 needs to resolve the name mediawiki.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 localhost mediawiki.site www.mediawiki.site |
|               ::1 meta.mediawiki.site simple.mediawiki.site |
|               ::1 en.mediawiki.site zh.mediawiki.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. |
+-----+

```

C.4 Full Error Messages (Obsolete)

```

+-----+
| 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:      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:      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. |
+-----+

```

Appendix D

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.

D.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.

D.1.1 Changes in WP-MIRROR 0.4 (2012-11-12)

D.1.1.1 Functionality Added or Changed

- **Command-line option added:** Users who mess up the configuration files, now have the option of starting over. The `--restore-default` 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 `xxwiki.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.mediawiki.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 `TeX`, 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`, `dhhelp` 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 40G.
- **Menufile:** WP-MIRROR is now automatically registered with `menufile`, so that it 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.

D.1.1.2 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 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.

D.1.2 Changes in WP-MIRROR 0.3 (2012-03-04)

D.1.2.1 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 the task of coding and debugging.

D.1.2.2 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).

D.1.3 Changes in WP-MIRROR 0.2 (2011-12-25)

D.1.3.1 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>

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 E

Experiments (Autumn 2010—Spring 2011)

This is a transcript of my original notes, edited, and formatted for [A_TE_X](#).

Source: `/root/sysadmin/2010_09_sysadmin.log`.

Date: 2010-09-12 through 2011-04

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 0.2 TB. The images might also occupy an additional 1.0 TB. We do not want to mirror all its revisions, user talk, etc. as the entire en.wikipedia.org site would require over 10 TB. Likewise we do not want to mirror other languages as this could require over 100TB.

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 1TB. So to allow for future growth we should purchase a 1.5TB or 2TB 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-yyyyymmdd-		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](#) diskpace 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 diskpace occupied by [InnoDB](#) data is computed as follows:

1. [vg5](#) has two logical volumes

```
root-shell# lvsdisplay -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 $(314572800000-256567672832)/(1024*1024*1024) = 54\text{G}$

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 (2.0 TB 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 -f -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 2TB, 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 “meta-data=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 2TB 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. `sendmail`, `drupal`, 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: your-pass-phrase
Verify passphrase: your-pass-phrase
Command successful.
```


Where *your-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-tXU1-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-tXU1-t5wCqd

```

```
root-shell# lvs 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-qd0X-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 (`plasma-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 a 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 http://download.wikimedia.org/tools/mwdumper.jar
shell$ wget http://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/wiki/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=filesystems, 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-ibdata0:100000Mrw
[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 <code>buffer pool</code>
4	D2F-L250P100H-	12	53.4	2.38	1.00	
5	D2F-L250P1000H-	12	131.8	5.88	2.47	larger <code>buffer pool</code>
7	D2F+L250P50H+	60.5	134		1.00	
8	D2F+L250P1000H+	63	389		2.90	larger <code>buffer pool</code>
5	D2F-L250P1000H-	12	131.8	5.88	1.00	
6b	D2F-L250P1000H+	12	128.7	5.75	0.98	<code>hugepages</code>
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 <code>InnoDB buffer pool</code>	Reduce disk activity	2-5x
D2	Separate disks for <code>InnoDB table space</code> and log files	Parallelize disk activity	1.07
L250	Larger <code>InnoDB</code> log files	Reduce disk activity	1.08
F-	Use raw partition for <code>InnoDB table space</code>	Eliminate double journaling	1.07
H+	Enable <code>hugepages</code>	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;
```

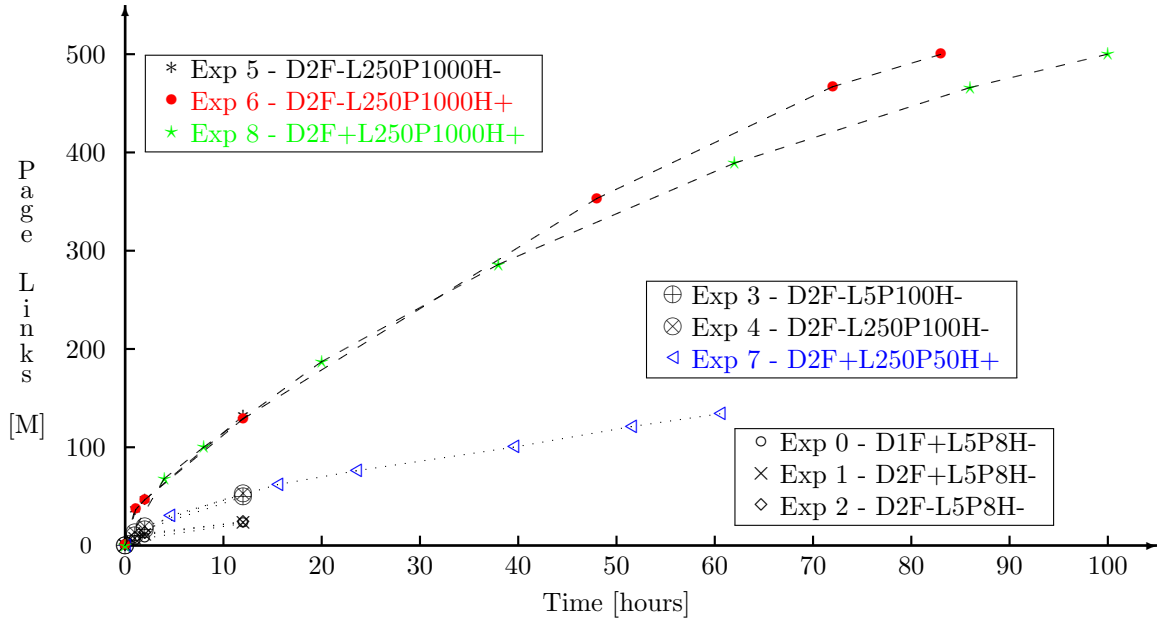
and run:

```
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` TableTable E.4: Speed of `MWdumper.jar` v. `importDump.php`

Method	Dry Run	MySQL	Rate [page/s]	Est. Time [s]	Est. Time [d]	Outcome
<code>mwumper.jar</code>	true	N/A	400	25,000	0.3	Fail
	false	Exper.6	200	50,000	0.6	Fail 309,000
<code>importDump.php</code>	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
		Exper.6	395			Fail 509,200

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. (Sheesh! 12 crashes. It took me 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 mediawiki-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 `mwddumper.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 `rawdumper.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 `rawdumper.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 counter 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 -xzipdf 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
Warning: Transient problem: FTP error Will retry in 1 seconds. 7 retries left.							
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
Warning: Transient problem: FTP error Will retry in 4 seconds. 5 retries left.							
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
Warning: Transient problem: FTP error Will retry in 16 seconds. 3 retries							
Warning: left.							
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
Warning: Transient problem: FTP error Will retry in 64 seconds. 1 retries							
Warning: left.							
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 a 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 `wiki` 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 ($2998\text{M}/16\text{K} = 187\text{k}$ 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:

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, strigi, updatedb.mlocat.	Yes	weekly
	php, gm	cpu	plasma-desktop with compositing.	Yes	daily

```
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
Modified db pages     35426
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages read 37501, created 44386, written 622566
0.00 reads/s, 0.81 creates/s, 6.26 writes/s
Buffer pool hit rate 1000 / 1000
...
-- database pages
-- read + created
-- hit rate
```

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. 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-unique> 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	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` presumably 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 yields 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	<code>rebuildImages.php</code>	<code>importDump.php</code>
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/mediawiki/images/f/f7/Bunny_Breckinridge_(cropped).png).
```

```
/usr/bin/gm convert: Invalid JPEG file structure: missing SOS marker (/var/lib/mediawiki/images/9/9b/Vw-baunatal-brand.jpg).
```

```
/usr/bin/gm convert: Invalid SOS parameters for sequential JPEG (/var/lib/mediawiki/images/e/e2/Vf142ghost.jpg).
```

```
/usr/bin/gm convert: Missing PLTE before bKGD (/var/lib/mediawiki/images/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 (/var/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_depth (/var/lib/mediawiki/images/7/76/Communes_of_Luxembourg.png).
```

```
/usr/bin/gm convert: Warning: unknown JFIF revision number 2.01 (/var/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/includes/GlobalFunctions.php on line 2423

PHP Notice: Undefined index: scheme in /usr/share/mediawiki/includes/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_\(Official\)](http://www.mediawiki.org/wiki/Extension:FCKeditor_(Official))

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/gephi.php on line 3300
```

GeSHi stands for Generic Syntax Highlighter. Debian packages are `php-geshi` and `mediawiki-extensions-gephi`.

Appendix F

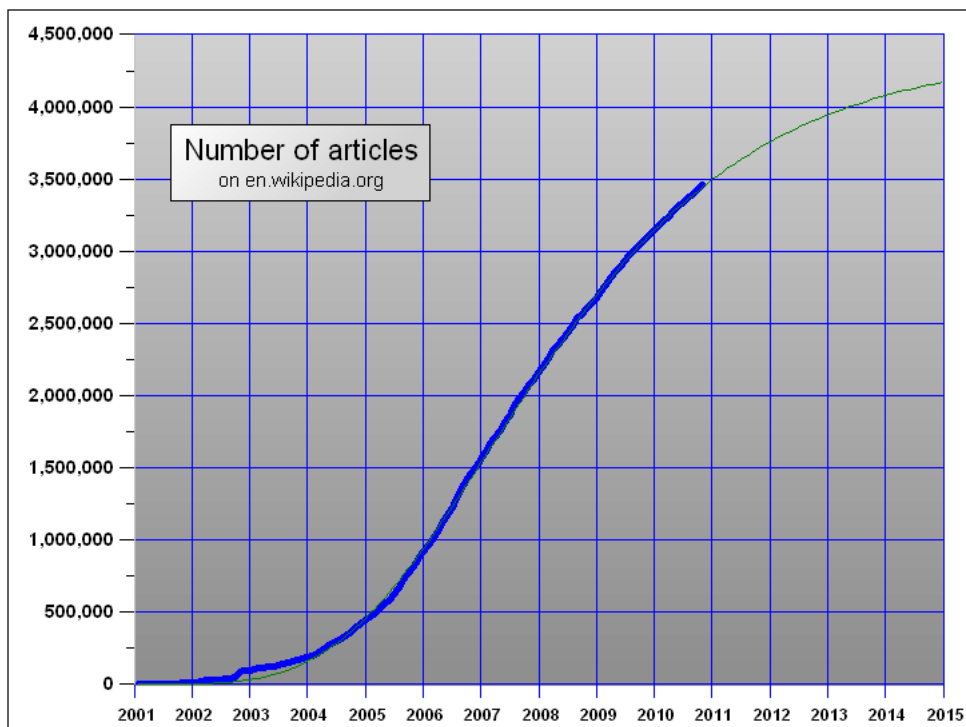
Trends (2011-Dec-21)

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	
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	40G
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 2T.

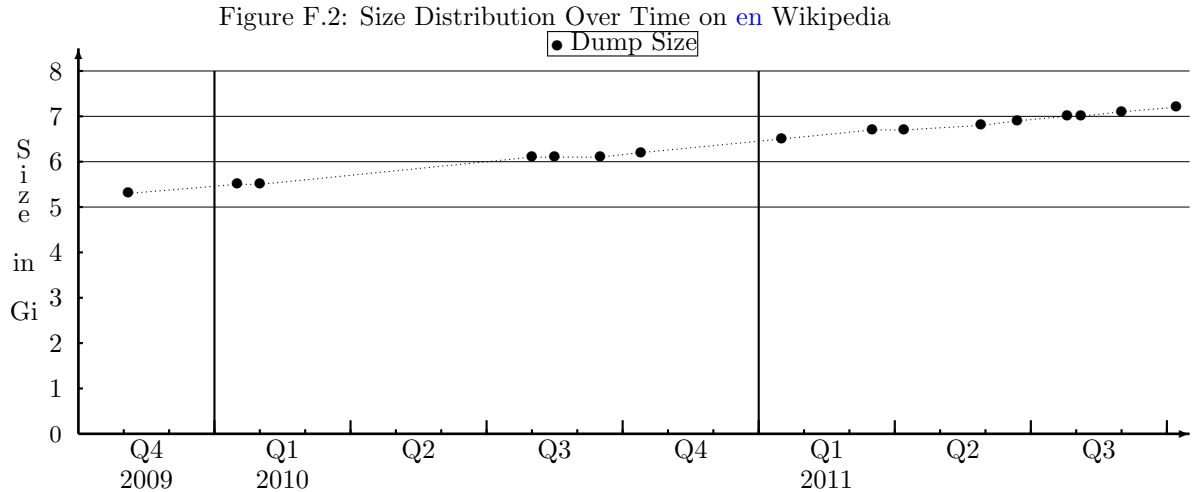
F.3 Size Distribution

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-yyyyymmdd-pages-articles.xml.bz2](#) file has been growing. See [Figure F.2, Size Distribution Over Time on en Wikipedia](#).

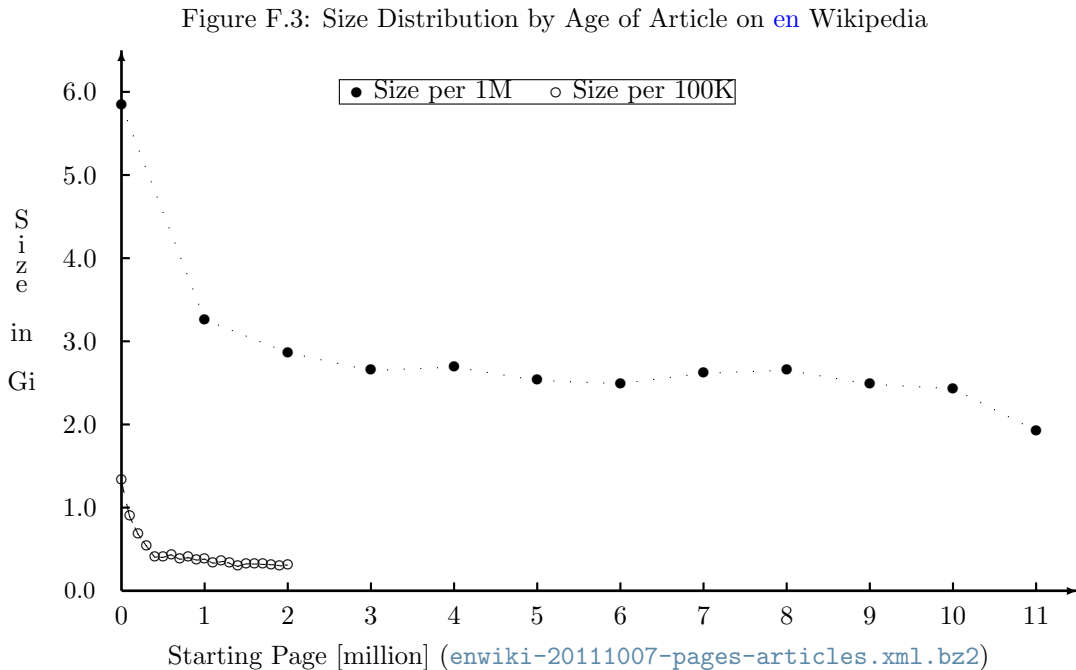


F.3.3 Size Distribution—by Age of Article

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](#).

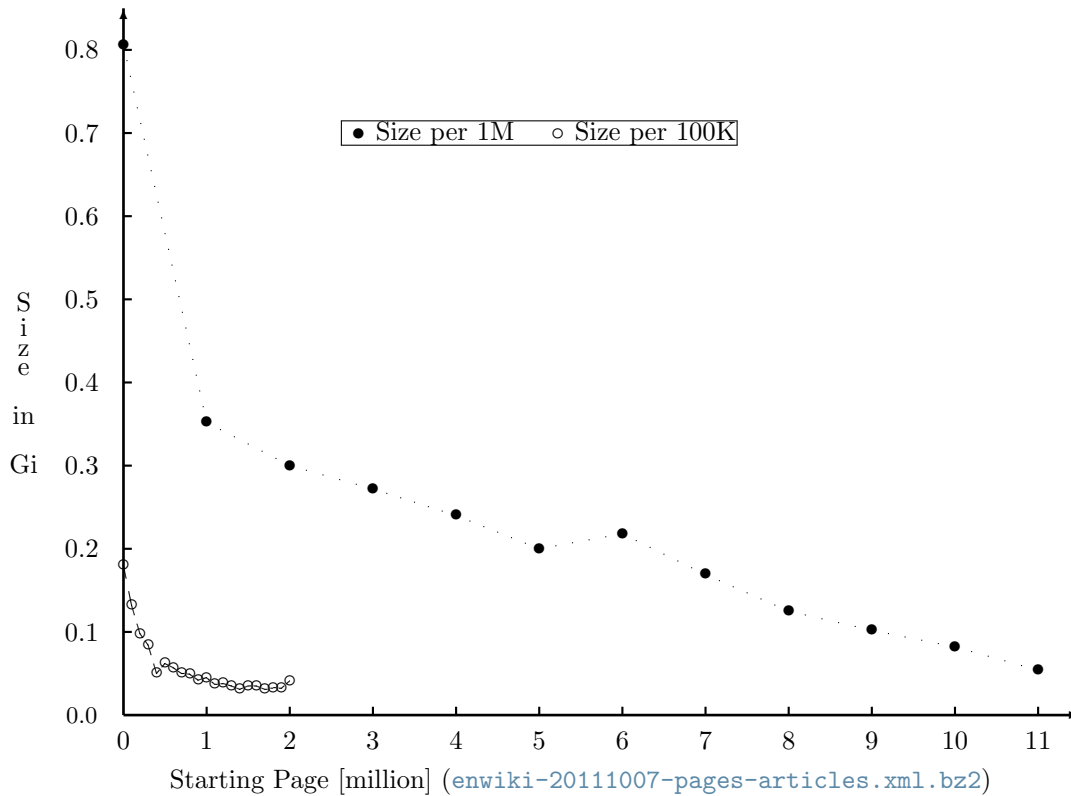
Processing of the oldest pages consumes more resources (disk, cpu, bandwidth, and time) than newer pages. The image download rate (measured in [ichunks](#) per day) accelerates, because the newer pages have fewer images. However, the page importation rate (measured in [xchunks](#) per day):

- 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.



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](#).

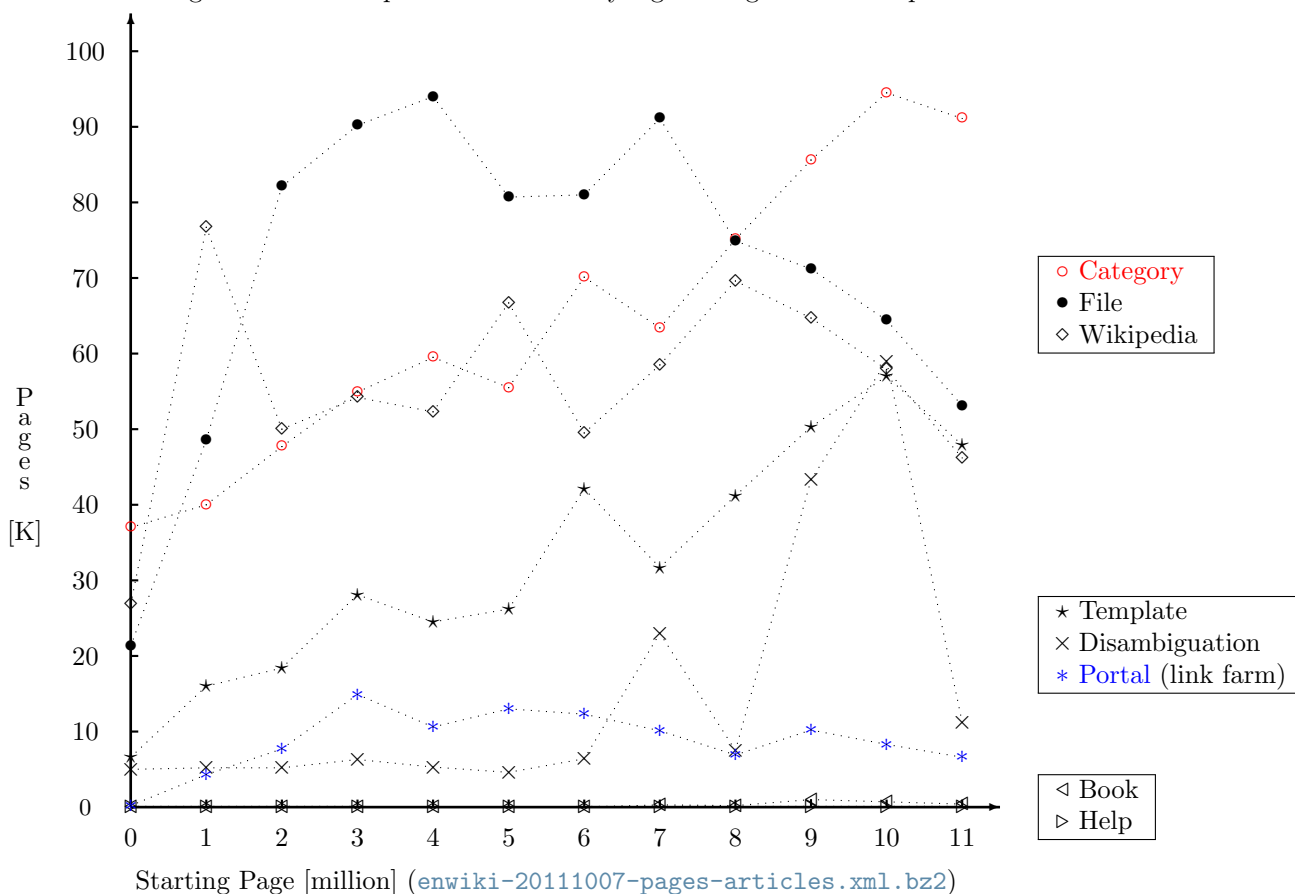
Some trends are apparent. The Category and Template namespaces have become more popular with time. Recently there was a burst of activity writing Disambiguation pages (between pages 9,000,000 and 11,000,000). The File namespace is gradually declining, perhaps due to the

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

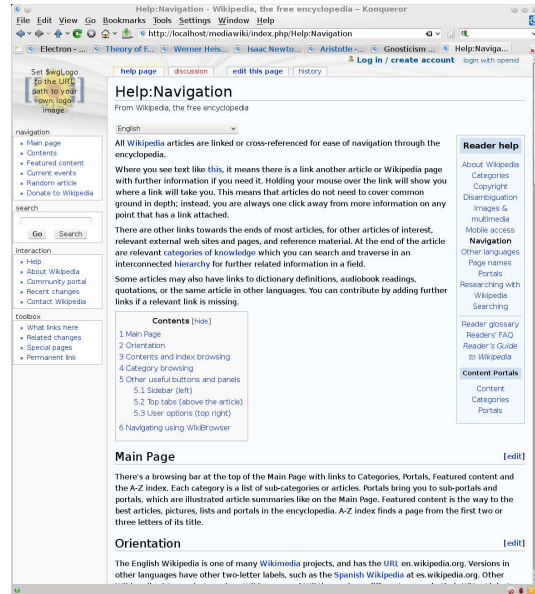


Figure F.8: Page in Main Namespace

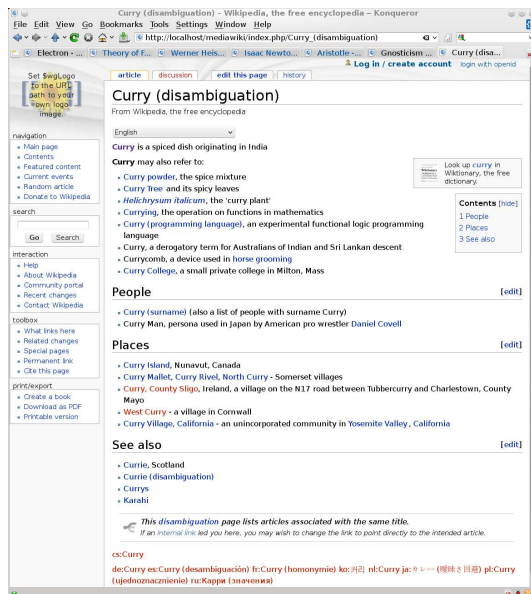


Figure F.9: Page in Portal Namespace



Figure F.10: Page with Small Template

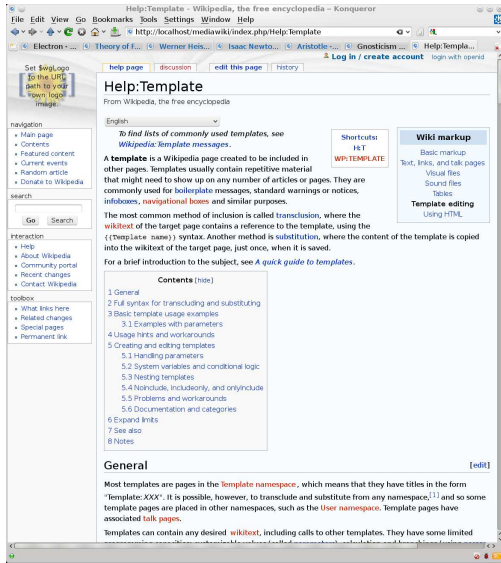


Figure F.11: Page with Large Template

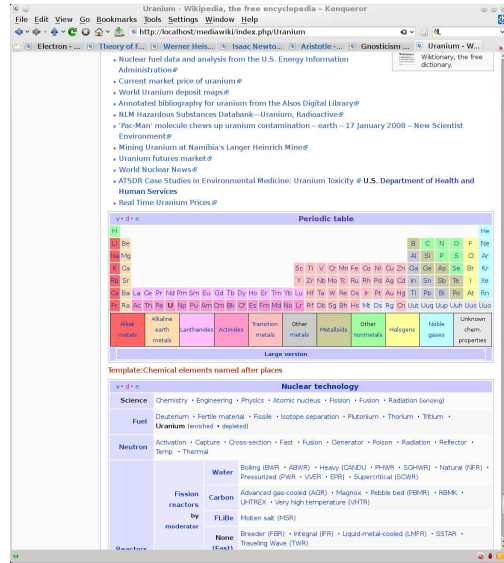
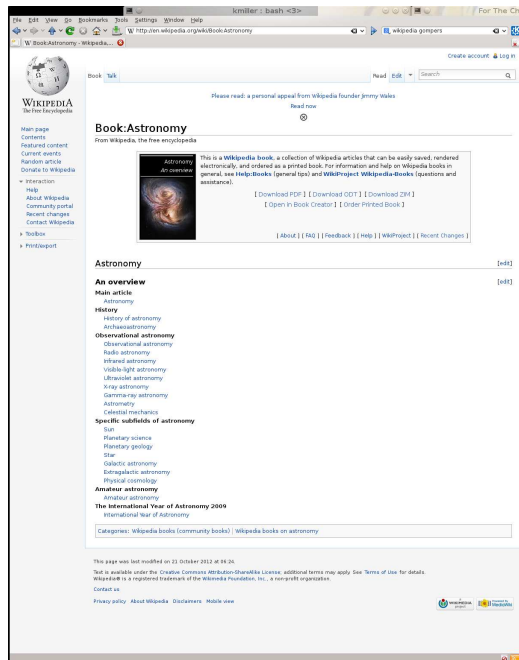


Figure F.12: Page in Wikipedia Namespace



Figure F.13: Page in Book Namespace



GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

`<http://fsf.org/>`

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

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. Secondly, 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 a 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.