MirrorBrain Documentation

Release 2.19.0

Peter Poeml

Feb 27, 2022

CONTENTS

Release 2.19.0

Date Feb 27, 2022

CHAPTER

ONE

INTRODUCTION

This documentation describes how to install, configure and use MirrorBrain.

It may be useful to consult the *Release Notes/Change History* for changes.

The detailed history of changes in the documentation itself could be looked at here.

1.1 How to improve this documentation

Working on the documentation is easy. This section explains how it is done.

1.1.1 Sources

The documentation sources are maintained in the docs subdirectory of the MirrorBrain Subversion repository. The source is formatted in reStructuredText. reStructuredText is simple to learn, and resembling Wiki markup which is quick to edit. Every page of the online documentation has a link named "Source" on top of the page which displays the corresponding source file in the subversion repository. You can use this to look at the source and get a feeling for the way the format works.

HTML is generated using the Sphinx Python Documentation Generator. Every change in the git repository becomes visible at http://mirrorbrain.org/docs/, through a post-commit hook running the generator.

Check out a working copy of the source with this command:

```
git clone https://github.com/poeml/mirrorbrain.git
```

The reStructuredText Primer is a helpful resource.

1.1.2 Submitting changes

To submit changes, there are several options:

- sending patches to the mirrorbrain@mirrorbrain.org mailing list
- requesting write access to the subversion repository (do so by writing to the mailing list)
- of course, it is fully appropriate (and appreciated) if you simply send plain mail, pointing out deficiencies, or giving suggestions.

1.1.3 Testing documentation locally before committing

It is useful to test changes by generating HTML locally before committing. To be able to do this, you need to install the Sphinx Python Documentation Generator. It is available readily packaged on most platforms, named `python-sphinx`, `py25-sphinx` or similarly.

Generating the documentation is as easy as:

```
cd docs
make html
```

Then just open `_build/html/index.html` in your web browser. Simply rerun the **make** command after changes, watch its output for errors or warnings, and reload your browser window.

1.2 Implementation and design notes

1.2.1 Database hash store

Since 2.13.0, all hashes are stored in the database. Before, they were kept in files in the file system. The old store was suitable only for generation of old-style Metalinks.

Inside the database, the hashes are stored as compart binary blobs. For transfer, they are converted to hexadecimal. This is due to the following design decision: Storage is binary in so-called `bytea` columns. PostgreSQL automatically escapes binary (bytea) data on output in its own way. But this encoding is not very efficient in space. Hex encoding is more efficient (it results in shorter strings, and thus less data to transfer over the wire, and it's also faster). The escape format is kind phased out, and it doesn't make sense to use it in a new application (which we are). On the other hand, storage in bytea is as compact as it can be, which is good. So we store the data in binary, and provide a database view which converts to hex on the fly. The hex encoding function in PostgreSQL seems to be fast.

CHAPTER TWO

FAQ - FREQUENTLY ASKED QUESTIONS

2.1 Is it required that all mirror servers have the same data set of files?

No. On the contrary, this is one of the things where MirrorBrain shines, in that it doesn't matter at all what the mirrors chose to mirror. In the extreme, a mirror could have just a single file; it doesn't matter.

2.2 Is MirrorBrain suitable only for Linux distributions?

No, it is designed explicitely for normal web browsers and other download clients, for normal files to be downloaded. It does not require special software, nor does it require users to have any Linux experience. Generic applicability and modularity were among the primary design goals from the beginning. Another explicit goal was to not design for a specific system or specific software, or particular file sizes (large or small).

Having said that, MirrorBrain provides useful metadata for more sophisticated download clients.

2.3 Is only HTTP supported as protocol?

No — FTP mirrors are fully supported, in addition to HTTP. Furthermore, BitTorrent and Metalinks can be used.

To scan mirrors for their content, rsync is used. It is the most efficient method for that purpose. However, if rsync isn't available on a mirror, FTP and HTTP can be used as fallback.

2.4 Your fundamental design looks like that everything has to be done in a single machine. Right?

No, it only means that clients are sent to machines under control of the owner of the distribution and from there to mirrors. If it is a single machine or many does not matter. For instance, you could run different MirrorBrain servers on different continents, and send clients to the nearest server via GeoDNS.

2.5 Do you have plans to support MySQL in addition to PostgreSQL?

No way, I started with MySQL initially, and that was all good, but then I needed more compact storage for the vast openSUSE file tree. I could reduce database size to 30% by migrating to PostgreSQLs more flexible data types. Speed was increased by factor 5. See here for details.

But what's more, I later started to use indexed network addresses (required for autonomous system lookups, in order to match clients to networks). There is no performant way to do this in MySQL, since it requires a Patricia Trie as index. Only PostgreSQL offers a (third party) data type that can do this, called ip4r. See mod_asn.

By the way, so-called stored procedures are a very nice thing that PostgreSQL brings along. It makes programming so much easier if the database can be equipped with high-level procedures on board, that don't need to be sent to the database, don't need to be parsed over and over. Very elegant to use. I never looked back.

2.6 How can requests/traffic be logged?

With MirrorBrain, Apache can log additional details like client location and the chosen mirror. Since MirrorBrain is designed to see all requests (instead of having clients go directly to mirrors, and not seeing them ever again), rich logging is possible.

This design also allows to collect download statistics easily, something that most large projects want. Please refer to the concept for download statistics for more on this.

2.7 How much traffic can MirrorBrain handle?

The openSUSE project has been using MirrorBrain since its inception, and has pushed at least 250 GBit/s to users (probably more, that's a hard number measured that doesn't comprise all used mirrors). This has been served from a really old two-way box getting 20.000.000 to 40.000.000 hits a day, i.e. 250-450 per second, with a load average of 1 or less. The traffic has been distributed to more than 150 active mirrors (the mirrors obviously being the ones doing the hard work).

2.8 Does a copy of the mirrored content have to be kept locally?

Yes, and no.

Normally, yes. This is by design: it allows for correct handling of If-modified-since requests, you can always deliver files directly, you can always look at the master server(s) what's there (conveniently), and you can exclude certain files from being redirected at all (e.g. signatures, or quickly changing metadata).

There are some more subtle implications regarding the local file tree, for example that Apache can check for the existance of a requested file very quickly, while it would require a database hit otherwise. For tiny files, the database lookup can be saved as well; delivering a file 2048 bytes in size is faster than doing a database lookup and returning a redirect which would be about the same size anyway (and it saves the clients an additional roundtrip to another place).

Of course, each setup is different and all of the above might not matter under given circumstances. As an alternative, there is a way to set up the system without a full local copy of the file tree: the local file tree can be substituted with a dummy file tree. This still needs to be updated, but at least it doesn't take significant space, and guarantees correct handling of If-modified-since requests. You will find details here.

CHAPTER

THREE

PLATFORMS

This table is work in progress (started in 2014), in an attempt to collect an overview of platform features.

Platform	Apache	PostgreSQL	ip4r	packaged	tested	remarks
Debian 9.0 (next)						
Debian 8.0 (Jessie)	2.4.10	9.4	2.0.2	obs		
Debian 7.0 (Wheezy)	2.2.22	9.1	1.05	obs	yes	stephan48
Debian 6.0 (Squeeze)	2.2.16	8.4	1.04			mod_geoip only 1.2.5, GeoIP too
Debian 5.0 (Lenny)						
Ubuntu 13.10 (Saucy Salamander)	2.4.6	9.1	2.0			
Ubuntu 13.04 (Raring Ringtail)	2.2.22	9.1	1.05			
Ubuntu 12.10 (Quantal Quetzal)	2.2.22	9.1	1.05			sqlobject upstream bug was fixed
Ubuntu 12.04 LTS (Precise Pangolin)	2.2.22	9.1	1.05	obs	2.28.1	floeff; mod_geoip only 1.2.5, but
Ubuntu 11.10 (Oneiric Ocelot)					2.17.0	floeff
Ubuntu 10.04 LTS (Lucid Lynx)	2.2.14	8.4	1.04			mod_geoip too old, GeoIP too ol
openSUSE 13.2						
openSUSE 13.1	2.4.6	9.2	1.05			
openSUSE 12.3	2.2.22	9.2	1.05			
openSUSE 12.2	2.2.22	9.1	1.05			
openSUSE 12.1	2.2.21	9.1	1.05	no longer	2.17.0	
SUSE SLE 11	2.2.10		1.05			
SUSE SLE 10	2.2.0(?)					
Rawhide	2.4.7	9.3	2.0			
Fedora 20	2.4.6	9.3	1.05			
Fedora 19	2.4.6	9.2	1.05			
Fedora 18	2.4.6	9.2	1.05			
RHEL 6					no	mirrorbrain package builds in OE
RHEL 5						
RHEL 4						
CentOS 6						
CentOS 5						

Suggestions for other features to track:

- mod_geoip with IPv6 capability

- see also http://mirrorbrain.org/issues/issue16
- very old mod_geoip versions *1.1.8* didn't return continent lookup data

Ubuntu mod_geoip versions: http://packages.ubuntu.com/search?keywords=mod-geoip&searchon=names Debian mod_geoipip versions http://packages.debian.org/search?suite=all&searchon=names&keywords=libapache2-mod-geoip

Package search URLs:

- https://packages.debian.org/search
- http://packages.ubuntu.com/ and https://launchpad.net/

CHAPTER

FOUR

INSTALLATION

4.1 Prerequirements

4.1.1 General

The file tree to be served needs to be accessible locally by the Apache that runs mod_mirrorbrain. (See the FAQ as well as *Creating a file tree*.)

The hardware needs are mediocre; MirrorBrain needs few resources.

4.1.2 Apache

A recent enough version of the Apache HTTP server is required. **2.2.6** or later should be used. In addition, the apr-util library should be **1.3.0** or newer, or at least a not-too-old **1.2.x** release. This is because the DBD database pool functionality was developed mainly around 2006 and 2007, and reached production quality at the time. This will mean that you have to upgrade Apache when installing on an oldish enterprise platform.

Status of Apache version on individual platforms

openSUSE/SLE Sufficiently new, since openSUSE 11.0 and SLE11. SLE11 does not ship the required PostgreSQL database adapter. The one from openSUSE 11.1 would work. Current and stable Apache for openSUSE/SLE can always be found here: http://download.opensuse.org/repositories/Apache/

CentOS 5/RHEL Ancient Apache. It *might* work or not – you might need to get a newer one. Feedback would be appreciated!

Debian The Apache in *Lenny* (or newer) is fine. The version in *Etch* was too old.

Ubuntu Apache is new enough and known to work at least since *9.04*.

Arch Linux Has a new enough Apache.

Gentoo Has a new enough Apache.

4.1.3 Frontend (mod_mirrorbrain, the redirector)

- if geographical mirror selection is going to be used, mod_geoip and libGeoIP are required.
- If mod_geoip is used, it needs to be version 1.2.0 or newer. See http://mirrorbrain.org/issues/issue16
- mod_form, plus a patch preserving the arguments that it parses for other modules, like mod_autoindex.
- if you want to compile with the optional memcache support (there should not be reason for it, though), you would need libapr_memcache, mod_memcache, memcache daemon

4.1.4 Database

- PostgreSQL
- mod_mirrorbrain, the core of MirrorBrain, is not really bound to a particular database; you could use MySQL just as well, SQLite, or Oracle - everything that the Apache DBD API has a driver for. The admin framework and tool set however is currently provided for PostgreSQL only. Therefore, it doesn't make sense to use a different database, unless you are prepared to extend MirrorBrain, or have a special setup in mind (like, integrating MirrorBrain with some existing database).
- mod_asn is optional, and needed only for refined mirror selection and full exploitation of network locality. It works only with PostgreSQL as database. It needs a data type for PostgreSQL called "ip4r" that needs to be installed additionally.

  The question whether you should (or should not) extend MirrorBrain with mod_asn is discussed in the section *Installing mod_asn*.

  If you intend to install it, confer to its documentation and its prerequirements).

4.1.5 Python and Perl modules

The toolset for database maintenance needs Python (an old 2.4.x is sufficient) and the following Python modules:

- `cmdln`
- `sqlobject`
- `psycopg2`

The following Perl modules are required:

- `Config::IniFiles`
- `libwww::perl`
- `DBD::Pg`
- `Digest::MD4`
- `Date::Parse`

(If you install MirrorBrain in pre-packaged form, all these requirements should automatically be met.)

The following sections will guide you through installing the various components.

4.2 Installing from source

4.2.1 Installing Apache

Build or install Apache 2.2.6 or later.

When installing/building the Apache Portable Runtime (APR), make sure that you build the required database adapter for the DBD module for PostgreSQL.

4.2.2 Building Apache modules

mod_form

Install mod_form. The sources are here:

- http://apache.webthing.com/svn/apache/forms/mod_form.c
- http://apache.webthing.com/svn/apache/forms/mod_form.h

Its header file will be needed later to compile mod_mirrorbrain.

It is useful to apply the following patch to mod_form.c:

```
Tue Mar 13 15:16:30 CET 2007 - poeml@cmdline.net

preserve r->args (apr_strtok is destructive in this regard). Makes
mod_autoindex work again in conjunction with directories where FormGET is
enabled.

--- mod_form.c.old      2007-03-13 15:05:13.872945000 +0100
+++ mod_form.c  2007-03-13 15:06:26.378367000 +0100
@@ -61,6 +61,7 @@
   char* pair ;
   char* last = NULL ;
   char* eq ;
+  char* a ;
   if ( ! ctx ) {
     ctx = apr_pcalloc(r->pool, sizeof(form_ctx)) ;
     ctx->delim = delim[0];
@@ -69,7 +70,8 @@
   if ( ! ctx->vars ) {
     ctx->vars = apr_table_make(r->pool, 10) ;
   }
-  for ( pair = apr_strtok(args, delim, &last) ; pair ;
+  a = apr_pstrdup(r->pool, args);
+  for ( pair = apr_strtok(a, delim, &last) ; pair ;
         pair = apr_strtok(NULL, delim, &last) ) {
     for (eq = pair ; *eq ; ++eq)
       if ( *eq == '+' )
```

mod_mirrorbrain

The main Apache module, **mod_mirrorbrain**, can be built with the following steps:

```
# unpack the tarball and go inside the source tree
cd mod_mirrorbrain
apxs -c -lm mod_mirrorbrain.c
```

To install the module in the right place, you would normally call:

```
apxs -i mod_mirrorbrain.c
```

Building, installing and activation can typically be combined in one **apxs** call:

```
apxs -cia -lm mod_mirrorbrain.c
```

mod_autoindex_mb

Build and install mod_autoindex_mb:

```
# in mirrorbrain source tree
cd mod_autoindex_mb
apxs -cia mod_autoindex_mb.c
```

mod_geoip

Install the GeoIP library, the accompanying commandline tools, and the geoip Apache module.

Configure mod_geoip:

```
GeoIPEnable On
GeoIPOutput Env
GeoIPDBFile /var/lib/GeoIP/GeoIP.dat MMapCache
```

(You would typically put this into the server-wide context of a virtual host.)

Note that a caching mode like MMapCache needs to be used, when Apache runs with the worker MPM. See http://forum.maxmind.com/viewtopic.php?p=2078 for more information.

You need to build two commandline tools for GeoIP:

```
cd tools
gcc -Wall -o geoiplookup_continent geoiplookup_continent.c -lGeoIP
gcc -Wall -o geoiplookup_city geoiplookup_city.c -lGeoIP
```

4.2.3 Installing Python and Perl modules

Install the following Python modules:

- python-cmdln python-SQLObject python-FormEncode python-psycopg2

Install a few Perl modules as well (required for the mirror scanner, which is written in Perl):

- perl-Config-IniFiles perl-libwww-perl perl-DBD-Pg perl-TimeDate
- perl-Digest-MD4 (it is not *really* needed, but prevents an ugly error message)

4.2.4 Installing PostgreSQL

Install the PostgreSQL server, start it and create a user and a database:

```
su - postgres
postgres@powerpc:~> createuser -P mirrorbrain
postgres@powerpc:~> createdb -O mirrorbrain mirrorbrain
postgres@powerpc:~> createlang plpgsql mirrorbrain
```

Maybe it is a good idea to check PostgreSQL's access policy configuration at this point. The default should already be fine for local access via password. But you can add access from a remote host if needed, as shown in the bottom line:

```
postgres@powerpc:~> cp data/pg_hba.conf data/pg_hba.conf.orig
postgres@powerpc:~> vi data/pg_hba.conf

# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
# "local" is for Unix domain socket connections only
local   all         all                               password
# IPv4 local connections:
host    all         all         127.0.0.1/32          password
# IPv6 local connections:
host    all         all         ::1/128               password
# remote connections:
host    mirrorbrain mirrorbrain 10.10.2.3/32          md5
```

Note: I recommend to use `password` or `md5` authentication in all lines. Remove `ident`; it is not so handy, at least not if you aren't always the same user. `password` is transmitting in clear text, so one should use `md5` for all remote connections.

If you plan to use mod_asn for lookup of AS (autonomous system) data, now's the moment to install the `ip4r` data type into PostgreSQL. See the mod_asn documentation for instructions.

4.2.5 Creating database schema

Import the table structure and initial data:

```
psql -U mirrorbrain -f sql/schema-postgresql.sql mirrorbrain
psql -U mirrorbrain -f sql/initialdata-postgresql.sql mirrorbrain
```

4.2.6 Creating a "mirrorbrain" user and group

Create a "mirrorbrain" user and group:

```
groupadd -r mirrorbrain
useradd -r -g mirrorbrain -s /bin/bash -c "MirrorBrain user" -d /home/mirrorbrain␣
↪mirrorbrain
```

4.2.7 Installation of the tools

You need to install a number of the provided tools to a location in your $PATH. Unfortunately, there is no Makefile to take this work off you. Hopefully, one can be provided later:

```
install -m 755 tools/geoiplookup_continent /usr/bin/geoiplookup_continent
install -m 755 tools/geoiplookup_city      /usr/bin/geoiplookup_city
install -m 755 tools/geoip-lite-update     /usr/bin/geoip-lite-update
install -m 755 tools/null-rsync            /usr/bin/null-rsync
install -m 755 tools/scanner.pl            /usr/bin/scanner
install -m 755 mirrorprobe/mirrorprobe.py  /usr/bin/mirrorprobe
```

The following command should build and install the **mb** admin tool:

```
setup.py install [--prefix=...]
```

4.2.8 Configuring Apache

Load the Apache modules:

```
a2enmod form
a2enmod geoip
a2enmod dbd
a2enmod mirrorbrain
```

Configure the database adapter (mod_dbd), resp. its connection pool. Put the configuration into server-wide context. Config example:

```
# for prefork, this configuration is inactive. prefork simply uses 1
# connection per child.
<IfModule !prefork.c>
        DBDMin  0
        DBDMax  32
        DBDKeep 4
        DBDExptime 10
</IfModule>
```

Configure the database driver. Put the following configuration into server-wide OR vhost context. Make the file chmod 0640, owned root:root because it will contain the database password:

```
DBDriver pgsql
# note that the connection string (which is passed straight through to
# PGconnectdb in this case) looks slightly different - pass vs. password
DBDParams "host=localhost user=mirrorbrain password=12345 dbname=mirrorbrain connect_
↪timeout=15"
```

Note: The database connection string must be unique per virtual host. This matters if several MirrorBrain instances are set up in one Apache. If the database connection string is identical in different virtual hosts, mod_dbd may fail to associate the connection string with the correct virtual host. A possible workaround is to use differing values in connect_timeout.

4.2.9 Next steps

From here, follow on with *Initial configuration steps on all platforms*.

4.3 Installation on openSUSE Linux or SLE

4.3.1 Adding package repositories

Add the needed repositories, using the subdirectory that matches your distribution version:

- http://download.opensuse.org/repositories/Apache:/MirrorBrain/
- http://download.opensuse.org/repositories/devel:/languages:/python/
- http://download.opensuse.org/repositories/server:/database:/postgresql/

You can do this via commandline (we are using openSUSE 13.1 in our example):

```
zypper addrepo --refresh http://download.opensuse.org/repositories/Apache:/MirrorBrain/
↪Apache_openSUSE_13.1 Apache:MirrorBrain
zypper addrepo --refresh http://download.opensuse.org/repositories/devel:/languages:/
↪python/openSUSE_13.1 devel:languages:python
zypper addrepo --refresh http://download.opensuse.org/repositories/server:/database:/
↪postgresql/openSUSE_13.1 server:database:postgresql
```

4.3.2 Installing the MirrorBrain packages

Here's a list of packages needed to have one host running the database and the redirector:

> apache2 apache2-mod_asn apache2-mod_geoip apache2-mod_mirrorbrain apache2-mod_form apache2-worker GeoIP libapr-util1-dbd-pgsql libGeoIP1 mirrorbrain mirrorbrain-scanner mirrorbrain-tools perl-Config-IniFiles perl-DBD-Pg perl-Digest-MD4 perl-libwww-perl perl-TimeDate postgresql postgresql-ip4r postgresql-server python-cmdln python-psycopg2 python-SQLObject python-FormEncode

Note: If the web server is set up on a separate host than the database server, the web server needs only the package libapr-util1-dbd-pgsql and no other postgresql* packages.

You can install the packages via the following commandline:

```
zypper install apache2-worker \
apache2-mod_asn apache2-mod_mirrorbrain \
postgresql-server postgresql-ip4r \
mirrorbrain mirrorbrain-scanner mirrorbrain-tools
```

The packages not mentioned in this commandline are drawn in via package dependencies.

4.3.3 Next steps

From here, follow on with *Initial configuration steps on all platforms*.

4.4 Installation on Debian/Ubuntu Linux

Note: The following recipe for installing MirrorBrain was tested on Ubuntu 9.04 and 10.04. A similar procedure should work on other Ubuntu versions, and on Debian 5.0.

Install a standard Ubuntu LAMP server.

4.4.1 Add package repository

To subscribe to the repository with packages for Ubuntu 10.04, add the following to `/etc/apt/sources.list`:

```
 sudo vim /etc/apt/sources.list
[...]
deb http://download.opensuse.org/repositories/Apache:/MirrorBrain/Ubuntu_10.04/ /
```

There are more repositories at http://download.opensuse.org/repositories/Apache:/MirrorBrain/ for other Ubuntu and Debian releases.

After adding the repository, update the local **apt-get** package cache:

```
sudo apt-get update
```

That will produce a warning message saying that a GPG key isn't known for the new package repository. Take note of the key ID and import the key with *apt-key*:

```
 # sudo apt-key adv --keyserver hkp://wwwkeys.de.pgp.net --recv-keys BD6D129A
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring \
  --secret-keyring /etc/apt/secring.gpg --trustdb-name /etc/apt/trustdb.gpg \
  --keyring /etc/apt/trusted.gpg --keyserver hkp://keys.gnupg.net --recv-keys \
  BD6D129A
gpg: requesting key BD6D129A from hkp server hkp://wwwkeys.de.pgp.net
gpg: key BD6D129A: public key "Apache OBS Project <Apache@build.opensuse.org>" imported
```

(continues on next page)

(continued from previous page)

```
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:               imported: 1
```

If you now run **apt-get** again, the warning should be gone:

```
sudo apt-get update
```

Note: The key's validity may need to be refreshed later. If apt-get stops working, see *Refreshing package sign keys*.

4.4.2 Install the MirrorBrain packages

The following commands will install all needed software via **apt-get**:

```
sudo apt-get install mirrorbrain mirrorbrain-tools mirrorbrain-scanner \
libapache2-mod-mirrorbrain libapache2-mod-autoindex-mb
```

4.4.3 Select and install an Apache MPM

The MirrorBrain packages have dependencies on the Apache common packages, but not on a MPM, since the choice of an MPM is one that the system admin must make, and the MPMs cannot be installed in parallel. Thus, an MPM needs to be installed (unless a LAMP package selection was installed initially).

To install the worker MPM, run:

```
sudo apt-get install apache2-mpm-worker
```

If the LAMP server has been installed, the prefork MPM was probably preselected. It may make sense to switch to the worker MPM in such cases, which is a good choice for busy download servers. If something like PHP is in use as embedded interpreter (mod_php), though, then you need to stick to the prefork MPM, because libraries that are used by PHP might not be threadsafe.

Loading Apache modules

Don't forget to load the needed Apache modules:

```
a2enmod form
a2enmod mirrorbrain
a2enmod geoip
a2enmod dbd
a2enmod autoindex_mb   # instead of Apache's own mod_autoindex
a2enmod asn # only if you use that module as well
```

Configure mod_geoip

Note: The mod_geoip Apache module that ships with Debian and Ubuntu is too old (1.1.x, see http://mirrorbrain.org/issues/issue16). Therefore the provided packages also include a newer mod_geoip (1.2.x). With the above apt-get line, you've got it already installed.

mod_geoip is configured to look for the GeoIP data set in `/usr/share/GeoIP`. A dataset may be installed already, but it is recommended to update it. In fact, that should be done regularly. There is handy tool that does this, and also downloads the larger "City" dataset:

```
# l /usr/share/GeoIP
total 1296
-rw-r--r-- 1 root root 1204947 2010-01-18 08:46 GeoIP.dat
-rw-r--r-- 1 root root  109251 2010-01-18 08:46 GeoIPv6.dat
# geoip-lite-update
 * Reloading web server config apache2
   ...done.
# l /usr/share/GeoIP
total 52884
-rw-r--r-- 1 root root  1204947 2010-01-18 08:46 GeoIP.dat
-rw-r--r-- 1 root root   591865 2010-09-26 14:26 GeoIP.dat.gz
-rw-r--r-- 1 root root  1072630 2010-09-26 14:26 GeoIP.dat.updated
-rw-r--r-- 1 root root   109251 2010-01-18 08:46 GeoIPv6.dat
-rw-r--r-- 1 root root   652498 2010-09-26 14:44 GeoIPv6.dat.gz
-rw-r--r-- 1 root root  1214981 2010-09-26 14:44 GeoIPv6.dat.updated
-rw-r--r-- 1 root root 20478077 2010-09-26 14:27 GeoLiteCity.dat.gz
-rw-r--r-- 1 root root 30605325 2010-09-26 14:27 GeoLiteCity.dat.updated
```

Now, one (or more) of the files ending in `.updated` can be used with Apache.

The larger dataset (`GeoLiteCity.dat.updated`) is recommended.

Configure mod_dbd

With Ubuntu 9.04, the DBD (Apache Portable Runtime DBD Framework) database adapter for PostgreSQL is already installed, because the driver is statically linked into the libaprutil1 shared object. libaprutil1-dbd-pgsql is a virtual package which is just a pointer to the libaprutil1 package.

Running the following snippet will create a configuration for mod_dbd:

```
sudo sh -c "cat > /etc/apache2/mods-available/dbd.conf << EOF
 <IfModule mod_dbd.c>
    DBDriver pgsql
    DBDParams 'host=localhost user=mirrorbrain password=12345 dbname=mirrorbrain connect_
↪timeout=15'
 </IfModule>
EOF
"
```

Note: Edit the password in the template here – take note of it, you'll need it below, when you create a database user account.

Note: Important: DBDParams strings must be unique; you cannot use the same string in another vhost. A possible workaround is to use differing connect_timeout values.

4.4.4 Install PostgreSQL

Install the PostgreSQL server (here, version 8.4 is the current version):

```
sudo apt-get install postgresql-8.4
```

Create the postgresql user account and database

Switch to user postgres:

```
sudo su - postgres
```

Create user:

```
createuser -P mirrorbrain
Enter password for new role:
Enter it again:
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n
Shall the new role be allowed to create more new roles? (y/n) n
```

Create database:

```
createdb -O mirrorbrain mirrorbrain
createlang plpgsql mirrorbrain
```

Exit user postgres:

```
exit
```

Edit host-based authentication

Add line `host mirrorbrain mirrorbrain 127.0.0.1/32 md5` to the end of `pg_hba.conf`, which is to be found here:

```
sudo vim /etc/postgresql/8.4/main/pg_hba.conf
```

Start the PostgreSQL server:

```
sudo /etc/init.d/postgresql-8.4 restart
```

Import initial mirrorbrain data

Import structure and data, running the commands as user mirrorbrain:

```
sudo su - mirrorbrain
gunzip -c /usr/share/doc/mirrorbrain/sql/schema-postgresql.sql.gz | psql -U mirrorbrain␣
↪mirrorbrain
gunzip -c /usr/share/doc/mirrorbrain/sql/initialdata-postgresql.sql.gz | psql -U␣
↪mirrorbrain mirrorbrain
exit
```

4.4.5 Next steps

From here, follow on with *Initial configuration steps on all platforms*.

4.5 Installation on Gentoo Linux

These are very rough notes, which might be helpful for someone as a start maybe.

Note: See also http://wiki.github.com/ramereth/ramereth-overlay/gentoo-mirrorbrain-howto (which should be merged with this documentation)

Warning: When I (Peter) took the following notes, it was my first experience in setting up a Gentoo system.

Because I initially emerged git, lots of Perl modules which are be neded later might already be on the system.

curl will be useful for testing later:

```
emerge curl
```

Build and start the PostgreSQL server:

```
emerge postgresql
emerge --config =postgresql-8.1.11
su - postgres
pg_ctl -D /var/lib/postgresql/data start
```

Build GeoIP:

```
emerge geoip
```

Edit `/etc/make.conf` and add the following settings:

```
USE="postgres"
APACHE2_MPMS="worker"
APACHE2_MODULES="actions alias auth_basic authn_default authn_file authz_host autoindex␣
↪dir env expires headers include info log_config logio mime mime_magic negotiation␣
↪rewrite setenvif status userdir dbd"
```

Build and start the apache server:

```
emerge --newuse apache
rc-update add apache2 default
```

Add Lance's overlay to `/etc/make.conf`:

```
PORTDIR_OVERLAY="/usr/portage/local/ramereth-overlay"
mkdir /usr/portage/local
cd /usr/portage/local
git clone git://github.com/ramereth/ramereth-overlay.git
```

Create/edit the file `/etc/portage/package.keywords` and add the following:

```
~www-misc/mirrorbrain-9999
~www-misc/mirrorbrain-2.8.1
~www-misc/mirrorbrain-2.8
~www-apache/mod_mirrorbrain-2.8.1
~www-apache/mod_form-132
~www-apache/mod_autoindex_mb-2.8.1

~dev-python/cmdln-1.1.2
~dev-python/sqlobject-0.10.4
~dev-python/formencode-1.2.1
```

Due to lack of a dependency in dev-python/cmdln, you need to do:

```
emerge unzip
```

Due to another little lack of a dependency, you also need to do:

```
emerge DateTime
```

Now, you can build mirrorbrain (and its components):

```
emerge -va mirrorbrain
```

You should get about the following output:

```
These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] dev-python/psycopg-2.0.8  USE="-debug -doc -examples -mxdatetime" 243 kB␣
↪[0]
[ebuild  N    ] www-apache/mod_form-132  0 kB [1]
[ebuild  N    ] dev-perl/HTML-Tagset-3.20  8 kB [0]
[ebuild  N    ] dev-python/setuptools-0.6_rc9  247 kB [0]
[ebuild  N    ] perl-core/Test-Simple-0.80  80 kB [0]
[ebuild  N    ] perl-core/Sys-Syslog-0.27  75 kB [0]
[ebuild  N    ] perl-core/Storable-2.18  174 kB [0]
[ebuild  N    ] dev-perl/Config-IniFiles-2.39  USE="-test" 38 kB [0]
[ebuild  N    ] dev-python/cmdln-1.1.2  86 kB [1]
[ebuild  N    ] dev-perl/Digest-MD4-1.5  29 kB [0]
[ebuild  N    ] www-apache/mod_autoindex_mb-2.8.1  302 kB [1]
[ebuild  N    ] virtual/perl-Test-Harness-3.10  0 kB [0]
[ebuild  N    ] dev-perl/Net-Daemon-0.43  28 kB [0]
```

(continues on next page)

(continued from previous page)

```
[ebuild  N    ] dev-perl/HTML-Parser-3.60  USE="-test" 86 kB [0]
[ebuild  N    ] dev-python/formencode-1.2.1  USE="-doc" 187 kB [0]
[ebuild  N    ] virtual/perl-Test-Simple-0.80  0 kB [0]
[ebuild  N    ] virtual/perl-Sys-Syslog-0.27  0 kB [0]
[ebuild  N    ] virtual/perl-Storable-2.18  0 kB [0]
[ebuild  N    ] dev-python/sqlobject-0.10.4  USE="postgres -doc -firebird -mysql -sqlite
↪" 253 kB [0]
[ebuild  N    ] dev-perl/HTML-Tree-3.23  119 kB [0]
[ebuild  N    ] dev-perl/PlRPC-0.2020-r1  18 kB [0]
[ebuild  N    ] dev-perl/DBI-1.601  484 kB [0]
[ebuild  N    ] dev-perl/DBD-Pg-1.49  144 kB [0]
[ebuild  N    ] www-apache/mod_mirrorbrain-2.8.1  USE="-memcache" 0 kB [1]
[ebuild  N    ] dev-perl/Crypt-SSLeay-0.57  121 kB [0]
[ebuild  N    ] dev-perl/libwww-perl-5.805  USE="ssl" 232 kB [0]
[ebuild  N    ] www-misc/mirrorbrain-2.8.1  0 kB [1]

Total: 27 packages (27 new), Size of downloads: 2,948 kB
Portage tree and overlays:
 [0] /usr/portage
 [1] /usr/portage/local/ramereth-overlay

Would you like to merge these packages? [Yes/No]
...
```

4.5.1 Next steps

From here, follow on with *Initial configuration steps on all platforms*.

4.6 Installing mod_asn

mod_asn is an optional extension for MirrorBrain. It is not required.

mod_asn allows a refined mirror selection upon certain network characteristics.

Without mod_asn, MirrorBrain selects an appropriate mirror by geolocation through the GeoIP database. Thus, the client gets his download from a mirror within the same country, or if that isn't possible from a mirror within the same continent at least. That's often all that you need.

With mod_asn, MirrorBrain additionally looks at the network prefix that the client's IP address is contained in, and the autonomous system (AS) that it belongs to, and (if possible) matches that to the networks that the mirrors live in. This can be useful in the following scenarios:

- You have a lot of mirrors - so many, that there is a chance that a client might have a mirror on his own network. mod_asn will find it by matching the AS and network prefix. For instance, if there's a large ISP who run a mirror, and they probably have a significant number of downloaders among their customers, it is generally desirable to send them to their ISPs mirror, instead of somewhere else.

  On the contrary, if such "network locality" does not exist, there is no reason to look up the network prefix or autonomous system and match them to the mirrors. For instance, say you have exactly one US mirror, one in India, and one in Germany; then GeoIP is good enough to assign the clients to their closest mirror simply by looking at the client's country.

- You have a certain mirror that has a lot of users on the same network, and you want to make sure that those clients all get sent to that mirror, and not just to *any* mirror in the same country (what GeoIP alone would do).
- You have a certain mirror that doesn't want to get requests from any network other than its own. Only local traffic is wanted, by policy. MirrorBrain knows which clients are "local", and you set a flag in the mirror's config (`prefixOnly` or `asOnly`) which tells MirrorBrain that the mirror should not get any other traffic. The mirrors will be happy to know about this possibility.

To install mod_asn, refer to the its documentation.

4.7 Initial configuration steps on all platforms

4.7.1 Creating a file tree

If you haven't got a file tree yet, you should create it now.

Make the directory for the file tree and fill it:

```
mkdir /srv/mytree
rsync .... /srv/mytree
```

Note that this file tree is a necessary prerequisite to running MirrorBrain, even though it intercepts the requests to those files and redirects them to mirrors.

Having said that, there *is* a way to get by without local files, which is by using the **`null-rsync`** (found in the source tree) tool instead of **`rsync`** to pull the files. **`null-rsync`** is used exactly as rsync, but it will create a pseudo file tree that requires very few local space. However, since those files are filled with zeroes (!), it is important to make sure that MirrorBrain *never* delivers content from those files. That is achieved by using the `MirrorBrainFallback` directive to define some mirrors that are *always* available and are guaranteed to have *all* those files. (The directive can be configured individually per directory in Apache config.) See the 2.11.0 release notes for details.

Note that if you *do* have the *real* files locally, you can automatically maintain cryptographic hashes of them in the database; running with pseudo files cuts on some very useful features. In addition, the local files are always available to deliver them directly, which is a good fallback behaviour for files that are not mirrored at all, files that have not arrived on any mirror just yet, and so on. Of course, you can also make sure that files are never delivered from the redirector (in other words, it redirects always).

Note: In summary: a tree with real files is required, if you want to serve any hashes, zsync, or torrents. But you can make sure that the content is always redirected. The "fake tree" that you can create with null-rsync is good *only* for pure redirection. (And Metalinks without hashes.) The server doesn't know any content then; only file path, size, mtime, nothing else.

4.7.2 Creating mirrorbrain.conf

Create a configuration file named `/etc/mirrorbrain.conf` with the content below:

```
[general]
instances = main

[main]
dbuser = mirrorbrain
dbpass = 12345
```

(continues on next page)

(continued from previous page)

```
dbdriver = postgresql
dbhost = 127.0.0.1
# optional: dbport = ...
dbname = mirrorbrain

[mirrorprobe]
# logfile = /var/log/mirrorbrain/mirrorprobe.log
# loglevel = INFO
```

Note: The database password in the above template is only a placeholder and you need to edit it: change it to the actual password, the one that you gave when you ran PostgreSQL's `createuser` command. Likewise, make sure that you picked the same username.

Set the following permissions and privileges on the file:

```
sudo chmod 0640 /etc/mirrorbrain.conf
sudo chown root:mirrorbrain /etc/mirrorbrain.conf
```

Other possible options per MirrorBrain instance are:

`scan_top_include`
: Directory names separated by spaces. Meaning: Scan only these directories, and ignore all other directories at the top level.

`scan_exclude_rsync`
: Exclude list for rsync scans (same rules as for rsyncs option `--exclude` apply). Meaning: Ignore all directories or path names that match, everywhere in the tree.

`scan_exclude`
: Exclude list for FTP scans. Meaning: Ignore all directories or path names that match, everywhere in the tree.

4.7.3 Testing the database admin tool

At this point, you should be able to type the following command without getting an error:

```
mb list
```

It'll produce no output, but exit with 0. If it gives an error, something is wrong.

Note: Do this to verify that the previous steps have been completed successfully.

Likewise, the following command should not return any error, but rather displays its usage info. If so, the installation should be quite fine:

```
mb help
```

Also, the following should work (you might have to change the path to `/usr/share/GeoIP` for your system):

```
 % geoiplookup_continent -f /var/lib/GeoIP/GeoIP.dat www.slashdot.org
NA
```

The `NA` stands for North America and indicates that the GeoIP lookup works correctly.

4.7.4 Creating some mirrors

Collect a list of mirrors (their HTTP baseurl, and their rsync or FTP baseurl for scanning). For example:

```
http://ftp.isr.ist.utl.pt/pub/MIRRORS/ftp.suse.com/projects/
rsync://ftp.isr.ist.utl.pt/suse/projects/

http://ftp.kddilabs.jp/Linux/distributions/ftp.suse.com/projects/
rsync://ftp.kddilabs.jp/suse/projects/
```

Now you need to enter the mirrors into the database; it could be done using the "mb" mirrorbrain tool. (See 'mb help new' for full option list.):

```
mb new ftp.isr.ist.utl.pt \
       --http http://ftp.isr.ist.utl.pt/pub/MIRRORS/ftp.suse.com/projects/ \
       --rsync rsync://ftp.isr.ist.utl.pt/suse/projects/

mb new ftp.kddilabs.jp \
       --http http://ftp.kddilabs.jp/Linux/distributions/ftp.suse.com/projects/ \
       --rsync rsync://ftp.kddilabs.jp/suse/projects/
```

The tool automatically figures out the GeoIP location of each mirror by itself. But you could also specify them on the commandline.

If you want to edit a mirror later, use:

```
mb edit <identifier>
```

To simply display a mirror, you could use 'mb show kddi', for instance.

Finally, each mirror needs to be scanned and enabled:

```
mb scan --enable <identifier>
```

See the output of **mb help** for more commands. Refer to *Maintaining the mirror database* for a full reference documentation to the **mb** tool.

4.7.5 Setting up required cron jobs

Setting up mirror monitoring

Mirror monitoring needs to be set up to run automatically. Put this into `/etc/crontab`:

The following cron job is needed to check which mirrors are reachable. This command is responsible for checking the mirrors in short intervals, and marking them online/offline in the database:

```
* * * * *                    mirrorbrain   mirrorprobe
```

Setting up mirror scanning

Configure mirror scanning:

```
45 * * * *                  mirrorbrain   mb scan --quiet --jobs 4 --all
```

Use more parallel scanners (`-j|--jobs ...`) if you have a beefy machine.

The `--quiet` option can be used twice (e.g. as `-qq`), which will totally silence the scanner, except for error messages. This means that you get a mail only when there is something wrong.

Maintenance

Another cron job is useful to remove unreferenced files from the database:

```
# Monday: database clean-up day...
30 1 * * mon                mirrorbrain   mb db vacuum
```

Keeping the GeoIP database uptodate

The GeoIP database is changed at least once a month, so a new copy should be downloaded regularly:

```
# update GeoIP database on Mondays
31 2 * * mon                root  sleep $(($RANDOM/1024)); /usr/bin/geoip-lite-update
```

(The ‘sleep’ is there so you can copy the line, don’t need to adjust the time, and still the GeoIP servers will not get a lot of simultaneous hits at exactly the same time. That’s all.)

4.7.6 Testing

TODO: describe how to test that the install was successful (When testing, consider any excludes that you configured, and which might introduce confusion.)

- Many HTTP clients can be used for testing, but cURL is a most helpful tool for that. Here are some examples.

  Showy the HTTP response code and the Location header pointing to the new location:

  ```
  curl -sI <url>
  ```

  Display the metalink:

  ```
  curl -s <url>.metalink
  ```

  Show a HTML list with the available mirrors:

  ```
  curl -s <url>?mirrorlist
  ```

4.7.7 Setting up logging

You may want to log more details than Apache normally logs into the access_log file. You can define a new log format that gives you an access_log, with details from MirrorBrain added:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" \
want:%{WANT}e give:%{GIVE}e r:%{MB_REALM}e %{X-MirrorBrain-Mirror}o \
%{MB_CONTINENT_CODE}e:%{MB_COUNTRY_CODE}e ASN:%{ASN}e P:%{PFX}e \
size:%{MB_FILESIZE}e %{Range}i" combined_redirect
```

This defines a new log format called “combined_redirect”, which you can use in your virtual hosts with the CustomLog directive.

Instead of:

```
CustomLog /var/log/apache2/myhost/access_log combined
```

you would use:

```
CustomLog /var/log/apache2/myhost/access_log combined_redirect
```

4.7.8 Creating hashes

First, add some configuration (example):

```
MirrorBrainMetalinkPublisher "openSUSE" http://download.opensuse.org
```

You need to create a directory where to store the hashes. For instance, `/srv/hashes/srv/opensuse`. Note that the full pathname to the filetree (`/srv/opensuse`) is part of this target path.

Make the directory owned by the `mirrorbrain` user.

Now, create the hashes with the following command. It is best run as unprivileged user (`mirrorbrain`):

```
mb makehashes /srv/opensuse -t /srv/hashes/srv/opensuse
```

Add the hashing command to /etc/crontab to be run every few hours. Alternatively, run it after changes in the file tree happen, coupled to some trigger etc.

Note: The path names used here need to match the `DocumentRoot` in Apache’s virtual host setup.

(This command was called `metalink-hasher` in previous releases of MirrorBrain.)

4.7.9 Optional things you might want

- further things that you might want to configure:
  - mod_autoindex_mb, a replacement for the standard module mod_autoindex:

    ```
    a2dismod autoindex
    a2enmod autoindex_mb
    Add IndexOptions Mirrorlist
    ```

(continues on next page)

(continued from previous page)

```
# or IndexOptions +Mirrorlist, depending on your config (if you have options
# that are inherited from above in the tree, and you just want to add this one␣
↪here)
```

- add a link to a CSS stylesheet for mirror lists:

```
MirrorBrainMirrorlistStylesheet "http://static.opensuse.org/css/mirrorbrain.css"
```

and for the autoindex:

```
IndexStyleSheet "http://static.opensuse.org/css/mirrorbrain.css"
```

4.7.10 Configuring GeoIP

Note: It is better to use the larger GeoLiteCity database, instead of the minimal GeoIP database that contains only country information. With the more detailed info in the former database, a better mirror selection is achieved in many cases.

Edit /etc/apache2/conf.d/mod_geoip.conf:

```
<IfModule mod_geoip.c>
   GeoIPEnable On
   GeoIPDBFile /var/lib/GeoIP/GeoLiteCity.dat.updated
   #GeoIPOutput [Notes|Env|All]
   GeoIPOutput Env
</IfModule>
```

(Change GeoIPOutput All to GeoIPOutput Env)

Note that a caching mode like MMapCache needs to be used, when Apache runs with the worker MPM.In this case, use:

```
<IfModule mod_geoip.c>
   GeoIPEnable On
   GeoIPDBFile /var/lib/GeoIP/GeoLiteCity.dat.updated MMapCache
   GeoIPOutput Env
</IfModule>
```

Seting up automatic updates of the GeoIP database

New versions of the GeoIP database are released each month. You can set up a cron job to automatically fetch new updates as follows. If you do that, make sure to set the GeoIPDBFile path (see above) to `/var/lib/GeoIP/GeoLiteCity.dat.updated`:

```
# update GeoIP database on Mondays
31 2 * * mon   root    sleep $(($RANDOM/1024)); /usr/bin/geoip-lite-update
```

4.7.11 Creating a virtual host

Maybe create a DNS alias for your web host, if needed.

Note: A complete reference of all Apache directives can be found here.

The following snippet would create a new site as virtual host:

```
sudo sh -c "cat > /etc/apache2/sites-available/mirrorbrain << EOF
<VirtualHost 127.0.0.1>
    ServerName mirrors.example.org
    ServerAdmin webmaster@example.org
    DocumentRoot /var/www/downloads
    ErrorLog     /var/log/apache2/mirrors.example.org/error.log
    CustomLog    /var/log/apache2/mirrors.example.org/access.log combined
    <Directory /var/www/downloads>
        MirrorBrainEngine On
        MirrorBrainDebug Off
        FormGET On
        MirrorBrainHandleHEADRequestLocally Off
        MirrorBrainMinSize 2048
        MirrorBrainExcludeUserAgent rpm/4.4.2*
        MirrorBrainExcludeUserAgent *APT-HTTP*
        MirrorBrainExcludeMimeType application/pgp-keys
        Options FollowSymLinks Indexes
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
EOF
"
```

Another example:

```
<VirtualHost your.host.name:80>
    ServerName samba.mirrorbrain.org

    ServerAdmin webmaster@example.org

    DocumentRoot /srv/samba/pub/projects

    ErrorLog     /var/log/apache/samba.mirrorbrain.org/logs/error_log
    CustomLog    /var/log/apache/samba.mirrorbrain.org/logs/access_log combined

    <Directory /srv/samba/pub/projects>
        MirrorBrainEngine On
        MirrorBrainDebug Off
        FormGET On
        MirrorBrainHandleHEADRequestLocally Off
        MirrorBrainMinSize 2048
        MirrorBrainExcludeUserAgent rpm/4.4.2*
        MirrorBrainExcludeUserAgent *APT-HTTP*
```

(continues on next page)

(continued from previous page)

```
        MirrorBrainExcludeMimeType application/pgp-keys

        Options FollowSymLinks Indexes
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>

</VirtualHost>
```

Make the log directory for the virtual host:

```
sudo mkdir /var/log/apache2/mirrors.example.org/
```

Enable the site:

```
sudo a2ensite mirrorbrain
```

Restart Apache, best while watching the error log:

```
sudo tail -f /var/log/apache2/error.log &
sudo /etc/init.d/apache2 restart
```

4.8 Troubleshooting

If Apache doesn't start, or anything else seems wrong, make sure to check Apache's error_log. It usually points into the right direction.

A general note about Apache configuration which might be in order. With most config directives, it is important to pay attention where to put them - the order does not matter, but the context does. There is the concept of directory contexts and vhost contexts, which must not be overlooked. Things can be "global", or inside a <VirtualHost> container, or within a <Directory> container.

This matters because Apache applies the config recursively onto subdirectories, and for each request it does a "merge" of possibly overlapping directives. Settings in vhost context are merged only when the server forks, while settings in directory context are merged for each request. This is also the reason why some of mod_asn's config directives are programmed to be used in one or the other context, for performance reasons.

The install docs you are reading attempt to always point out in which context the directives belong.

Note: To get help, please subscribe to the mirrorbrain mailing list, see http://mirrorbrain.org/communication . Questions can be answered there, and all kind of feedback is appreciated.

Note: Bugs should be reported via http://mirrorbrain.org/issues/ or via the mailing list.

CHAPTER

FIVE

MAINTAINING THE MIRROR DATABASE

5.1 Concepts – the mb command

mb is a commandline tool to do maintain the mirror database, create mirrors, edit them, work with files and other tasks.

It has several subcommands, and it is typically used in one the following forms:

```
mb <command>
mb <command> <identifier>
```

A typical example would be:

```
mb edit opensuse.uib.no
```

Note the first argument (after `edit`), which is the *mirror identifier*. It serves as a name that uniquely identifies a single mirror. It can be useful if these identifiers are memorizable by a human.

For all **mb** commands where a mirror (or several) needs to be specified, you can abbreviate the identifier by typing part of it. For instance, instead of:

```
mb show opensuse.uib.no
```

you could just type:

```
mb show uib
```

as long as `uib` is uniquely identifying a mirrors among the others.

The **mb** command is extensible. See the developers documentation for instructions. (To be written yet.) .. TODO: add reference

5.1.1 Built-in help

mb has reference documentation built-in. If you just call **mb** or **mb -h** or **mb help**, it will print out the list of known subcommands:

```
 % mb
Usage:
    mb COMMAND [ARGS...]
    mb help [COMMAND]

Options:
```

(continues on next page)

(continued from previous page)

```
    --version           show program's version number and exit
    -h, --help          show this help message and exit
    -d, --debug         print info useful for debugging
    -b BRAIN_INSTANCE, --brain-instance=BRAIN_INSTANCE
                        the mirrorbrain instance to use. Corresponds to a
                        section in /etc/mirrorbrain.conf which is named the
                        same. Can also specified via environment variable MB.

Commands:
    commentadd     add a comment about a mirror
    db (vacuum)    perform database maintenance
    delete         delete a mirror from the database
    dirs           show directories that are in the database
    disable        disable a mirror
    edit           edit a new mirror entry in $EDITOR
    enable         enable a mirror
    export         export the mirror list as text file
    file           operations on files: ls/rm/add
    help (?)       give detailed help on a specific sub-command
    instances      list all configured mirrorbrain instances
    iplookup       lookup stuff about an IP address
    list           list mirrors
    markers        show or edit marker files
    mirrorlist     generate a mirror list
    new            insert a new mirror into the database
    probefile      list mirrors on which a given file is present by probing...
    rename         rename a mirror's identifier
    scan           scan mirrors
    score          show or change the score of a mirror
    show           show a mirror entry
    test           test if a mirror is working
    update         update mirrors network data in the database
```

By typing **mb <command> -h** or **mb help <command>**, help for the individual command will be printed:

```
 % mb help list
list: list mirrors

Usage:
    mb list [IDENTIFIER]
Options:
    -h, --help          show this help message and exit
    -r XY               show only mirrors whose region matches XY (possible
                        values: sa,na,oc,af,as,eu)
    -c XY               show only mirrors whose country matches XY
    -a, --show-disabled
                        do not hide disabled mirrors
    --disabled          show only disabled mirrors
    --prio              also display priorities
    --asn               also display the AS
    --prefix            also display the network prefix
    --region            also display the region
```

(continues on next page)

(continued from previous page)

```
    --country           also display the country
    --other-countries   also display other countries that a mirror is
                        configured to handle
```

5.2 Creating a new mirror

As necessary ingredient, there need to be mirror servers. They need to serve content via HTTP or FTP. To be scanned, they need to run rsync, FTP or HTTP. rsync is most efficient for this. FTP is second choice. At last, HTTP may be used, however it'll work only if the HTTP server provides a reasonable "standard" directory index.

To make a new mirror known to the database, you use the **mb** command, specifically the **mb new** subcommand. An example would be the following:

```
mb new opensuse.uib.no -H http://opensuse.uib.no/ \
                       -F ftp://opensuse.uib.no/pub/Linux/Distributions/opensuse/ \
                       -R rsync://opensuse.uib.no/opensuse-full/
```

This creates a new entry in the mirror database with the data provided on the commandline.

Because providing a lot of data on the commandline can be tiresome, and incremental changes are often needed to get the data right, there is a command to edit the data later: **mb edit**.

A new mirror created this way is disabled in the beginning, because it needs to be scanned first before it can be useful.

5.3 Enabling mirror

Enabling a mirror, or more correctly *enabling redirections* to a mirror, can be done with the command **mb enable**.

Before doing this for the first time, the mirror needs to be scanned to be useful; see below (*Scanning mirrors*).

Another way to enable a mirror is to edit its database record directly (see below, where this is explained).

5.4 Disabling a mirror

Using the **mb disable** command, a mirror can be disabled, and MirrorBrain will immediately stop to send requests to it.

Another way to disable a mirror is to use **mb edit** to edit its database record, and changing the `enabled` field to `False` or `0`. At the same time, a comment about the reason could be left in the `comment` field.

Disabled mirrors are not scanned. Thus, it is usually advisable to scan a mirror before reenabling it after inactivity for prolonged time, using **mb scan -e**.

A mirror will also effectively be disabled if the `score` is set to `0`.

5.5 Deleting a mirror

A mirror is deleted with the **mb delete** command. This command is an exception of the rule of abbreviating mirror identifiers; here, the full and exact identifier of the mirror to be deleted must be specified. This is to prevent typos.

A deleted mirror is permanently pruned from the database upon completion of the command.

5.6 Displaying details about a mirror

mb show will print out the metadata of a mirror. Example:

```
 % mb show uib
identifier     : opensuse.uib.no
operatorName   : UiB - University of Bergen, IT services
operatorUrl    : http://it.uib.no/
baseurl        : http://opensuse.uib.no/
baseurlFtp     : ftp://opensuse.uib.no/pub/Linux/Distributions/opensuse/opensuse/
baseurlRsync   : rsync://opensuse.uib.no/opensuse-full/
region         : eu
country        : no
asn            : 224
prefix         : 129.177.0.0/16
regionOnly     : False
countryOnly    : False
asOnly         : False
prefixOnly     : False
otherCountries :
fileMaxsize    : 0
publicNotes    :
score          : 100
enabled        : True
statusBaseurl  : True
admin          : X, Y, ...
adminEmail     : mail@example.com
---------- comments ----------
Added - Wed May  6 14:36:10 2009

*** scanned and enabled at Wed May  6 14:47:56 2009.

Gave stage access.
poeml, Mon May 11 16:11:56 CEST 2009

Adjusted FTP URL after they switched to stage. (appended "opensuse").
rsync down at the moment.
poeml, Mon May 11 17:18:06 CEST 2009
---------- comments ----------
```

5.7 A mirror record explained

Field	Explanation
`identifier`	This is the unique identifier of the mirror server. In the table shown by mb edit, this is the only field that cannot be edited. To rename an identifier, you can use the **`mb rename`** command.
`operatorName`	The realname of the mirror operator. This could be a person, an the organization running the mirror, or a sponsor. If the mirror list is exposed in some way, this field could be used to give the operator some visibility. Otherwise, it is of no significance than for your information.
`operatorUrl`	A contact or informative URL.
`baseurl`	The root HTTP URL of the mirrored file tree on the mirror. Used by the redirector to redirect requests via HTTP. If a mirror doesn't offer HTTP, but only FTP, an FTP URL can be entered here as well.
`baseurlFtp`	The root FTP URL of the mirrored file tree on the mirror. Used by the scanner to retrieve the file list - if rsync isn't available..
`baseurlRsync`	The root rsync URL used by the scanner to find the files via rsync. It's possible to use URLs with credentials, like `rsync://<username>:<password>@<hostname>/module`. rsync is the preferred method of scanning, so it is beneficial if rsync access exists. If it doesn't, the scanner falls back to FTP or HTTP.
`region`	The region code specifying the continent the mirror server is located in. See also `regionOnly`. If you create a new mirror, **`mb new`** tries to fill in this field and the following field for you; it's possible to edit it later, though.
`country`	The country code for the server. See also `countryOnly`.
`asn`	This is optional and is a number of the autonomous system the mirror is located in. It may serve as a more specific "network location" than the country, and is filled in automatically when a mirror is created. If you don't use the autonomous system database together with MirrorBrain, the value will be zero and will be ignored by MirrorBrain. It is not strictly needed. It can also be edited manually, or updated via **`mb update --asn <identifier>`** from looked up data. *Only meaningful if MirrorBrain is used together with mod_asn.*
`prefix`	Same as `asn`, this value is optional, and if present, it is used for a possibly finer-grained mirror selection. It is filled in automatically, and can be edited like asn. Use **`mb update --prefix <identifier>`** to fill in data from a routing table lookup.
`regionOnly`	If true, only clients from the same region (continent) as the mirror are redirected to this mirror.
`countryOnly`	If true, only clients from the same country as the mirror are redirected to this mirror.
`asOnly`	If true, the mirror will only get requests from clients that are located within the same network autonomous system (using the value in `asn`).
	If true, the mirror will only get requests from clients

5.8 Editing a mirror

A mirror (in the mirror database) can be edited with the **mb edit** command.

The command will bring up an editor with the mirror's metadata. The `EDITOR` and `VISUAL` environmental variable is respected, and the editor defaults to **vim**.

For fields where a Boolean is expected, you can type the value (while editing) in the form of 0/1 instead of true/false (shorter to type).

When you save the text and close the editor, you'll be asked whether to save the data to the database.

5.9 Editing a mirrors network location

There are some fields in the mirror record, for which manual editing doesn't make so much sense. These are:

- country,
- region,
- autonomous system number,
- network prefix,
- geographical coordinates.

When a mirror is created (using **mb new** *), then all these fields are automatically filled in.* This requires a working DNS lookup and a GeoIP database.

The lookup of the autonomous system number and network prefix require mod_asn to be configured.

The geographical coordinates require the GeoIP database to be the GeoIP city (lite) version. The smaller database versions don't contain the coordinates.

The data can be updated later with the **mb update** command. Regularly running this command (say, once a month) is a good idea because the data sometimes might change over time. However, this also means that manual edits will be overwritten.

To update all network data for all mirrors, simply run:

```
% mb update -A --all-mirrors
```

The command can also be used for individual mirrors, and to update only some data:

```
 % mb update --coordinates --asn --prefix ftp5
updating geographical coordinates for ftp5.gwdg.de (0.000 0.000 -> 53.083 8.8)
```

Or it can be applied to all active mirrors:

```
 % mb update --coordinates --asn --prefix
updating geographical coordinates for ring.yamanashi.ac.jp (0.000 0.000 -> 36.0 138.0)
updating network prefix for mirror.lupaworld.com (122.224.0.0/12 -> 115.224.0.0/12)
[...]
```

5.10 Listing mirrors

mb list lists mirrors, with less or more details. In its simplest form, the command will simply print all identifiers of enabled mirrors. **mb list -a** includes also the disabled mirrors.

More useful is to add filters, or display more data.

Examples of filtering by country code (here: Bulgaria, `bg`):

```
% mb list -c bg
mirrors.netbg.com
bgadmin.com
```

Example of filtering by region (here: Oceania, `oc`), and also displaying the value of the `otherCountries` field for each mirror:

```
% mb list -r oc --other-countries
ftp.iinet.net.au               nz
mirror.aarnet.edu.au           nz
mirror.pacific.net.au          nz
mirror.internode.on.net        nz
mirror.3fl.net.au              nz
netspace.net.au                nz
optusnet.com.au                nz
```

Example of listing all mirrors in Portugal and showing their `score` (their priority):

```
% mb list -c pt --prio
lisa.gov.pt                    100
ftp.isr.ist.utl.pt              50
uminho.pt                       50
ftp.nux.ipb.pt                   3
```

Showing priority, network prefix and autonomous system of Chinese mirrors:

```
% mb list -c cn --prio --as --prefix
mirror.lupaworld.com           100  4134 122.224.0.0/12
lizardsource.cn                 30  9389 211.166.8.0/21
lcuc.org.cn                    100 17816 218.249.128.0/17
```

When *not* filtering the output, the `--country` and `--region` commandline options are useful, because they add that data into the output. An example would be listing all mirrors with the command **mb list --prio --as --prefix --country --region**.

5.11 Scanning mirrors

Mirrors need to be scanned for their file lists. This is done with the **mb scan** command. The program will try rsync, if available, FTP if not, or HTTP if it's the only option.

An individual mirror can be scanned like this:

```
% mb scan roxen
Fri Jul 31 21:31:50 2009 roxen.integrity.hu: starting
```

(continues on next page)

(continued from previous page)

```
Fri Jul 31 21:31:51 2009 roxen.integrity.hu: total files before scan: 17248
Fri Jul 31 21:31:59 2009 roxen.integrity.hu: scanned 17248 files (1935/s) in 8s
Fri Jul 31 21:31:59 2009 roxen.integrity.hu: files to be purged: 0
Fri Jul 31 21:32:00 2009 roxen.integrity.hu: total files after scan: 17248
Fri Jul 31 21:32:00 2009 roxen.integrity.hu: purged old files in 1s.
Fri Jul 31 21:32:00 2009 roxen.integrity.hu: done.
Completed in 9 seconds
```

After creation of a new mirror, it is disabled first. A typical workflow would be to scan it, after creating it, and then enabling redirection. **mb scan** command can be used with the `-e`/`--enable` option to make this happen. If the scan went successfully, the mirror will be enabled afterwards:

```
 % mb scan -e tuwien
Fri Jul 31 21:50:45 2009 gd.tuwien.ac.at: starting
Fri Jul 31 21:50:45 2009 gd.tuwien.ac.at: total files before scan: 712
Fri Jul 31 21:50:46 2009 gd.tuwien.ac.at: scanned 712 files (511/s) in 1s
Fri Jul 31 21:50:46 2009 gd.tuwien.ac.at: files to be purged: 0
Fri Jul 31 21:50:46 2009 gd.tuwien.ac.at: total files after scan: 712
Fri Jul 31 21:50:46 2009 gd.tuwien.ac.at: purged old files in 0s.
gd.tuwien.ac.at: now enabled.
Fri Jul 31 21:50:46 2009 gd.tuwien.ac.at: done.
Completed in 1 seconds
```

To scan all enabled mirrors in parallel, you would use `-j`/`--jobs=N` option to specify the number of scanners to start in parallel, and the `-a`/`--all` option:

```
% mb scan -j 16 -a
```

This is likely what you would configure to be done periodically by cron.

To scan only a subdirectory on the mirrors, the `-d` option can be used. This can be useful when it is known that content has been added or removed in particular places of large trees, in the following example shown with a single mirror only:

```
 % mb scan -d repositories/Apache ftp5
Checking for existance of 'repositories/Apache' directory
.
Scheduling scan on:
    ftp5.gwdg.de
Completed in 0 seconds
Fri Jul 31 21:41:37 2009 ftp5.gwdg.de: starting
Fri Jul 31 21:41:38 2009 ftp5.gwdg.de: files in 'repositories/Apache' before scan: 780
Fri Jul 31 21:41:40 2009 ftp5.gwdg.de: scanned 780 files (636/s) in 1s
Fri Jul 31 21:41:40 2009 ftp5.gwdg.de: files to be purged: 0
Fri Jul 31 21:41:42 2009 ftp5.gwdg.de: total files after scan: 760122
Fri Jul 31 21:41:42 2009 ftp5.gwdg.de: purged old files in 2s.
Fri Jul 31 21:41:42 2009 ftp5.gwdg.de: done.
Completed in 4 seconds
```

For debugging purposes, the `-v` option is useful. It can be repeated several times to enable more output.

5.12 Listing files

Files known to the database can be listed with the **mb file ls** command. When specifying a path name, the leading slash is optional and not relevant. (Internally, the filenames are stored without.)

Example:

```
 % mb file ls /distribution/11.1/repo/oss/suse/ppc/tcsh-6.15.00-93.3.ppc.rpm
as th  100 ok       ok   mirror.in.th
eu at  100 disabled dead tugraz.at
eu at  100 ok       ok   gd.tuwien.ac.at
eu de  100 ok       ok   ftp5.gwdg.de
eu hu  100 ok       ok   roxen.integrity.hu
```

Globbing can be used. Then, to get more than a list or mirrors, but also the filenames, the -u/--url option is useful:

```
 % mb file ls \*.iso -u
as th  100 ok       ok   mirror.in.th                      http://mirror.in.th/opensuse/
↪ppc/factory/iso/openSUSE-NET-ppc-Build0137-Media.iso
as th  100 ok       ok   mirror.in.th                      http://mirror.in.th/opensuse/
↪ppc/factory/iso/openSUSE-Factory-NET-ppc-Build0051-Media.iso
as th  100 ok       ok   mirror.in.th                      http://mirror.in.th/opensuse/
↪ppc/factory/iso/openSUSE-Factory-NET-ppc-Build0059-Media.iso
as th  100 ok       ok   mirror.in.th                      http://mirror.in.th/opensuse/
↪ppc/factory/iso/openSUSE-NET-ppc-Build0116-Media.iso
eu de  100 ok       ok   ftp5.gwdg.de                      http://ftp5.gwdg.de/pub/
↪opensuse/ppc/factory/iso/openSUSE-NET-ppc-Build0179-Media.iso
eu hu  100 ok       ok   roxen.integrity.hu                http://roxen.integrity.hu/pub/
↪opensuse/ppc/factory/iso/openSUSE-NET-ppc-Build0179-Media.iso
```

In addition to just listing what's known to the database, the command can also do probing. The number is the HTTP return code (200 for OK):

```
 % mb file ls /distribution/11.1/repo/oss/suse/ppc/tcsh-6.15.00-93.3.ppc.rpm --probe
.....
as th  100 ok       ok   mirror.in.th                      200
eu at  100 disabled dead tugraz.at
eu at  100 ok       ok   gd.tuwien.ac.at                   200
eu de  100 ok       ok   ftp5.gwdg.de                      200
eu hu  100 ok       ok   roxen.integrity.hu                200
```

When used with probing, there is the additional option to actually download the content and display a checksum of what was returned:

```
 % mb file ls --probe /distribution/11.1/repo/oss/suse/ppc/tcsh-6.15.00-93.3.ppc.rpm --
↪md5
.....
as th  100 ok       ok   mirror.in.th                      200␣
↪50dc50b20a97783a51ff402359456e3a
eu at  100 disabled dead tugraz.at
eu at  100 ok       ok   gd.tuwien.ac.at                   200␣
↪50dc50b20a97783a51ff402359456e3a
eu de  100 ok       ok   ftp5.gwdg.de                      200␣
↪50dc50b20a97783a51ff402359456e3a
```

(continues on next page)

(continued from previous page)

```
eu hu  100 ok       ok   roxen.integrity.hu              200␣
↪50dc50b20a97783a51ff402359456e3a
```

To be usable with lots of mirrors, the probing is done in parallel.

The **mb file** command can also be used as **mb file add** and **mb file rm** to manipulate the database. See the help output of the command for details.

5.13 Exporting mirror lists

The **mb export** command can export data from the mirror database in several different formats, for different purposes.

5.13.1 Exporting in mirmon format

mirmon is a program written by Henk P. Penning which monitors the status of mirrors. The format "mirmon" exports a list of mirrors in a text format that can be read by mirmon.

With this, it is straighforward to deploy mirmon and automate it to use the mirrors from the database. Thus, no separate list of mirrors needs to be maintained for it.

The command `mb export --format=mirmon` generates the list that mirmon needs, which basically looks like this:

```
 % mb export --format=mirmon | head
de      http://ftp-stud.fht-esslingen.de/pub/Mirrors/ftp.opensuse.org/  <...@...>
de      ftp://ftp-stud.fht-esslingen.de/pub/Mirrors/ftp.opensuse.org/   <...@...>
de      rsync://ftp-stud.fht-esslingen.de/opensuse/     <...@...>
us      http://mirror.anl.gov/pub/opensuse/opensuse/    <...@...>
us      ftp://mirror.anl.gov/pub/opensuse/opensuse/     <...@...>
us      rsync://mirror.anl.gov/opensuse/opensuse/       <...@...>
...
```

To give a full example, here's how the actual mirmon config file would look like. Note the `mirror_list` line which pulls the generated list in:

```
project_name example.org
project_url http://www.example.org/mirrors/
mirror_list /usr/bin/mb export --format=mirmon |
web_page /var/www/example.org/mirmon/index.html
icons icons
probe /usr/bin/wget -q -O - -T %TIMEOUT% -t 1 %URL%timestamp.txt
state /home/mirrorbrain/mirmon/state
countries /usr/local/mirmon-2.3/countries.list
project_logo http://www.example.org/images/logo.gif
list_style plain
timeout 20
```

The cron job to create the list and run mirmon would look like this:

```
30 * * * *   mirrorbrain    perl /usr/local/mirmon-2.3/mirmon -q -get update -c /etc/
↪mirmon.conf
```

Note: when mirmon is run for the first time, the state file needs to be touched, or the script will not run.

The icons which are included in the resulting HTML page need to made available by Apache:

```
Alias /mirmon/icons /usr/local/mirmon-2.3/icons
<Directory /usr/local/mirmon-2.3/icons>
    Options None
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Further tips:

1) If your mirmon is configured with `list_style apache` instead of `list_style plain`, a different mirror list format is needed; use **`mb export`** with the `mb export --format=mirmon-apache` option then.
2) If you prefer to run **`mb export`** under a different user id than mirmon, you can write the mirror list to an intermediate file, and configure mirmon to use the file like this:

   ```
   mirror_list /path/to/mirmon/mirrorlist-export
   ```

5.13.2 Exporting to a Version Control System (VCS)

Exporting data in text format is a dead easy way to keep a history of changes that happen in the mirror database — and mail them around, so everybody involved is kept updated. At the same time, it serves archival purposes.

The idea is to export snapshots of the data in text format. The resulting files are put into a standard version control system, and standard post-commit hook scripts can be used to trigger certain actions (e.g. email).

The resulting archive of changes is all human-readable (much more useful than raw database backups). The changes can actually be mailed around in the form of a diff, showing some context.

A different way to implement a notification system for mirror changes would be to notify about each and every change done to the database — however, often changes have to be done incrementally and this would be a noisy method when working on a mirror's configuration.

Instead, an hourly snapshot is normally sufficient to keep others informed, and shouldn't be too noisy.

Subversion is the only version control system supported at the moment, but should hopefully be ubiquitous enough.

To set this up, first a repository needs to be created:

```
doozer:~ # su - mirrorbrain
mirrorbrain@doozer:~> svnadmin create mirrors-svn-repos
mirrorbrain@doozer:~> svn co file://$PWD/mirrors-svn-repos mirrors-svn
Checked out revision 0.
mirrorbrain@doozer:~>
```

Then, set up a cron job to run every hour, calling **`mb export`** with the `--format=vcs` and the `--commit=svn` options. The latter automatically runs `svn commit` after the export (taking into account files that have been deleted, or occur for the first time):

```
 # export mirrordb contents to SVN and send commit mails
7 * * * *      mirrorbrain   mb export --format vcs --target-dir ~/mirrors-svn --
↪commit=svn
```

Finally, the post-commit hook script is missing, which takes care of sending mails. Create and edit it as follows:

```
mirrorbrain@doozer:~> touch mirrors-svn-repos/hooks/post-commit
mirrorbrain@doozer:~> chmod +x mirrors-svn-repos/hooks/post-commit
mirrorbrain@doozer:~> vi mirrors-svn-repos/hooks/post-commit

#!/bin/sh
REPOS="$1"
REV="$2"
/usr/share/subversion/tools/hook-scripts/mailer/mailer.py commit "$REPOS" "$REV" /etc/
↪mailer.conf
```

The path to the **`mailer.py`** script likely needs adjustment. The configuration (`/etc/mailer.conf`) could look like this:

```
[general]
mail_command = /usr/sbin/sendmail

[defaults]
diff = /usr/bin/diff -u -L %(label_from)s -L %(label_to)s %(from)s %(to)s
generate_diffs = add copy modify
show_nonmatching_paths = yes

[mirrordb]
for_repos = /home/mirrorbrain/mirrors-svn-repos
from_addr = mirrorbrain@...
to_addr = admin@foo bar@...
commit_subject_prefix = [mirrordb]
propchange_subject_prefix = [mirrordb]
```

5.13.3 Exporting in PostgreSQL format

The format "postgresql" creates SQL INSERT statements that can be run on a PostgreSQL database. This can e.g. be used to migrate the data into another database.

The resulting dump could be loaded into a mirrorbrain instance like this:

```
mb db shell < db.dump
```

5.13.4 Exporting in Django format

This is experimental stuff — intended for hacking on the Django web framework. Data is exported in the form of Django ORM objects, and the export routine will very likely need modification for particular purposes. The existing code has been used to experiment with. Get in contact if you are interested in hacking on this!

5.14 Performing database maintenance

The **mb db** command offers some helpful functionality regarding database maintenance. It has several subcommands.

5.14.1 Regular cleanups with `mb db vacuum`

This command cleans up unreferenced files from the mirror database.

This should be done once a week for a busy file tree. Otherwise it should be rarely needed, but can possibly improve performance if it is able to shrink the database.

When called with the `-n` option, only the number of files to be cleaned up is printed, so it's purely for information. No cleanup is performed.

The recommended cron job looks like this:

```
# Monday: database clean-up day...
30 1 * * mon              mirrorbrain   mb db vacuum
```

Note: This functionality is not to be confused with the PostgreSQL-internal vacuuming, which typically happens automatic these days (8.x), but was a manual process at some time in the past.

5.14.2 Database shell with `mb db shell`

With this command, you can conveniently open a database shell:

```
 % mb db shell
psql (8.4.1)
Type "help" for help.

mb_opensuse=>
```

. . . ready to enter commands in psql, the PostgreSQL interactive terminal.

5.14.3 Database size info with `mb db size`

The command **mb db size** prints the size of each database relation. (In PostgreSQL speak, a *relation* is a table or an index.) This provides insight for appropriate database tuning and planning. Here's an example:

```
 % mb db sizes
Size(MB) Relation
464.5    filearr
532.9    filearr_path_key
 74.3    filearr_pkey
 23.8    pfx2asn
```

(continues on next page)

(continued from previous page)

```
 30.1    pfx2asn_pfx_key
 19.9    pfx2asn_pkey
  0.0    pg_foreign_server
  0.0    pg_foreign_server_name_index
  0.0    pg_foreign_server_oid_index
  0.0    pg_user_mapping_user_server_index
  0.2    server
  0.0    server_enabled_status_baseurl_score_key
  0.0    server_identifier_key
  0.0    server_pkey
  0.0    sql_sizing_profiles
Total: 1145.9
```

This example shows a really, really large database, containing nearly 3 millions (!) of files. It uses a good gigabyte of disk space.

`filearr` contains the file names and associations to the mirrors. `filearr_path_key` is the index on the file names. `filearr_pkey` is the primary key. These will be the largest things in a database filled with millions of files.

The `pfx*` relations are only present when mod_asn is installed. The size they use is always the same.

CHAPTER

SIX

FURTHER CONFIGURING MIRRORBRAIN

This chapter contains notes about important extra things that you may want to configure, and also some more exotic configurations.

For basic configuration, please refer to the installation section (*Initial configuration steps on all platforms*).

6.1 Generating Torrents

From the hashes generated with **`mb makehashes`**, MirrorBrain can generate not only Metalinks, but also Torrents. The required chunked hashes are the same.

The generation is triggered by appending `.torrent` to an URL.

The torrents have the following properties:

- They include (a selection of closest) mirror URLs, for web seeding. They are included both in a `url-list` field as well as a `sources` field. Only a selection of mirrors is included, because there is a chance that some clients will just pick a random mirror from the list. Thus, only the mirrors that are closest to a client are included.
- Some clients (at least the original BitTorrent client) can have difficulties to grok modern fields that they do not know. (They might error out and claim invalid bencoding, for instance.) Putting some keys behind, not before, the info dictionary seems to help. Therefore, `url-list` and `sources` are included at the end.
- A DHT "nodes" list is included in Torrents (see issue 49). It needs to be configured with the `MirrorBrainDHTNode` Apache configuration directive.
- They include the MD5, SHA1 and SHA256 hash into the info dictionary.
- If a torrent is requested, but data needed for it is missing, a 404 is returned.
- Multiple trackers can be included. See below.
- The BitTorrent Info Hash (`btih`), which uniquely identifies a torrent (by an SHA1 hash over the bencoded `info` dict) is shown in the mirror list (metadata view). The info hash can also be requested separately from the server by appending `.btih` to an URL.
- Hashes that require Base32 encoding are currently not included. Base32 encoding could be added, of course, if somebody is willing to do this.

6.1.1 Performance considerations

The piece length is configurable by a parameter in `/etc/mirrorbrain.conf`. The default is `chunk_size = 262144` bytes (see `mb.hashes`). The size has a direct relation to the space that the hashes occupy in the database. To find out how much it is, the **`mb db sizes`** command can be helpful. Note the size of the `hash` table.

If space is of utter concern, generation of chunked hashes by can be switched off with `chunked_hashes = 0` in `/etc/mirrorbrain.conf`. This effectively disables generation of torrents. (Metalinks will still be generated, they'll just not contain piece-wise hashes.)

Parameters need to go into the mb instance section, not into the `[general]` section.

6.1.2 Configuration

There are the following Apache configuration directives to configure the torrents. Both go into a virtualhost context, and not inside a directory context.

- `MirrorBrainTorrentTrackerURL`

  Defines the URL a BitTorrent Tracker to be included in Torrents and in Magnet links. The directive can be repeated to specify multiple URLs. Here's an example:

  ```
  MirrorBrainTorrentTrackerURL "http://bt.mirrorbrain.org:8091/announce"
  MirrorBrainTorrentTrackerURL "udp://bt.mirrorbrain.org:8091/announce"
  ```

  The first URL is put into the `announce` key. All URLs will be listed in the `announce-list` key.

- `MirrorBrainDHTNode`

  Defines a DHT node to be included in Torrents links. The directive can be repeated to specify multiple nodes, and takes two arguments (hostname, port). Example:

  ```
  MirrorBrainDHTNode router.bittorrent.com 6881
  MirrorBrainDHTNode router.utorrent.com 6881
  ```

6.1.3 Registering with a tracker

Most trackers require registration of the torrents, or they refuse to deal with them. Unless you use a tracker that allows access to unregistered torrents, you will have to take care of the registration.

At the moment, this means downloading the torrents from MirrorBrain and doing this somehow. Or using an open tracker.

In the future, it is planned that MirrorBrain writes copies of the torrents to the file system, so they can easily be rsynced to somewhere else, where the registering process runs.

6.2 Configuring zsync support

The zsync distribution method resembles rsync over HTTP, so to speak, and is a very bandwidth-efficient way to synchronize changed individual files; at the same time, it is very scalable, because the main work is not done at the server (as with rsync), but distributed to the clients.

Normally, zsync metadata needs to be generated manually and saved in form of .zsync files next to the real files.

MirrorBrain however has support for generating the required checksums, and can then serve the zsync files generated on-the-fly. The generation is triggered by appending `.zsync` to an URL.

The supported method reflects the "simpler" zsync variant, which doesn't look into compressed content. It is compatible to, and was tested with, the current zsync release (0.6.2). 0.6.1 worked as well.

If mirrors are available for a file, MirrorBrain adds them into the zsync as URLs where missing data can be downloaded. zsync (0.6.1) requires real mirrors as URLs, not one URL that redirects to mirrors. It is noteworthy in this context that zsync client (as of 0.6.1) tries the provided URLs in random order. Thus, the sent URLs are restricted to the ones that are closest. Thereby, the zsync client will use more nearby mirrors to download data from.

If no mirrors are available, a valid zsync is still generated. The content will then be delivered directly by MirrorBrain.

Note: This feature is off by default, because Apache can allocate large amounts of memory when retrieving very large rows from database (and keeps it). This may or may not affect you; and it may be worked around in the future. (The amount of memory that Apache allocates is about twice the size of the largest zsync, so in the end it depends on the file sizes.)

To activate zsync support, you need to switch on the generation of the special zsync checksums. That is done like shown below in the MirrorBrain instance section into `/etc/mirrorbrain.conf`:

```
[general]
# not here!

[your mb instance]
dbuser = ...
...
zsync_hashes = 1
```

This will cause **`mb makehashes`** generate the zsync checksums and store them into the database. See *Creating hashes* for more info on this tool. This tool needs to be run periodically, or after known changes in the file tree, to update the checksums.

Note: For the fastest possible checksumming, the algorithm is implemented in C (zsync's own "rsum" checksum) and integrated via a C Python extension.

The checksums occupy space in the database. To find out how much it is, the **`mb db sizes`** command can be helpful. Note the size of the `hash` table.

6.3 Magnet links

The Magnet URI scheme allows to reference a file for download via P2P networks. See Wikipedia and the project website.

Magnet links are automatically included in Metalinks (v3 Metalinks as well as IETF Metalinks). They also appear in the mirror list.

Magnet links can contain a BitTorrent tracker URL. MirrorBrain includes tracker URLs configured via the Apache `MirrorBrainTrackerURL` directive into magnet links. This means that multiple trackers can be listed. Configuring tracker URLs is explained in the *Generating Torrents* section.

A magnet link can be requested from MirrorBrain simply by appending `.magnet` or `?magnet` to an URL.

Implementation notes:

- Hashes are hex-encoded, because Base32 encoding would be awkward to add and there seems to be a transition to hex encoding.
- The `urn:sha1` scheme is currently also not supported, because it is required to be Base32-encoded. Base32 encoding could be added in the future, of course. Contributions welcome!

6.4 Serving Yum-style mirror lists

Yum has a way to request a plain-text list of mirrors from a server. This is useful because it can then has a number of mirrors available as fallbacks, or can test which one works fastest for it. Traditionally, Yum is configured with a URL containing passing a few query arguments to select the appropriate repository. For example, the communication with the server could look like this:

```
request:
  http://centos.mirrorbrain.org/?release=5.5&arch=x86_64&repo=os
reply from server:
  http://ftp.uni-bayreuth.de/linux/CentOS/5.5/os/x86_64/
  http://ftp.hosteurope.de/mirror/centos.org/5.5/os/x86_64/
  http://mirror.sov.uk.goscomb.net/centos/5.5/os/x86_64/
```

MirrorBrain has support to answer these queries in a meaningful way. The returned list of mirrors will be sorted by suitability for the client – with the full-blown selection algorithm being applied. 10 mirrors will be returned (that could be made configurable, if need be).

The Apache config directive `MirrorBrainYumDir` is used to map the various possible query arguments to actual directories. Below these directories, MirrorBrain checks if a certain file present. Only mirrors that list that marker file will turn up in the mirror list.

The syntax of the directive is:

```
MirrorBrainYumDir <arg>=<regexp> [<arg>=<regexp> ...] <basedir> <mandatory_file>
```

Here, the meaning of the arguments is:

`arg`
: One ore more keys that the client uses in the query. The order in these are given does not matter.

`regexp`
: This is a regular expression against the values are matched which Yum sends. You are free to use a simple string here, like `updates`, or a regular expression that catches several things.

  These regular expressions are forced to be anchored to start and end, for security reasons.

When you use strings like `10.0`, remember that the dot is a magic in regular expressions (matching any character), so you should escape it with a backslash (see examples below).

See the next section for a discussion of security implications.

`subdir`
This is the directory that corresponds to the specified query arguments. Its path is given relative to the top of the file tree. At the same time, this is the path on the mirror servers, relative to the base URL of their mirror.

Since there can be a need for many different, but similar mappings, there is a way to automatically insert values from the query into this path. Any `$1` to `$9` specified in the `subdir` string will be replaced with the respective value.

For instance, `$2` is replaced with the *second* `arg` value defined here. (*Not* with the second argument that yum passes in. That would make no sense – since the order in which it places the values is not specified.)

It is not necessary to put `()` braces around the regexps, to declare them as a "match group". *Every* regexp is a match group.

`mandatory_file`
This is a file that needs to exist in the specified subdirectory on the mirrors. Again, its path is given relative.

Here's an example with two similar mappings:

```
MirrorBrainYumDir release=5\.5 repo=os arch=i586 5.5/os/i386 repodata/repomd.xml
MirrorBrainYumDir release=4\.8 repo=os arch=i586 4.8/os/i386 repodata/repomd.xml
```

If the client requests `?release=4.8&arch=i586&repo=os`, the second rule will match. MirrorBrain will create a list of all mirrors that are known to have the file `4.8/os/i386/repodata/repomd.xml`, and send the 10 best of these mirrors to the client. The mirror URLs will have the `subdir` appended as appropriate.

Here is a more complex (but not contrived) example which demonstrates how a multitude of cases can be handled with a single rule:

```
MirrorBrainYumDir release=(4|4\.8|5|5\.5) repo=(os|updates|extra) arch=i586 $1/$2/i386␣
↪repodata/repomd.xml
```

If the client sends `?repo=extra&release=5`, the directory becomes `5/extra/i386`, and so on.

Things you should note:

- Symlinks are automatically handled. Remember that symlinks on mirrors are invisible when scanning is done only via HTTP. To avoid having multiple trees in your database, you should make sure that only "real" directories are scanned via `scan_top_include` in `/etc/mirrorbrain.conf`. MirrorBrain canonicalizes all paths in the local filesystem before doing a lookup in the database. Therefore, it doesn't matter if the query arguments are in fact resolving to a symlink to a file tree.
- If no mirror is found in the database, any mirror configured via `MirrorBrainFallback` is considered. If none of the latter is configured, MirrorBrain at least returns its own URL, which, after all, will give the client a working download, so it should be better than nothing.
- The client needs to specify all arguments defined in a MirrorBrainYumDir, and all must match.

Security considerations are discussed in the following section.

6.4.1 Security notes

- Because MirrorBrain uses strings that are passed in from clients, which could potentially be malicious, these are handled with care.

  The normal resource limits of request processing in Apache apply. There is no special sanitizing for '/../' elements in the path.

  Such arguments are accepted if the regular expression leaves room for that – for instance, if it contains wildcards as `.*`.

  In any case, the resulting path is canonicalized in the local file system. It is assumed that this cannot have bad effects. The most that an attacker can achieve is that a path is canonicalized to an existing file on the server. For Apache, this will mean that it (debug-) logs something like:

  ```
  Error canonicalizing filename '/srv/nullmirrors/centos/5/os/x86_64/../../../../../..
  ↪/etc//repodata/repomd.xml'
  ```

  Canonicalization still fails because the file checked is always the one specified by the admin. Even if the canonicalization would accidentally work, no information about the file, or even about the success or failure, would be returned to the client.

  Any error in canonicalization will stop processing of the request. Success (let's assume it is possible) will result in the path being used in a database SQL query, asking for mirrors that have the file, which is highly unlikely. The SQL argument is passed to the database via a prepared statement with bound parameters, so there is no chance for SQL injection attacks.

  (And all Apache logging undergoes escaping, excluding viewing logs etc. as attack vector.)

  Still, it seems prudent to recommend to downright avoid the issue by using more specific regular expressions that accept only what you want.

6.5 Styling the mirrorlist / details pages

MirrorBrain generates per-file pages with all known metadata and a list of mirrors. This is triggered by appending `.mirrorlist` or `?mirrorlist` to a request. The pages are delivered with character set UTF-8 in the Content-type header.

The content of these pages are wrapped into a XHTML/HTML `DIV` container with `id="mirrorbrain-details"`. This gives a means for styling in conjunction with a stylesheet linked in via the `MirrorBrainMirrorlistStyleSheet` Apache config directive. `MirrorBrainMirrorlistStyleSheet` goes into virtualhost context and takes an URL, which can be relative or absolute. Either of the following would work:

```
MirrorBrainMirrorlistStyleSheet "http://www.poeml.de/~poeml/mirrorbrain.css"
MirrorBrainMirrorlistStyleSheet "/mirrorbrain.css"
```

Further, arbitrary design can be achieved by specifying the XHTML/HTML header and footer which are placed around the page body instead of the built-in XHTML. This is configured with the following two Apache configuration directives, which go into virtualhost context:

```
# Absolute path to header to be included at the top
MirrorBrainMirrorlistHeader /srv/www/htdocs/mb-header.html

# Absolute path to footer to be appended
MirrorBrainMirrorlistFooter /srv/www/htdocs/mb-footer.html
```

6.6 Configuring URL signatures

Signing URLs can be a way to protect content from unauthorized access. How does this work?

The redirector can generate redirection URLs that contain a signature with temporary validity. The signature can be checked on the mirror servers to restrict access to authenticated clients. This means that authentication and authorization of clients can be validated centrally (by the redirector), and the content on all mirrors protected.

To configure URL signing, the server needs to be configured with a secret key:

```
MirrorBrainRedirectStampKey my_key
```

In this example, `my_key` is an arbitrary string known only to the server administrators (including the mirror servers).

The directive may be used server-wide, or applied to certain directories.

It makes sense to combine this with setting up authentication to restrict access. When clients access the server *and* are authorized to access the file, and the server decides to redirect the request, the result is as demonstrated in the following example:

```
 # curl -sI http://192.168.0.117/extended/3.2.0rc5/OOo_3.2.0rc5_20100203_LinuxIntel_
↪install_ar.tar.gz
HTTP/1.1 302 Found
[...]
http://ftp.udc.es/OpenOffice/extended/3.2.0rc5/OOo_3.2.0rc5_20100203_LinuxIntel_install_
↪ar.tar.gz?time=1288879347&stamp=e215bb55bbea2c133145330f9e061f5b
```

Note the two arguments that are appended to the URL in the Location header.

The mirrors which receives these requests need to check two things:

1) Check if the "ticket" is a valid one, by hashing the timestamp and the shared secret – here in shell code:

   ```
    # echo -n '1288879347 my_key' | md5sum
   e215bb55bbea2c133145330f9e061f5b  -
   ```

   That hash is the same as came with the URL, and thus the ticket is valid (or was valid in the past).

2) Check if the "ticket" is still fresh enough, simply by comparing to current time. Shell example:

   ```
    # echo $(( $(date +'%s') - 1288879347 ))
   380
   ```

   While I was typing, the ticked became 380 seconds old (because I'm slow :-), which makes it outdated, and the mirror would reject it, based on the policy that it must not be older than e.g. 30 seconds.

These two checks to be done on the mirrors can easily be destilled in a script consisting of only a few lines.

The fact that MD5 collisions can be found nowadays should be pretty irrelevant considering the short-lived-ness of the tickets. That's one reason why the mirrors should check for the age. The other reason is because it allows using a one-way, non-revertible hash, instead of a symmetric encryption which would be less trivial to implement (md5 is available anywhere, while symmetric ciphers are not).

6.7 Using mod_mirrorbrain without GeoIP

mod_mirrorbrain can be used without GeoIP. This could happen in (at least) two ways:

1) Let's assume that GeoIP *cannot* be used: this would be the case if the traffic to be redirected is in an intranet.

   Country information that mod_mirrorbrain uses to select mirrors can be faked with the standard Apache module mod_setenvif as in the following example:

   ```
   SetEnvIf Remote_Addr ^10\.10\.*      GEOIP_COUNTRY_CODE=us
   SetEnvIf Remote_Addr ^10\.10\.*      GEOIP_CONTINENT_CODE=na
   SetEnvIf Remote_Addr ^192\.168\.2\.* GEOIP_COUNTRY_CODE=de
   SetEnvIf Remote_Addr ^192\.168\.2\.* GEOIP_CONTINENT_CODE=eu
   ```

   The `SetEnvIf` directive sets two variables for each client per its network address. Thus, you can configure your mirrors in the database to reflect those countries. You could make up pseudo country codes, if needed.

   Based on this information, mod_mirrorbrain will chose an appropriate server.

2) Let's assume that a simple round-robin distribution of requests is sufficient. This would be the case if the clients are all located in the same network or country. There is nothing to configure for this.

   In such a scenario, mod_mirrorbrain will farm out the requests to all the available mirrors by random. It will still do this according to the preference of each mirror, and according to availability of the requested files on each mirror. Mirror selection criteria as the online status of each mirror will still apply.

   Thus, this solution is more powerful than simple DNS-based round robin, or random request distribution via mod_rewrite. (Of course, contrary to those other solutions, it requires tracking the mirrors' status and contents.)

CHAPTER

SEVEN

USE CASES

7.1 Generating Hashes

See *Creating hashes* for now

7.2 Torrent server with webseeds

See *Generating Torrents* for now

…

7.3 Giving mirrors a different weight - balancing the load

The main criterion for load balancing is of course the network proximity, and (failing that) the geographic location (country/region); but sometimes there are several matching mirrors that are in the same country or network. Then there are two additional mechanisms for choosing the best mirror:

1) influencing preference by an assigned "score" value

2) geographical distance

These two mechanisms work together. Let's look at them in detail now.

7.3.1 1) Score value

The `score` value that is assigned to each mirror is a unitless number where the absolute value doesn't matter, but the relative height in comparison to other mirrors makes the difference. The default value is always `100`. If mirrors have the same score, they are randomly used equally often. Thus, the load is distributed equally. If the values differ, the randomization becomes "weighted" towards the mirrors with higher scores.

If there is only one mirror in a country, the value doesn't matter, because country is the stronger criterion. When there are several mirrors per country, you'll often simply stick to the default value of 100. However, if there's a small mirror that can't take much traffic, you can assign it a e.g. 20 or 50, and it will get much less load. How much, depends on the number of number of other mirrors available in the same country. If, on the other hand, you have a particular powerful mirror, you could assign it a score of 200, for instance.

Here's an example with three mirrors in different combinations of score values:

```
score value              100    100    100
percentage of requests   33%    33%    33%

score value              100     50     50
percentage of requests   60%    20%    20%

score value              100    200     10
percentage of requests   25%    73%     2%
```

In real life, you might have more mirrors. The effect of score values can be approximated with the following formula:

```
                 s
P = ---------------------------- * 100
       s + (s2 + s3 + ... + sn)

where
P = percentage of requests to a mirror
s = score of mirror
s2...sn = scores of the other mirrors
```

Imagine that you have a mirror with `score=50`, and other mirrors in the same country with the following scores: `150, 100, 100, 100, 100, 50, 50, 30`. Thus,

50 / (50 + 150+100+100+100+100+50+50+30) * 100 = 6%

about 6% of requests will routed to that mirror.

(However, also remeber that mirrors might not always be complete mirrors, so if they don't have certain files, they are automatically left out from the equation. The calculation is always file-based, and thus never static.)

7.3.2 2) Geographical distance

Geographical distance is calculated by an approximation that is both lightweight and fast, but sufficiently accurate. Similar as the score value, it's not about absolute numbers in a certain unit like km. It's more about giving mirrors a relative weight according to the distance.

Each mirror is rated with this formula (C code from the Apache module:

```
new->dist = (int) ( sqrt( pow((lat - new->lat), 2) + pow((lng - new->lng), 2) ) * 1000 );
```

This approximation is simple enough but works around the (spherical) globe.

Mirrors are then ranked against each other according to the calculated `dist` values. Internally, this can all be done with simple&quick integer arithmetic.

Specifically, at this point we also give the **score** some influence:

```
d = mirror->dist + distprio / mirror->score;
```

where:

```
int distprio = DISTANCE_PRIO / arr->nelts;
```

where `arr->nelts` is the number of mirrors in that particular group in that the ranking is calculated. `DISTANCE_PRIO` is defined to a number which you could change at compile time, but where this default value gives about the results I wanted:

```
#define DISTANCE_PRIO 2000000
```

A comment in the code reads:

```
/* the smaller, the smaller the effect of a raised prio in comparison to distance */
/* 5000000 -> mirror in 200km distance is preferred already when it has prio 100
 * 1000000 -> mirror in 200km distance is preferred not before it has prio 300
 * (compared to 100 as normal priority for other mirrors, and tested in
 * Germany, which is a small country with many mirrors) */
```

You see, both the geographical distance and the score value work together and both has some influence. This prevents the choice of a mirror that's either far away or has a low score value, or one of them – and vice versa.

CHAPTER

EIGHT

TUNING GUIDE

The following sections describe how to tune PostgreSQL and Apache for good performance.

Depending on the size of your install, this can be mandatory.

8.1 Tuning Apache

8.1.1 MPM

In general, it makes sense to use a threaded MPM for Apache. Prefork is less suitable, because the database connection pool would not be shared across the worker children. With prefork, each process would open its own connection to the database. For small installations, this might not matter. However, for a busy download server, the threaded event MPM or worker MPM are better choices.

The following would be a configuration for the event MPM which serves up to 960 connections in parallel, using 64 threads per process. (Values for threads per process that scale best on Linux are 32 or 64.):

```
# event MPM
<IfModule event.c>
    ServerLimit          15
    MaxClients          960

    StartServers          2

    ThreadsPerChild      64
    ThreadLimit          64

    MinSpareThreads      32
    # must be >= (MinSpareThreads + ThreadsPerChild)
    MaxSpareThreads     112

    # at 200 r/s, 20000 r results in a process lifetime of 2 minutes
    MaxRequestsPerChild 20000
</IfModule>
```

Refer to the Apache MPM documentation for details.

8.1.2 DBD connection pool

With threaded MPMs, the database connection pool is shared among all threads of an Apache child. Thus, it makes sense to tune the size of the pool to the number of processes and threads. The following should fit the above Apache dimensions quite well:

```
DBDMin  0
DBDMax  12
DBDKeep 3
DBDExptime 10
```

The total number of connections that Apache might open needs to be reflected in the PostgreSQL setup accordingly. The above example would be served safely with a database with a `max_connections = 500` setting. (This setting may seem far too high, but it keeps even enough headroom to start separate Apache servers for testing purposes, or for upgrade purposes (killing Apache with SIGWINCH for graceful stop, and starting a new one while the old one still continues to serve requests to their end).

Having said that, if you find that you need a hundred connections or more in an everyday situation, there is something wrong – then you need to check if you chose the Apache threading model, check for the size of the database connection pool, and verify that there are no big bottlenecks in the database (which causes Apache to stall and stack up working threads and connections).

8.1.3 Thread stack size

When using lots of threads, their might be funny effects on some platforms. The default stack size allocated by the operating system for threads might be quite large, e.g. 8 MB on Linux. If this leads to problems, you could considerably decrease the stack size as such:

```
# If this isn't set, the OS' default will be used (8 MB)
# which is way more than necessary
ThreadStackSize 1048576
```

But normally the default settings should just work, I guess.

8.1.4 HTTP/1.1 KeepAlive

KeepAlive, a HTTP/1.1 feature, saves overhead by reusing already existing TCP connections to process further HTTP requests. If no additional request arrives after n seconds, the server closes the connection.

This is a good thing, but it can also become a problem when too many threads/processes linger around waiting for the next request in the connection. Each such thread would occupy a slot that could otherwise be used to handle other requests. In addition, even the number of unused ephemeral ports could become scarce under extreme conditions. The configured default of a KeepAlive time of 15 seconds in most Apache installs is far more than necessary. A good value is 2 seconds, which keeps the good side of KeepAlive, but avoids the drawbacks to the extent that they don't tend to be a problem.

Hence, good settings are:

```
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 2
```

Note: When it is getting *really* serious, like when you are slashdotted, don't hesitate to simply switch KeepAlive off. You will see a drastic improvement, and probably save you. (If you did your homework and your website is lean and fast otherwise ;-) If it is fat and bloated, there is not much to do.)

8.2 Tuning PostgreSQL

To tune PostgreSQL for good performance, you should tweak some or all of the parameters below in `postgresql.conf`.

This config file often lives in the same directory as the PostgreSQL database, which would be `/var/lib/pgsql/data` on an openSUSE system – or it could be in `/etc`, as in `/etc/postgresql/8.3/main` on Debian Lenny.

8.2.1 Memory sizing

With a small database, using only a few megabytes, there will not be much need for tuning. With larger databases, that go into the hundreds of megabytes, tuning becomes important.

Note: Make sure to reserve enough memory for the database, especially if it will be large. As a rough first estimate, it is usually sufficient and optimal if the reserved RAM is about the same as the database size on disk.

Allocating memory to the database is done in the following way. PostgreSQL largely relies on buffer caching done by the OS. Thus, the first measure in "reserving" memory is to *not* run too much other stuff on the machine, which would compete for memory, or (in other words) having enough memory. In general, PostgreSQL's performance reaches its maximum when the whole dataset, including indexes, fits in to the amount of RAM available for caching. (That statement is true if the whole dataset is actually used – if only parts are used, top performance will be reached already with less memory. MirrorBrain tends to use the whole dataset, at least during mirror scanning.)

There is a special command **`mb db sizes`** that helps you to assess the size of your databases. See *Database size info with mb db size*. (Just note that changes may not be immediately reflected in the numbers, because the statistics are updated periodically by PostgreSQL.)

`shared_buffers`

This parameter should usually be set to about 10-25% of the available RAM. Maximum value may be limited by the SHMMAX tunable of the OS.

(Of course, if your database is only 10 MB in size, there is no benefit in increasing this value that far. It obviously depends on the database size.)

Mirror scanning can incur heavy write activity, if there is a lot of fluctuation in the file tree, and when done in a massive parallel way. Scanning performance can benefit from higher values (25-50%) here. For read performance, (as affecting Apache and mod_mirrorbrain) higher values are not needed.

`effective_cache_size`

This is the effective amount of caching between the actual PostgreSQL buffers, and the OS buffers.

This does not create RAM allocations nor does it change how PostgreSQL uses RAM – it just gives PostgreSQL an assumption about the availability of memory to the OS cache. This influences decisions in the query planner, regarding usage of indexes.

In principle, this value could be set to the sum of `cached + free` in the **`free -m`** output. However, this value needs to be divided by the number of processes using this memory simultaneously. To estimate the latter, you could use **`top`** to see how many PostgreSQL processes are busy at the same time.

Anyway, it is better to set this parameter too low rather than too high, because that could result in too many index scans.

Other memory parameters that you might want to increase are:

- `maintenance_work_mem` (generously)
- `work_mem` (a bit)

8.2.2 Connection setup

`listen_addresses`
: You'll need to change the parameter listen_addresses if you

a) run the web server on a different host than the database server, or if you

b) want to use the **`mb`** admin tool from a different host than the the database host.

The default is localhost only. Add '*' or comma-separated addresses.

`max_connections`
: The default of 100 should fit many cases. Apache's re-usal of connections is so efficient (and MirrorBrain quickly done with answering queries) that a handful connections is enough. However, if you use Apache's prefork MPM, every child will use a connection. Thus, if you allow to have 200 Apache processes running you will need to adjust max_connections accordingly. With a threaded Apache, the connection pool is shared, so no problem. This is further discussed above, in the notes regarding Apache tuning.

8.2.3 Transaction log

The transaction log (called Write-Ahead-Log or WAL) is a central thing in PostgreSQL, and the configuration of its handling important.

`synchronous_commit`
: In any case, you should disable the synchronous commit mode (synchronous_commit = off). The only case where you don't want that is if you have other databases than MirrorBrain, which require a higher level of data integrity than MirrorBrain does.

`wal_buffers`
: The default (64kB) may be increased to e.g. 256kB.

`checkpoint_segments`
: For big databases (hundreds of MB in size), increase this from 3 to 32.

`checkpoint_timeout`
: Increase to 15min.

To log a checkpoint whenever one occurs, set `log_checkpoints = on` and `checkpoint_warning to 1h`.

8.2.4 Deadlocks

`deadlock_timeout`
As described below, set this parameter to 30s.

Concurrent write access by different processes to the same rows causes a queue-up of those write-requests. A row can be written only by one process at a time. If a process waits too long, it gives up after a while. Its lock times out, so to speak, which is called a deadlock in this context. It's not a real deadlock in the common sense, it's just giving up after a while.

Read activity (as done by Apache + mod_mirrorbrain, serving users) is not affected by write activity locks. Write activity is mainly caused by mirror scanning. Scanning then again is often done in parallel, to save time, so it is typical to have to wait for locks (when two scanners happen to want to write to similar regions in the database).

The default time waiting for a lock is 1s in PostgreSQL, which is often too short for MirrorBrain. That could be too long for other applications in fact, but for a mirror scanner it doesn't matter if it has to wait many seconds now and then. In fact, it is best to increase the lock waiting time to something like 30 seconds. The deadlocks don't occur frequently when scanning, but when they occur, you don't want a scanner to give up on that part and have some missing files on the mirror later.

Such deadlocks are more likely to occur when scanning a new mirror, which means that every database row has to be touched (for each file found on the mirror). Even more likely (actually, unavoidable) are they when you fill your database for the first time, after installing, and the rows are created at the first place. In that case, you will see deadlocks occur frequently. The best advice is to ignore them and simply scan once again, after the first run has completed.

Later scans mainly see what they know already, so there is no reason to write to the database, which means that deadlocks don't occur.

8.2.5 Enhancing logging

Logging can be enhanced with some details that might be relevant or helpful to tuning for MirrorBrain:

`log_line_prefix`
To get more detailed log lines, set it to:

```
'%m [%p:%l] %u@%d '
```

Note the trailing space!

`log_lock_waits`
To log lock waits >= deadlock_timeout, set to `on`.

`log_min_duration_statement`
You could log all long-running queries by configuring this to e.g. `2000` (value is in milliseconds).

8.2.6 Preventing kernel "write floods"

The kernel might decide to write a whole bunch of changes to disk in one step. This might block other operations for several seconds. That's deadly for performance of a web server.

This may affect you if you run a huge MirrorBrain database (like, say, 1 GB in size) in conjunction with heavy write activity (scanning many mirrors).

You should first make sure that you have followed all the tuning advice given above.

Now, if

1) that is all fine and

2) you are sure that your database server principally has enough memory and
3) you see webserver hangs occuring intermittently and
4) there is no other memory-intensive task to be done by the database server

then you can tune your kernel to handle write in a (for us) more efficient way. (The kernel finds it more efficient, by default, to wait a while until there is a lot to write – for us it is more efficient to write out data more frequently, in smaller portions.)

The two kernel tunables that let us achieve the desired behaviour are:

```
vm.overcommit_memory = 2
vm.dirty_background_ratio = 0
```

Put them into `/etc/sysctl.conf` to make it a permanent change.

You'll find additional details in the official PostgreSQL documentation: http://www.postgresql.org/docs/9.3/static/kernel-resources.html#LINUX-MEMORY-OVERCOMMIT (the rest of the page is also readworthy).

CHAPTER

NINE

UPGRADING

9.1 Refreshing package sign keys

The key that is used to sign packages may expire from time to time, and needs to be renewed. This is done by the build maintainers, and you may need update your installation to know about the refreshed key.

If you see the following (on Debian or Ubuntu), this affects you:

```
lenny:~# apt-get update
[...]
Reading package lists... Done
W: GPG error: http://download.opensuse.org  Release: The following signatures were␣
↪invalid: KEYEXPIRED 1270154736
W: You may want to run apt-get update to correct these problems
lenny:~#
```

After running the following command, all should work fine again:

```
lenny:~# apt-key adv --keyserver hkp://wwwkeys.de.pgp.net --recv-keys BD6D129A
```

If it does not help, it is probably best to contact the build maintainers or the mirrorbrain mailing list.

9.2 Upgrading PostgreSQL

9.2.1 General notes

When upgrading PostgreSQL, it is important to look at the version number difference. If the third digit changes, no special procedure is needed (except when the release notes explicitely hint about it).

When the first or second digit change, then a dump-and-reload cycle is usually needed.

For instance, when upgrading from 8.3.5 to 8.3.7 nothing needs to be done. When upgrading from 8.3.7 to 8.4, you need to dump and reload.

You might want to follow the instructions that your vendor provides. If your vendor doesn't provide an upgrade procedure, be warned that the database needs to be dumped before upgrading PostgreSQL.

See `pg_dumpall(1)` for how to dump and reload the complete database.

Warning: If you use mod_asn, there is one more caveat. If your vendor's upgrade procedure automatically saves the previous PostgreSQL binaries in case they are needed later, the procedure might not take into account that the `ip4r.so` shared object might need to be saved as well. Hence, you might be unable to start the old binaries, when the ip4r shared object has been upgraded already.

Hence, it is recommended that you do a complete dump of the databases before upgrading, and load that after upgrading.

Note: When upgrading to 8.4, `ident sameuser` is no longer a valid value in `pg_hba.conf`. Replace it with `ident`.

9.2.2 Offline upgrade

The following console transcripts should give an idea about upgrading an PostgreSQL installation. It was done on an openSUSE system, but a similar procedure should work on other platforms.

The upgrade in this example is done while the database is taken offline, i.e. you need to plan for a downtime of the server. The cron daemon is stopped so that there are no attempted writes to the database. **`pg_dumpall`** is used to save the complete database to a file:

```
root@doozer ~ # rccron stop
Shutting down CRON daemon                                          done
root@doozer ~ # su - postgres
postgres@doozer:~> pg_dumpall > SAVE
postgres@doozer:~> exit
root@doozer ~ # rcpostgresql stop
Shutting down PostgreSQL server stopped                            done
```

At this point, you would upgrade the PostgreSQL software.

Next, the `data` directory is moved away, a new one created:

```
root@doozer ~ # old /var/lib/pgsql/data
moving /var/lib/pgsql/data to /var/lib/pgsql/data-20090728
root@doozer ~ # rcpostgresql start
Initializing the PostgreSQL database at location /var/lib/pgsql/data  done
Starting PostgreSQL                                                done
root@doozer ~ #
```

Now, the authentication setup and the configuration need to be migrated from the old install to the new one:

```
root@doozer ~ # su - postgres
postgres@doozer:~> cp data/pg_hba.conf data/pg_hba.conf.orig
postgres@doozer:~> cp data/postgresql.conf data/postgresql.conf.orig
postgres@doozer:~> vi -d data-20090728/pg_hba.conf data/pg_hba.conf
postgres@doozer:~> vi -d data-20090728/postgresql.conf data/postgresql.conf
```

Then, load the dump into the new database:

```
postgres@doozer:~> psql template1 -f SAVE
[...]
```

Finally, restart PostgreSQL, Apache and cron:

```
root@doozer ~ # rcpostgresql restart
Shutting down PostgreSQL server stopped                             done
Starting PostgreSQL                                                 done
root@doozer ~ # rcapache2 reload
Reload httpd2 (graceful restart)                                    done
root@doozer ~ # rccron start
Starting CRON daemon                                                done
```

9.2.3 Online upgrade

Using a second PostgreSQL daemon, started temporarily, an online upgrade can be performed as follows.

First, create space for the temporary database:

```
root@mirrordb ~ # mkdir /space/pgsql-tmp
root@mirrordb ~ # chown postgres:postgres /space/pgsql-tmp
```

Create the new (temporary) database:

```
root@mirrordb ~ # su - postgres
postgres@mirrordb:~> initdb /space/pgsql-tmp/data
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale en_US.UTF-8.
The default database encoding has accordingly been set to UTF8.
The default text search configuration will be set to "english".

creating directory /space/pgsql-tmp/data ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers/max_fsm_pages ... 32MB/204800
creating configuration files ... ok
creating template1 database in /space/pgsql-tmp/data/base/1 ... ok
initializing pg_authid ... ok
initializing dependencies ... ok
creating system views ... ok
loading system objects' descriptions ... ok
creating conversions ... ok
creating dictionaries ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the -A option the
next time you run initdb.

Success. You can now start the database server using:
```

(continues on next page)

(continued from previous page)

```
    postgres -D /space/pgsql-tmp/data
or
    pg_ctl -D /space/pgsql-tmp/data -l logfile start

postgres@mirrordb:~>
```

Copy the configuration and the authentification setup to the temporary database:

```
postgres@mirrordb:~> cp /space/pgsql/data/postgresql.conf /space/pgsql-tmp/data/
postgres@mirrordb:~> cp /space/pgsql/data/pg_hba.conf /space/pgsql-tmp/data/
```

Note: The second database server will need RAM — maybe it will be necessary to adjust the `shared_buffers` setting in `postgresql.conf` for both daemons, so they don't try allocate more memory than physically available.

Next, change the port in the temporary `postgresql.conf` from 5432 to 5433 and start the second PostgreSQL server:

```
postgres@mirrordb:~> vi /space/pgsql-tmp/data/postgresql.conf
postgres@mirrordb:~> postgres -D /space/pgsql-tmp/data
```

Note: This assumes that Apache is configured to use a TCP connection to access the database server, not a UNIX domain socket.

Load the dumped data (not forgetting to use the differing port number):

```
postgres@doozer:~> psql -p 5433 template1 -f SAVE
[...]
```

Now the Apache server, and possibly other services (`/etc/mirrorbrain.conf`) need to be changed to the temporary port, and (gracefully) restarted.

Note: Verify that everything works as expected with the temporary database. If it does, stop the primary PostgreSQL server (and verify again that everything still works).

From here on, the next steps are probably obvious. You would proceed as described in the previous section. After upgrading the PostgreSQL install, loading the data, copying/merging `postgresql.conf` and `pg_hba.conf`, you would revert the Apache configuration to use port 5432 and reload it.

If everything works, you can stop and remove the temporary database installation.

9.3 Version-specific upgrade notes

9.3.1 To 2.18.0:

Due to the modernized HTML of the .mirrorlist page, please check your CSS styling for them (if you have some).

9.3.2 To 2.14.0:

To take advantage of mirror selection by geographical distance (as additional criterion to country, network prefix etc.), the free GeoLite City GeoIP database needs to be used. If you used the simpler database so far, you need to switch to the city edition which contains the required data. The following steps are necessary:

1) Use the provided **`geoip-lite-update`** tool to download it (and keep it updated regularly via cron).
2) Edit your mod_geoip configuration and change `GeoIP.dat` to `GeoLiteCity.dat.updated`
3) Run **`mb update -A --all-mirrors`** to update the mirrors' GeoIP data (coordinates, country and region), and (if **`mod_asn`** is used) autonomous system and prefix.

MirrorBrain will continue to run fine without the extra data. Thus, it is not obligatory to switch to the larger GeoIP database.

9.3.3 From 2.13.x to 2.13.3:

If you actively use Torrents, it may make sense to recreate the hashes. It is not strictly necessary, because the only hash that has been changed is the BitTorrent info hash (`btih`) that is cached in the database. That hash is used only if requested by `<URL>.btih` and it is shown on the details page, but it is not used in actual Torrent generation. Thus it is unlikely to matter. Anyhow, it could cause confusion.

You can use **`mb makehashes`** with the `--force` option once to recreate the hashes.

9.3.4 From 2.12.x to 2.13.0:

- If you created hashes in the past, please edit your `/etc/crontab` and replace the calls to the former tool `metalink-hasher update` with a call to `mb makehashes`. Example:

```
# old tool:  metalink-hasher update -t /srv/metalink-hashes/srv/ooo /srv/ooo
# new tool:  mb makehashes -t /srv/metalink-hashes/srv/ooo /srv/ooo
```

- If you used the the **`metalink-hasher`** with the `-b` option in the past, check the usage examples that come with **`mb help makehashes`**. The option has been rewritten to be easier to use, and it should now be easier to get it to do what you want.

9.3.5 From 2.11.1 to 2.11.2:

The **`mb vacuum`** command has been renamed to **`mb db vacuum`**. The old command will continue to work for now - but existing cron jobs should be updated; the old command might be depracated later.

Users that happen to use the **`mirrorprobe`** with the default timeout of 60 seconds should now run it with `-t 60`, because the default has been lowered to 20 seconds with release.

9.3.6 From 2.10.3 to 2.11.0:

The `MirrorBrainHandleDirectoryIndexLocally` directive has been removed. A warning is issued where it is still found in the config. It didn't really have a function.

CHAPTER

TEN

DOCUMENTATION FOR DEVELOPERS

If you are contributing to the MirrorBrain project, please read this first.

10.1 Participating in the community

The community exists mainly through mailing lists and a Subversion repository. To participate:

Go to http://mirrorbrain.org/communication/ and join the "mirrorbrain", "mirrorbrain-commits", and "mirrorbrain-announce" mailing lists. The dev list, "mirrorbrain", is where almost all discussion takes place. All development questions should go there, though you might want to check the list archives first. The "mirrorbrain-commits" list receives automated commit emails.

Get a copy of the latest development sources from http://svn.mirrorbrain.org/svn/mirrorbrain/trunk/. New development always takes place on trunk. Bugfixes, enhancements, and new features might be backported from there to the various release branches.

How to work on the documentation is described in *How to improve this documentation*.

CHAPTER

ELEVEN

KNOWN PROBLEMS

11.1 Database reconnections issues

If the database goes away (or is restarted), it is not clear at the moment if the pgsql database adapter used by Apache's DBD library reconnects cleanly. Sometimes it seems that it doesn't reconnect, or that at least some of the Apache children don't do it, and a graceful restart of Apache is needed. This needs further inspection.

11.2 Sporadic corruption of ASN and PFX variables in the subprocess environment

mod_asn lookup data is sometimes garbled (subprocess env table):

```
87.79.141.235 - - [16/Feb/2009:14:41:38 +0100] "GET /factory/repo/oss/suse/setup/descr/
↪packages.gz HTTP/1.1" 200 2416300 "-" "ZYpp 5.25.0 (curl 7.19.0)" - r:- 145 2416594 -:-
↪ ASN:8422 P:87.78.0.0/15 size:- - - "-"
87.79.141.235 - - [16/Feb/2009:14:41:38 +0100] "GET /factory/repo/oss/suse/setup/descr/
↪patterns HTTP/1.1" 200 164 "-" "ZYpp 5.25.0 (curl 7.19.0)" - r:- 142 431 -:- ASN:{&\
↪x80\x02 P: size:- - - "-"
87.79.141.235 - - [16/Feb/2009:14:41:39 +0100] "GET /factory/repo/oss/license.tar.gz␣
↪HTTP/1.1" 200 24492 "-" "ZYpp 5.25.0 (curl 7.19.0)" - r:- 131 24782 -:- ASN:8422 P:87.
↪78.0.0/15 size:- - - "-"
```

As a further data point, the following has been seen:

```
90.182.x.x - - [20/Jul/2009:12:16:19 +0200] "GET /distribution/10.3/repo/oss/content␣
↪HTTP/1.1" 302 348 "-" "Novell ZYPP Installer" ftp.linux.cz r:country 170 901 EU:CZ␣
↪ASN:z,ne,ng,re,rw,sc,sd,sh,sl,sn,so,st,td,tf,tg,tn,tz,ug,yt,za,zm,zw a2 ge,kz,ru P:90.
↪180.0.0/14 size:44325 - - "-"
```

This further points to a memory corruption, happening while or after mod_mirrorbrain is running. It seems very hard to trigger though, I'd estimate 20 requests out of 10.000.000.

Grepping logs for `xaa` yields some places for examination:

```
zcat /var/log/apache2/download.opensuse.org/2009/07/download.opensuse.org-20090720-
↪access_log.gz| grep 'xaa'
```

Something might be corrupting data in the env table. It doesn't seem to affect other env variables though, only the two that are set by mod_asn.

All places where the subprocess env table is manipulated in mod_asn and mod_mirrorbrain seem safe.

A "soft" way of debugging would be to add a detailed debugging logging coupled to a trigger which pulls when non-numeric data is seen in `ASN` or `PFX`. When actually adding several such triggers at different stages of request processing, it should be possible to pinpoint the corruption. In addition, a core dump of the process could be saved to disk while running.

The bug doesn't seem to have noticeable negative consequences except the messed up logging of the two values.

11.3 Inaccurate logging of ASN lookup data for 404s

It was noticed that for 404s the lookup is correctly done (can be seen in the response headers), but `ASN:- P:-` is being logged.

Empty logging also happens for 200s, when e.g. requesting something from `/icons`, so that fits the configuration:

```
87.79.141.235 - - [16/Feb/2009:15:55:43 +0100] "GET /icons/torrent.png HTTP/1.1" 200␣
↪3445 "http://download.opensuse.org/distribution/11.1/iso/" "Mozilla/5.0 (Macintosh; U;␣
↪Intel Mac OS X 10_5_6; en-us) AppleWebKit/525.27.1 (KHTML, like Gecko) Shiira Safari/
↪125" - r:- 405 3744 -:- ASN:- P:- size:- - - "-"
```

But what about redirections exceptions, like size? Example:

```
78.34.103.82 - - [16/Feb/2009:16:05:54 +0100] "GET /distribution/11.1/repo/oss/suse/
↪setup/descr/patterns HTTP/1.1" 200 170 "-" "ZYpp 5.24.5 (curl 7.19.0)" - r:- 152 448 -
↪:- ASN:8422 P:78.34.0.0/15 size:- - - "-"
```

11.4 Inaccurate logging of ASN lookup data for overridden AS

When the AS is overridden with a query parameter, the one being logged/put into the headers is not the overriding one.

11.5 Wrong reporting by the scanner about symlinks

When scanning via rsync, the scanner reports symlinks (which are mode 0777) as world-writable directories:

```
lrwxrwxrwx 1 root root 13 2009-03-01 05:29 /mounts/dist/unpacked/head-i586/usr/bin/
↪pnmnoraw -> pnmtoplainpnm*

rsync dir: 777         13 Sun Mar  1 05:29:31 2009  unpacked/head-i586/usr/bin/pnmnoraw
```

That doesn't harm, because the symlinks are (intentionally) ignored anyway, but it clutters the output and of course is confusing.

11.6 Pondering a hard dependency on mod_geoip

One could argue that mod_geoip should be a hard requirement, as mod_form is - and Apache should check at startup. On the other hand, it could be optional, because mod_mirrorbrain also works without mod_geoip (distributing requests world-wide, according to mirror priorities). So it could be a valid use case to run without mod_geoip. On the other hand, at least a warning should be issued, so admins get a hint when mod_geoip was simply forgotten. On the other hand, we could have a check that prevents starting, unless geoip usage is explicitely disabled (e.g. MirrorBrainRequireGeoIP off)

Update: explicitely disabling GeoIP would somehow conflict with the "no GeoIP" use case.

11.7 Mirror list shows wrong region when overridden with query parameter

Since "override countries" (overridden via query paramter) are not resolved into region, a wrong region is given in generated mirror lists:

```
http://download.opensuse.org/distribution/11.1/repo/oss/suse/noarch/rubygem-rails-2.1.1-
↪1.14.noarch.rpm?mirrorlist&country=ZA
Found 2 mirrors which handle this country (ZA): <- ok
Found 61 mirrors in other countries, but same continent (EU): <- wrong
```

11.8 Mirror list gives inaccurate "number of mirrors", if mirrors were excluded

The mirrorlist gives inaccurate readings for "number of mirrors", if some mirrors where not considered, because they are configured `country-only` or `region-only` (`same_region=1` or `same_country=1`)

As further effect of this bug, it was noticed that a mirror is missing from the ?mirrorlist mirror lists if it is configured as fallback mirror for a country:

```
http://download.opensuse.org/repositories/KDE:/KDE4:/STABLE:/Desktop/openSUSE_11.1/KDE4-
↪DEVEL.ymp?mirrorlist&country=tw
```

ftp5 disappears from the list, when configured as fallback for Taiwan. It is correctly used though and appears on the list *when* actually used as fallback.

11.9 `mb file ls` crashes if probing for files that don't exist in the database

If globbing in the database for a file that doesn't exist, with the `--probe` option, probing shouldn't actually be attempted. The tool tries nevertheless and crashes:

```
 % mb file ls '*libqt4-debuginfo-4.5.2-51.1.x86_64.rpm' -u --md5
Traceback (most recent call last):
  File "/suse/poeml/bin/mb", line 1123, in <module>
    sys.exit( mirrordoctor.main() )
```

(continues on next page)

(continued from previous page)

```
  File "/usr/lib64/python2.5/site-packages/cmdln.py", line 257, in main
    return self.cmd(args)
  File "/usr/lib64/python2.5/site-packages/cmdln.py", line 280, in cmd
    retval = self.onecmd(argv)
  File "/usr/lib64/python2.5/site-packages/cmdln.py", line 412, in onecmd
    return self._dispatch_cmd(handler, argv)
  File "/usr/lib64/python2.5/site-packages/cmdln.py", line 1100, in _dispatch_cmd
    return handler(argv[0], opts, *args)
  File "/suse/poeml/bin/mb", line 854, in do_file
    samples = mb.testmirror.lookups_probe(rows, get_digest=opts.md5, get_content=False)
  File "/suse/poeml/mirrorbrain/mirrordoctor/mb/testmirror.py", line 201, in lookups_
↪probe
    return probes_run(probelist)
  File "/suse/poeml/mirrorbrain/mirrordoctor/mb/testmirror.py", line 228, in probes_run
    result = p.map_async(probe_report, probelist)
  File "/usr/lib64/python2.5/site-packages/processing/pool.py", line 186, in mapAsync
    chunksize, extra = divmod(len(iterable), len(self._pool) * 4)
ZeroDivisionError: integer division or modulo by zero
```

CHAPTER

TWELVE

RELEASE NOTES/CHANGE HISTORY

12.1 Release 2.18.1 (r8379, Feb 3, 2014)

Bug fixes:

- **geoip-lite-update**: This tool reloads Apache, but the `systemctl` call added in 2.18.0 for operating systems with `systemd` was incorrect. This has been fixed and was the reason for this release.
- **mod_mirrorbrain**: Some obsolete configuration directives have been removed (`MirrorBrainGeoIPFile`, `MirrorBrainHandleDirectoryIndexLocally`, `MirrorBrainMetalinkHashesPathPrefix`). `MirrorBrainGeoIPFile` was also still present in the example configuration file. (issue 146)
- mb scan: an obsolete reference to a -f switch was removed that exists only in the behind-the-scene scanner perl script, but not in mb scan. Thanks, Bart!

In the documentation about system tuning, a reference to the PostgreSQL docs has been added.

12.2 Release 2.18.0 (r8365, Feb 2, 2014)

(Pleae note that a new version of **mod_asn** was also issued recently.)

New features:

- **mod_mirrorbrain**: Nick Schermer from Xfce contributed a wonderful patch to improve the HTML output for the details pages that MirrorBrain generates (see issue 123). As a consequence, in some installations the web design needs to be adjusted, but hopefully people will value the better possibilities. The full list of changes can be viewed here.
- **mod_mirrorbrain**: If multiple instances of MirrorBrain run in Apache (or you have multiple vhosts using one MirrorBrain configuration), you would have multiple `DBDParams` statements which prepare SQL statements when Apache starts processes. Peculiarly, Apache doesn't allow the same connection string used more than once. (To make the connection strings unique, a possible workaround is to use differing `connect_timeout` values.) Anyhow, to help users running into this problem, MirrorBrain used to log a warning (added in 2009). This warning was removed when the DBD error handling was reworked in 2.16.1 (2012). It replaced with much more detailed error logging, but the helpful one-liner was missing hence. This release re-adds the helpful one-liner, and it'll also show the workaround at the same time. In addition, the documentation was enhanced. Thanks Stephan Jauernick.
- The installation documentation was updated in many places.
- **mb**: new command line option for configuration file path. Patch kindly provided by Gökdeniz Karadağ (see also issue 114)

- **`mb update`**: The **`geoiplookup`** and **`geoiplookup_city`** binaries are now also looked for in `/usr/share/mirrorbrain`. Helps to solve the packaging cleanup issue 110.
- **`geoip-lite-update`**: It is now possible to run this script without reloading Apache. On the other hand, it can now reload Apache on openSUSE, Ubuntu, Debian, Fedora, and via systemd. Thanks Andrea Veri for the report.
- The **`create_timestamp`** script no longer contains openSUSE specifics. Usage:

```
create_timestamp [username:groupname] timestampfile1 [timestampfile2...]
```

- **`geoiplookup_city`** and **`geoiplookup_continent`** tools: The path to the GeoIP database files is now configurable at compile time, so distribution builders could use preprocessor definitions to define them instead of patching the code. Patch by Dagobert Michelsen. Thanks! (issue 130)
- **`mirrorprobe`**: in the `--help` output, display the default value for the network timeout (20 seconds)

Bug fixes:

- There's a `version` table in the database since recently (created since 2.17.0), but it didn't contain an `id` column as primary key, so SQLObject couldn't work with it. So now we add an `id` column as primary key to the table (and an existing table from 2.17.0 is migrated by simply recreating it from scratch).
- The database SQL scheme for new installations has been updated to add the new column named `ipv6_only`. Thanks George Koutras, Raphael Hertzog and others for the report (and for their patience)! (issue 119)
- The SQL schema was updated to remove obsolete quotes around language names on function declarations: `'plpgsql'` -> `plpgsql`; `'SQL'` -> `SQL`. PostgreSQL 9.2 and newer no longer ignore these wrong quotes.
- **`mod_mirrorbrain`**: Compiler warnings about using `%d` for `size_t` were silenced, by now using `APR_SIZE_T_FMT` where appropriate. (issue 82)
- **`mb`** / `mirrorbrain.conf`: Trailing(!) spaces in passwords were taken literally so far, but were very hard to see and hard to debug. Now, trailing spaces are rightfully ignored (issue 112). Thanks to patch from Pat Riehecky!
- **`mb`** now gives sane error messages when a config statement is missing/misspelled in `/etc/mirrorbrain.conf`.
- **`mb`**: when mod_asn is not installed, an additional `ProgrammingError` exception from the `sqlobject.dberrors` can occur. This is now also caught. Thanks Gökdeniz very much for the patch!
- **`mb makehashes`**: "permission denied" errors are now handled gracefully, fixing issue 105. Thanks Tom Albers for report & patch!
- **`metalink-hasher.py`**: This very old (backward) compatibility wrapper has been removed from the tools directory, since it is long obsolete.
- **`mb iplookup`**: On the Solaris/OpenCSW platform, using `socket.getaddrinfo()` in Python for DNS lookups doesn't work with port `0`. Using `None` instead seems to be more correct and hopefully work on all platforms. Thanks Dagobert for the fix! (issue 135)
- **`mb edit`**: A mistyped dash in the commands help output was fixed. Thanks Dago! (issue 136)
- **`mb update`**: This command now handles errors that lead to `Null` as prefix or `Null` as AS number, so the command doesn't crash anymore under these conditions. (issue 137)
- **`mb dirs`**: When using the `-d` or `--missing` option, only enabled mirrors are shown now. Thanks Florian! (issue 116)
- **`tools/geoiplookup_*`**: They no longer segfault when opening a GeoIP database database fails. Patch courtesy of Dagobert Michelsen. (issue 138)
- **`mb scan`**:
  - When FTP URLs are not correct, and the directory is not found on an FTP server, the scanner bailed out. Fixed with patch from Dago. (issue 139)

  - when in verbose mode, don't wrongly log symlinks as directories (issue 141)
- The file `mb/countries.py` was never used; remove it so it doesn't confuse anyone. Thanks Gokdeniz for the hint.
- The build on openSUSE 13.1 was fixed.
- The build on RHEL6 was fixed, with a patch courtesy of jcpunk. (issue 125)

12.3 Release 2.17.0 (r8289, Apr 21, 2012)

New features:

- **`mod_mirrorbrain`**: IPv6 geolocation for IPv6 clients is now enabled. This requires **`GeoIP`** 1.4.8 and **`mod_geoip`** 1.2.7 or newer (which add experimental support for IPv6 resolution) (issue 106).
- **`mb update`**, **`mb iplookup`**: DNS resolution now works with IPv4 + IPv6.
- Support for Metalink/HTTP (**RFC 6249**) has been implemented (issue 15). This was long on my todo list! This makes MirrorBrain include in its server HTTP responses useful metadata like cryptohashes, mirror URLs and links to alternate representations. There's support for **RFC 5988** Web Linking, for **RFC 6249** Metalink/HTTP: Mirrors and Hashes, and for **RFC 3230** HTTP Instance Digests (including updates from **RFC 5843**). Here's an example (screenshot).
- **`mb edit`**: An editor set via the environmental variable `$VISUAL` is now used, if none is set in `$EDITOR`. This fixed issue 96.
- **`mb db vacuum`**: A new option `-q` allows to silence the commands output (issue 99).

New platforms:

- Ubuntu 11.10 packages are now built and tested.
- Debian 6.0 packages have been tested.

Bug fixes:

- **`mb makehashes`**: A problem was fixed with filenames containing characters that could be interpreted as magic characters in regular expressions (issue 94) Thanks, KDE sysadmins, for your help!
- **`mirrorprobe`**: Incomplete responses returned by mirrors (less bytes sent than announced) spawned an annoying error message. Now this error is just logged, as it should.
- **`mb scan`**: A typo has been fixed (patch by Oliver Beattie)

Internally, a way to migrate the database after updates has been implemented. A table named `version` keeps info about the state of the database. Thus, database schema upgrades can hopefully be done automatically when possible. This release adds a new flag to the database called `ipv6_only` to denote mirrors that are not reachable by IPv4. Once this new flag is used, it will allow to redirect clients to this type of mirrors. (And IPv6-only mirrors won't go away – rather the contrary :-)

12.4 Release 2.16.1 (r8261, Mar 25, 2012)

Bugs fixed:

- **`mb makehashes`**: It didn't work with PGP signature files that were not detached signatures. Non-detached (attached) signatures are now ignored because they could be very large (file size of the original file plus signature) (issue 102). Thanks to Tom Albers for his help here.
- **`mb makehashes`**: It no longer writes metalink data / cryptohashes into files. All hashes are stored in the database since 2.13.0. The obsolete storage in files had been kept only for backwards compatibility with 2.12 and earlier. (Which are outdated since ~18 months now. So let's avoid confusion.)
- **`mod_mirrorbrain`**: Reworked error handling regarding the acquisition of database connections, including more detailed logging of errors. This fixes a crash that affected only setups with `MirrorBrainFallback` configuration. The crash could occur when no database connection was available – because the logging code wrongly tried to log details about the (unavailable) connection.

New features:

- **`mb mirrorlist`**: Path names can now contain wildcards. (Edited via **`mb markers -e`**). Very nice improvement, thanks to idea and patch from Stephan Jauernick.
- There is a little new tool: **`tools/push2mirrors`**, an example script to run rsync processes in parallel to push content to mirrors.

12.5 Release 2.16.0 (r8251, Feb 21, 2012)

This release sums up small fixes that piled up slowly, over a good year.

URL signing is no longer regarded experimental. See *Configuring URL signatures* for more information on this interesting feature.

`mb dirs`: A new option was added to list all mirrors which don't have a specified directory: `mb dirs --missing DIR`

The following bugs were fixed:

- **`mod_mirrorbrain`**: The server could crash if there was configuration for fallback mirrors in place and acquisition of a database connection failed. This has been fixed (issue 84).
- **`mb scan`**:
  - If some directories on a mirror return 404, the scanner crashed. This is fixed with a patch kindly contributed by Thorsten Behrens.
  - As adjunct to r8180 (terse logging), logging messages in the large file check have been silenced (and their formatting improved, while at it)
- **`mb`**: A compatibility issue in Python has been fixed, by updating the way how exceptions are raised. Thanks to Christian Lohmaier for bringing this up.
- **`mb makehashes`**: Handling of non-availability of SHA256 cipher was fixed for old Python versions (issue 85).

In the documentation, the section about *Configuring URL signatures* has been added, and a general build problem has been documented: On some newer platforms, there is a need for linking the math library (as in `-lm`).

12.6 Release 2.15.0 (r8232, Nov 13, 2010)

This release comes with a new feature useful for RPM-based Linux distributions: generation of Yum mirror lists. Another new feature is that nginx directory indexes can be scanned. In addition, there are several bug fixes and improvements, and new documentation on tuning your database server for optimal performance.

Yum-style mirror list support is configured with a new Apache configuration directive which creates a mapping of Yum's query arguments to directories in the file tree. Please refer to the complete instructions in *Serving Yum-style mirror lists*.

To make this possible, the main handler function in `mod_mirrorbrain` is now run before all other configured handlers from other modules, not as the very last one. This means we can run before `mod_autoindex`, which would otherwise handle a request on a directory, despite the presence of query arguments requesting a yum mirror list. It also means that we run before `mod_php` (most modules' handlers run as middle hook and therefore not in strictly defined order).

A small bugfix is that, for generated torrent files, hashes from the database were retrieved twice from the database. This has been fixed.

The mirror scanner (`mb scan`) underwent the following small improvements, other than implementing support for **scanning Nginx directories**:

- When scanning only a subdirectory, the calculation of added/removed files was wrong. (It functinally did the right thing, but the logging was wrong.)
- The pre-scanning check for existance of a subdirectory is now skipped, when scanning only a single mirror.
- The messages logged when encoungering unparseable HTML index when scanning over HTTP have been improved.
- Logging messages about directories "not in top_include_list" have been silenced (but can be brought up again by increasing verbosity).
- Displaying of file sizes > 4GB when scanning over rsync has been fixed (a finding from issue 8).

In `mb makehashes`, issue 72 has been fixed: If specified path names contain duplicated slashes, these were introduced as wrong filenames into the database.

The documentation on *Tuning PostgreSQL* has been extensively reworked and gives a complete set of instructions now.

12.7 Release 2.14.0 (r8210, Nov 6, 2010)

This release brings a number of new features, and also some bug fixes.

- On the precondition that the "GeoLite City" GeoIP database is used, MirrorBrain now uses geographical distance as additional criterion in mirror selection. This is useful in

  1) large countries (like the US), and probably any countries with many mirrors
  2) countries without mirrors, where only a random mirror from the continent could be selected otherwise. (Defining fallback mirrors for such countries worked before, and still has precedence.)

  This implements issue 34. To take advantage of this feature, the free GeoLite City GeoIP database needs to be used. See the 2.14.0 upgrade notes for instructions.
- Per-file mirror lists have been improved by showing data in a better readable way, and by embedding a link to a Google map visualizing the 9 closest mirrors.
- When running behind a load balancer or other reverse proxy, prefix detection (for containment in network prefixes of mirrors) did not work because mod_mirrorbrain only saw the connecting IP address, and didn't look at an

address passed via HTTP headers from the proxy. This has been fixed. (AS, country and continent comparisons already did this.)

- Experimental support for restricted downloads has been implemented, by redirecting to temporary URLs whose validity can be verified by the mirrors. See http://www.mail-archive.com/mirrorbrain@mirrorbrain.org/msg00011.html This a prototype implementation that might still be changed, hence the new Apache config directive is called `MirrorBrainRedirectStampKey_EXPERIMENTAL` at the moment.
- MirrorBrain did not accept requests when access was restricted with authentication (e.g. Basic Authentication), due to a broken check which simply needed to be removed. (A bit of code inherited from mod_offload, and likely still dating back to old Apache 1.3 API.)
- MirrorBrain has been tested (successfully) against the latest **`zsync`** release (0.6.2) and the documentation updated.
- Minor optimizations and code cleanups have been done.

Please read the 2.14.0 upgrade notes before upgrading!

12.8 Release 2.13.4 (r8188, Oct 19, 2010)

This is a maintenance release with improvements in the mirror scan reporting, and small fixes and improved usability. In addition the documentation were enhanced and added to in some places.

Noteworthy are the added instructions on setting up automatic GeoIP database updates (see below).

- **`mb scan`**:
  - The output of the scanner has been improved, by introducing a `-q|--quiet` option. Used once, only a summary line per scanned mirror will be shown. Used twice, no output will be produced except errors.
  - When a scan via rsync ran into a timeout, the name of the affected mirror was not reported. The error message was only "rsync timeout", and while there normally were other messages giving a hint, output is now improved to include the mirror identifier.
  - When enabling a mirror after successful scanning, the scanner now makes sure that the mirror is not only marked "enabled" but also marked being "online". Mirrors are normally marked online by the mirrorprobe (which is typically run once per minute), but it is much more logical when a mirror is really directly available after scanning with `--enable`.
- **`mb scan`** and **`mirrorprobe`**:
  - There was a case of a quirky web server software that ignores requests without Accept header. The mirrorprobe and the scanner now send an Accept header with value '/', because sending this header in general should not harm.
- **`geoip-lite-update`**:
  - This script now works on Ubuntu. It no longer relies on a command named **`ftp`** being capable of doing HTTP downloads, and prefers **`curl`** or **`wget`** if available.
  - The script is quiet now, producing no output if no error is encountered.

Documentation improvements:

- The logging configuration example has been updated (See *Setting up logging*)
- The instructions to update the GeoIP databases on Ubuntu have been updated. (See *Installation on Debian/Ubuntu Linux*)
- Documentation (for all platforms) about setting up automatic updates of the GeoIP database was blatantly missing.

- A possibly disturbing '-' in front of cron examples has been removed, which work with Vixie cron but not with Anacron as used by Ubuntu.
- Ubuntu install docs for 10.04 have been updated.
- The example for using the **geoiplookup_continent** tool now shows how to specify the path to a GeoIP database.

12.9 Release 2.13.3 (r8166, Sep 26, 2010)

This is a release that fixes two important bugs in the Metalink generator. In addition, it includes a number of compatibility fixes for Torrents.

- **mod_mirrorbrain**:
  - The Magnet links embedded in Metalinks could cause the Metalink client **aria2c** to wait a long time on P2P connections, and not try the listed mirrors anymore (issue 73). These links are no longer included at the moment, unless `MirrorBrainMetalinkMagnetLinks On` is set in the Apache configuration.
  - Under the conditions that
    - an `Accept` header with `application/metalink+xml` or `metalink4+xml` is sent,
    - and the request goes to a path that doesn't exist,
    - but some extension (`.foo`) could be split off,
    - and a corresponding path without extension exists,

    mod_mirrorbrain delivered the file matching the path with the extension split off, instead of replying with a `404 Not found`. This affected **aria2c** when it requested non-existing files. The bug was found and fixed by Michael Schröder and closes issue 75.
  - When generating Torrents, the order of keys was not obeyed, which should be lexicographical. This is now the case, so the Torrents should be valid also for clients that insist on correct ordering. This should improve the compatibility to some clients, notably **rtorrent**. Tracked in issue 74 and issue 78.
  - The MD5 sum in Torrent info hashes was wrongly sent in binary form, instead of being hex-encoded. In addition, the key was wrongly named `md5` while `md5sum` is the correct name. Fixing issue 77.
  - Not a bugfix, but a hopefully useful addition is that Torrents now contain a "created by" key, indicating the generator of the torrent, and the version number (e.g. `MirrorBrain/2.13.3`). Suggested in issue 65.

Please read the 2.13.3 upgrade notes before upgrading.

Thanks for all kind help and contribution!

12.10 Release 2.13.2 (r8153, Sep 19, 2010)

This release adds worthwhile new features to the mirror list generator that you will enjoy:

- **mod_mirrorbrain**:
  - The content of the mirror lists (details pages) are now wrapped into a XHTML/HTML `DIV` container with `id="mirrorbrain-details"`. This improves the possibilities for styling in conjunction with a stylesheet linked in via the `MirrorBrainMirrorlistStyleSheet` directive (issue 63).
  - Further individual design can now be achieved by specifying the XHTML/HTML header and footer which are placed around the page body instead of the built-in XHTML (issue 63). This is configured with two new Apache configuration directives.

  This is documented here: *Styling the mirrorlist / details pages*.

  - Hashes can now be requested without a filename being included in the response, to simplify parsing (issue 68). This is done by sending the query string `only_hash`. This works with different ways to request a hash:

```
http://host.example.com/foo.md5?only_hash
http://host.example.com/foo?md5&only_hash
```

    Instead of `99eaed37390ba0571f8d285829ff63fc du.list`, the server will just return `99eaed37390ba0571f8d285829ff63fc`.

  - The filename in hashes can also be suppressed site-wide (and therewith, on the server side) with a new Apache config directive `MirrorBrainHashesSuppressFilenames On`. It goes into virtualhost context.
  - When sending out a hash to a client (as requested by appending e.g. `.md5`), there is now a *double* space between hash and filename – just like as the familiar tools like **`md5sum`** and **`sha1sum`** do it. This should avoid confusion and extra effort in parsing.
  - The mirror list's content type header now comes with UTF-8 as character set, instead of ISO-8859-1, which should make more sense.

- **`mb export --format=mirmon`**:
  - Exporting a mirror list for mirmon has been adjusted to the default in mirmon-2.3 of its option `list_style=plain`. The other format (`list_style=apache`) can also be generated, if mb export is used with `--format=mirmon-apache`. This fixes issue 62.

    The documentation *Exporting in mirmon format* has been updated to reflect this.

12.11 Release 2.13.1 (r8136, Sep 18, 2010)

This is a minor release, adding some improvements and fixing a bug that sneaked into the last release.

- **`mb edit`**:
  - A problem was fixed that made it impossible to remove an URL by setting it to an empty string. The fix for issue 30 was the culprit. This was a regression that came with the last release (2.13.0).
- **`mb list/edit/show/...`**:
  - In some situations, the fuzzy-matching on mirror identifiers made it impossible to select certain mirrors. Phillip Smith reported this issue and submitted a clever patch, which retains the convenient behaviour, but also allows for selection mirrors by their full name. This fixes issue 61.
- **`mb scan`**:
  - Scanning lighttpd web servers is now supported. Thanks to patch contributed by Phillip Smith. This fixes issue 60.
- Changes regarding packaging:
  - Thanks to the work of Phillip Smith, there are now packages for Arch Linux and the ArchServer distribution.
  - On Debian and Ubuntu, the mirrorbrain user and group are now automatically created by the package, as well as /var/log/mirrorbrain. This simplifies the installation procedure and fixes issue 4.
  - Thanks to the help of Cory Fields, the 2.12 -> 2.13.0 upgrade now works seamlessly on Debian/Ubuntu. Fixing issue 57.

12.12 Release 2.13.0 (r8123, Sep 6, 2010)

This is a big release, with many new features, and lots of bugs fixed. Big effort has also been put in to ensure a seamless upgrade.

Please read the 2.13.0 upgrade notes.

New features:

- This release **fully supports IETF Metalinks**, as finalized in **RFC 5854** early in 2010. The extension `.meta4` triggers the IETF Metalink response. An HTTP Accept header containing `metalink4+xml` also elicits this kind of response. This closes issue 14. The old (v3) Metalinks are still supported, and transparent content negotiation (TCN) is supported with both variants.
- As the cache of hashes needed to be restructured for this feature, it became possible to implement a number of additional features. Inclusion of **various metadata in the mirror lists** is supported now (issue 41):
  - file size and modification time
  - SHA256 hash
  - SHA1 hash
  - MD5 hashes
  - BitTorrent infohash
  - link to Metalink
  - link to Torrent
  - zsync link
  - Magnet link (needs testing)
  - link to PGP signature (if available)

  These metadata pages resp. mirror lists can now be requested by appending `.mirrorlist` to an URL. The previous way, using a question mark (`&mirrorlist`) continues to be supported for backwards compatibility.
- Thus, MirrorBrain is now a feature-rich **hash/metadata server**. A so-called "top hash" (cryptographic hash of the complete file) can now be requested. Depending on the extension added to the URL, like `.md5`, `.sha1`, or `.sha256`, the respective representation is returned. This closes issue 42.

  Like before, MirrorBrain also stores piece-wise hashes for chunks of the files. The chunk size is now configurable via `/etc/mirrorbrain.conf`, see *Generating Torrents*.

  All hashes are now stored in the database. (See *Database hash store* design notes.)

  A fallback mechanism is in place to read existing hashes from disk, if the database doesn't have the new hashes yet (useful for the migration period).
- Even though more hashes are calculated, and hashes stored in the database, hashing is **twice as fast** as before, not relying the external metalink binary any longer. All functionality of the **`metalink-hasher`** tool has been integrated into **`mb makehashes`**, which makes sure to never read data from disk more than once, regardless of how many hashes are calculated.

  The external tool names **`metalink`** is no longer used, and the package dependency on the **`metalink`** package is no longer there.
- MirrorBrain now has a **torrent generator embedded**. Torrents are generated in realtime (from hashes cached in the database). See *Generating Torrents* for details. This resolves issue 37.

- MirrorBrain now has basic **zsync support**. The zsync distribution method is rsync over HTTP, so to speak, and MirrorBrain can generate zsync files on-the-fly. MirrorBrain supports the simpler variant which doesn't look into compressed content. It is compatible to the current zsync release (0.6.1).

  See *Configuring zsync support* for details.

  This feature is off by default, because Apache allocates large amounts of memory for large rows from database; this may be worked around in the future.
- Initial support for Magnet links. This largely closes issue 38, but requires further testing/finetuning. See *Magnet links* for documentation.
- Ubuntu 10.04 (Lucid) support! (Issue 6 had to be fixed for this.)

While these are the main news, there is a number of smaller feature updates to be listed:

- **`mb makehashes`**:
  - This is the new tool for hashing files. It supersedes the previously used **`metalink-hasher`** and the external **`metalink`** tool.
  - **`metalink-hasher`** is a wrapper now, for backwards compatibility, to avoid breaking existing setups.
  - A `--force` option has been added to force refreshing existing hashes.
  - The usage example with `--base-dir` has been improved.
- **`mb list`**:
  - A new option `-N|--number-of-files` has been added, which displays the number of files that a mirror is known to have.

    To achieve this, a new stored procedure `mirr_get_nfiles()` has been implemented, which retrieves this number, given either a mirror id or its name. It is added automatically when migrating from previous versions, and made available in through the `mb.core.mirror_get_nfiles` method.
  - `mb list <mirror identifier>` did not work due to a missing module import in the Python script. This has been amended.
- **`mb update`**:
  - This command can now also update country & region info in mirror records (from GeoIP). Before, it updated only the network prefix and AS number, and geographical coordinates. But country and region assignments occasionally change as well.
  - A `--dry-run` option has been added, to allow seeing the changes before applying them.
  - An `--all` option has been added, which updates all metadata, same as when giving `-c -a -p --country --region` all at once.
  - The command now properly takes notice of hostnames that don't resolve in the DNS (so further action cannot be taken).
- **`mb db sizes`**:
  - The output of this command now includes also the size of the new hashes table.
- **`mb db vacuum`**:
  - The database cleanup now takes into account that files in the filearr table might not exist on any mirror, but only locally - so they could be referenced in the hash table.
- **`mod_mirrorbrain`**:
  - There is an additional logging handle which provides details about the request and the response. The Apache module takes note in the subprocess environment what the client requested and which representation of the

file was actually sent as response. Those variables can be used for logging with standard Apache CustomLog configuration with e.g. `want:%{WANT}e give:%{GIVE}e`.

- `mod_autoindex_mb`:
  - The link "Metalink" is no longer displayed. Instead, the link "Mirrors" has been renamed to "Details".

Bug fixes:

- `mod_mirrorbrain`:
  - When a client IP's network prefix did not match a mirror's network prefix exactly, the assignment of the client to this mirror would fail, even though the client IP was (also) contained in the mirror's network prefix. This has been rectified by properly checking for containment of the IP, fixing issue 52.
  - Requests with PATH_INFO were not ignored, as they should be. The default behaviour of Apache is to ignore such requests, and CGI or script handler deviate from that. **`mod_mirrorbrain`** now also correctly returns `404 Not Found` for such requests. This fixes issue 18, as well as openSUSE bug #546396 (which is not publicly readable).
  - When the only available mirror(s) had a limitation flag set (such as `region_only`), and a metalink was transparently negotiated, an empty metalink would result. This is now prevented, and the file delivered directly instead. Other representations (mirror lists, non-negotiated metalinks, torrents, hashes) are generated also if there is no mirror. This was tracked in openSUSE bug #602434. The mirrorlist is improved when there's no mirror, and can still list all hashes, and give the direct download URL.
  - The module now works when the path used in the Apache <Directory> block contains symlinks, fixing issue 17.
  - Errors from the database adapter (lower DBD layer) are now resolved to strings, where available.
  - Some variable types have been corrected from int to `apr_off_t`, using `apr_atoi64()` instead of `atoi()`. This applies to: `min_size`, `file_maxsize`, and the database identifier of a hash row. This at least fixes the info message given when a file is excluded from redirection due to its size. The checks seemed to work nevertheless, because the `min_size` numbers were small and `file_maxsize` numbers large, which helped to get the correct result when comparing.
- `mb scan`:
  - Usage of FTP authentication was fixed (with credentials encoded into the URL). The change done in January http://svn.mirrorbrain.org/viewvc/mirrorbrain/trunk/tools/scanner.pl?r1=7911&r2=7945 was incomplete in so far that the FTP client used a wrong path now when cd'ing into a directory (complete URL instead of only the path component). This may have worked with some FTP servers, but it definitely didn't work with vsftpd. Thanks to Deepak Gupta for raising this issue and providing means to analyse it.
  - When using the scanner with `--enable`, to enable a mirror after scanning, it was counter-intuitive that the redirection to the mirror was not immediately happening. The mirrorprobe first needs to mark the mirror online. The scan tool now does this right away. This issue (issue 59) had repeatedly puzzled people.
- `mb edit`:
  - Problems that occurred when copying and pasting data on the editing window have been fixed (reported in issue 30).
- `mirrorprobe`:
  - A hard-to-catch exception is now handled. If Python's socket module ran into a timeout while reading a chunked response, the exception would not be passed correctly to the upper layer, so it could not be caught by its name. We now wrap the entire thread into another exception, which would otherwise be bad practice, but is probably okay here, since we already catch all other exceptions. This should fix issue 46.
  - In case of exceptions we run into, allow logging the affected mirror's name.

   - If an unhandled exception occurs, a note is printed.

- `null-rsync`:
   - Broken links that are replaced by a directory, and point outside the tree, are now correctly removed in the destination tree. (A very special case.)
   - Some error messages were improved.

Internal changes:

- `mod_mirrorbrain`:
   - Code was generally cleaned up and logging improved.
   - A hex decoder for efficient handling of binary data from PostgreSQL was added.
   - Old obsolete code has been removed, which was needed before 2009 when mod_geoip didn't support continent codes yet. Since then, compiling with GeoIP support built-in was still optionally possible, but this old code is now removed.
   - The code path has been cleaned up a lot for easier handling of different representation, like hashes that are requested.
   - The message which is logged when no hashes where found in the database has been enhanced.
   - The obsolete support for generation of plaintext mirror lists (application/mirrorlist-txt) has been removed.
- `mb`:
   - Interruptions by Ctrl-C and various other signals are now properly caught.
   - The error classes have been revamped and modernized for Python 2.6.
   - The script mirrordoctor.py has been renamed to mb.py, in order to avoid confusion. The tool should now be installed with its own name now, and no further symlinking is needed upon installation.
- `mb makehashes`:
   - Hashes are also stored for files which exists only locally, and not on any mirror (and which weren't present in the `filearr` table yet, therefore). The cleanup mechanism had to be reworked to take this into account.

Documentations improvements:

- The installation docs have been restructured: Now there's a new section explaining the *Initial configuration steps on all platforms*, and this part is linked from all platform-specific sections as "next step" at their end. This should avoid some confusion. Hand in hand with this change, a cleanup of things scattered in all places is in progress.
- A few hints about *Tuning PostgreSQL* were added to the *Tuning guide*.
- *Setting up logging* is described in more detail.
- Notes about the necessity of *Creating a file tree* have been added, and alternatives explained.
- Reasons why or why not to use mod_asn are discussed in *Installing mod_asn*.
- Installing from Debian packages: There is now a note about expired keys, and how to renew them.
- The obsolete MySQL database schema has been removed, which could theoretically be useful for people aiming to run only mod_mirrorbrain, but not the rest of the framework - but is confusing and may cause people assume that MySQL is supported as backend.

Other improvements:

- `rsyncinfo`:

  This script is easier to use now. Instead of the arkward syntax it now also takes simple rsync URLs. Before:

```
rsyncinfo size gd.tuwien.ac.at -m openoffice
```

Now:

```
rsyncinfo size gd.tuwien.ac.at::openoffice
rsyncinfo size rsync://gd.tuwien.ac.at/openoffice
```

- `bdecode`:

  A new tool bdecode to parse a Torrent file (or other BEncoded input), and pretty-print it. Useful mainly to work on the Torrent generator in mod_mirrorbrain, but also to compare the generated torrents with torrents that you get from other generators. The tool can take an argument, or read from standard input:

  ```
  bdecode foo.torrent
  curl -s <url> | bdecode
  ```

Please read the 2.13.0 upgrade notes before upgrading.

Thanks for all the help!

12.13 Release 2.12.0 (r7957, Feb 10, 2010)

This release contains several important bug fixes, a new feature, and documentation fixes.

The new feature is that geographical coordinates of mirrors are stored. This affects newly created mirrors, as well as mirrors whose metadata is updated with **`mb update -c`**. The data are obtained from the GeoIP database, if available. Note that only the GeoIP city (lite) database contains this kind of data. The coordinates aren't used for anything yet, but it's easily possible now to display mirrors on a map, or to use them to aid mirror selection (which seems helpful in some cases; see issue 34 for a proposal).

For that, **`mb update`** got a new option `--coordinates` to insert (or update) geographical coordinates in the mirror's database records. The command can be used to add the data to existing mirrors. Just use `mb update --coordinates --asn --prefix` to update all mirror records with the coordinates, as well as refreshing asn and prefix data.

Bug fixes:

- `mb scan`
  - If **`rsync`** is 3.0.0 or newer, **`mb`** now uses the `--contimeout` option in addition to `--timeout`. This fixes issue 12, where problems during opening the connection could lead to an infinite hang, because that period isn't covered by rsync's `--timeout` option. The additional option to configure this timeout became available with rsync 3.0.0.
  - Scanning with FTP authentication has been implemented (URLs in the format *ftp://user:pass@hostname/path*).
- `mb mirrorlist`
  - When generating mirror lists, authentication data (in the form of *user:password@*) is now removed from URLs. The assumption is that if URLs contain such data, it will almost surely be not the intention to publish them.
- `mod_mirrorbrain`
  - On some platforms, **`mod_mirrorbrain`** didn't construct proper filenames for the metalink hash cache. The bug was reported for Debian Lenny, and probably also affected some version of Ubuntu (issue 35). This is fixed by using the APR library function `apr_off_t_toa()` instead of `%llu` in the format string fix. Thanks Cory for reporting and tracking this down!

  - When Metalinks contained FTP URLs, the URL scheme (`url type` in the XML) was incorrectly set to `http`. (issue 23). This has been fixed.
- **`mb db shell`**
  - This new command to spawn a database shell turned out to work only by accident – `os.execlp()` was used wrongly (missing its 0th argument). This has been correected.
- **`mb file ls -u`**
  - When using the `-u` option with this command to display URLs, broken URLs could result if a base URL doesn't end in a slash (issue 36). Thanks Vittorio for reporting!
- **`mb new`** and **`mb update`**
  - A stupid error in the selection of the best GeoIP database has been fixed. A forgotten *break* in the code caused the least preferable database to be chosen, of more than one acceptable database file was available.
  - Geographical coordinates are saved to mirror database records.
  - The readability of DNSrr warnings is improved.

Since when the metalink hash cache had been reimplemented with release 2.10.0 and 2.10.1, there remained a migration path in **`mod_mirrorbrain`** and **`metalink-hasher`** for reusing the existing hash files. Since this is several versions away (or 5 months), this migration path has been cleaned up in both **`mod_mirrorbrain`** and **`metalink-hasher`**.

- Backward compatibility and migration support (added around r7794) for old filename scheme (`.inode_$INODE`) in the metalink hash cache removed.
- Backward compatibility (added in r7787) for old filename scheme (`.metalink-hashes`) in the metalink hash cache removed.

When updating from an installation older than 2.10.1, that is no problem – it just means that metalink hashes will be regenerated before they can be used again.

The documentation was enhanced in the following places:

- A few examples for using cURL for testing have been added.
- The example for creating metalink hashes was wrong. This was fixed, and some more details added.
- The usage info of **`mb update`** was improved.
- The **`mb update`** command has been documented (*Editing a mirrors network location*).

12.14 Release 2.11.3 (r7933, Dec 16, 2009)

This release contains a number of small improvements in the toolchain, plus small documentation fixes.

- **`null-rsync`**:
  - IO errors returned by rsync are handled now
  - remote errors from rsync are ignored now, and we let rsync continue with dry-run deletions.
- **`mb db sizes`**:
  - Sizes of tables from mod_stats are now shown in addition to MirrorBrain's own tables.
- **`mb db shell`**:
  - The script now uses `os.execlp()` instead of `os.system()` to spawn the database commandline interpreter, because the latter doesn't reliably pass `SIGCONT` to the subprocess when resuming.
- **`mb list`**:

   - New options `-H`, `-F`, `-R` to display HTTP/FTP/rsync base URLs have been added.
- **`mb mirrorlist`**:
   - The script now tries harder to not leave temp files – also in case of a crash (which may happen when working with templates).
   - Add a link to our project in the footer.

Changes in the documentation were:

- The new `MirrorBrainFallback` directive is now documented in the example `mod_mirrorbrain.conf`.
- The `-t 20` option has been removed from the **`mirrorprobe`** call, since that is the default now. The scan cronjob also has been simplified.
- A hint about ulimits has been removed, which turned out to be a band-aid for a purely local problem.
- A hint how to load a database dump with **`mb db shell`** has been added.

12.15 Release 2.11.2 (r7917, Dec 5, 2009)

This release improves scanning via FTP and adds a few small features:

- **`mb scan`**:
   - When scanning via FTP, filenames containing whitespace would not be recognized. The regular expression that parses the FTP directory listing has been extended. In addition, a warning is now printed when a line can't be parsed. This hopefully fixes issue 31.
   - when using the FTP protocol for probing for a file or directory, the wrong use of a variable let the result always be negative. This affected subdirectory scans (using `mb scan -d path/to/dir`), which would igore some mirrors.
- **`mb db`**:
   - new command for database maintenance tasks:
      * **`mb db sizes`** — shows sizes of all relations
      * **`mb db shell`** — conveniently open a shell for the database
      * **`mb db vacuum`** — cleans up dead references (previously: **`mb vacuum`**, which still can be used for backwards compatibility.)
- **`mirrorprobe`**:
   - 60 seconds as timeout have always been a bit long. Change the default timeout to 20 seconds, which is also the value suggested in the documentation.

12.16 Release 2.11.1 (r7899, Dec 3, 2009)

This release fixes a regression in **`mod_mirrorbrain`** that was introduced with the 2.11.0 release. It affected Debian and Ubuntu, or more generally all platforms where the APR (Apache Portable Runtime) is version 1.2, not 1.3. The version detection at compile time was not working. This has been corrected, fixing issue 29. Thanks to Cory Fields in tracking down this bug!

12.17 Release 2.11.0 (r7896, Dec 2, 2009)

A new feature and lots of bug fixes and minor corrections come with this release.

It's now possible to configure fallback mirrors, via Apache config, in the following form:

```
MirrorBrainFallback na us ftp://linuxfreedom.com/ultimate/
MirrorBrainFallback eu de http://www.ultimate-edition.org/~ue/
```

Those mirrors are used when no reachable mirror is found in the database. Thus, these mirrors get all those requests that MirrorBrain would normally deliver itself (you know, the default fallback behaviour).

They are also used in the mirror lists (with priority 1) and metalinks, and country/region selection is done like for normal mirrors. They are used blindly, without knowing their file lists.

This actually allows to run a MirrorBrain instance with a pseudo file tree (cf. recently added **null-rsync** script.)

A "degraded mode" that continues to work in case of database complete outages is easily achievable now, however for now the code path is less robust in that regard (*if* fallback mirrors are configured. Otherwise, it shouldn't). This should be fixed later.

This new feature is still its infancy, but ready to be tested. It may be subject to refinement, based on future discussion.

- Other changes in **mod_mirrorbrain** are:
  - The module now automatically makes sure at compile time that its usage of the DBD database API fits to the APR (Apache Portable Runtime) version. The issue was that the semantics of reading result rows was with APR 1.3. With older APR, different semantics need to be used, which hits Debian and Ubuntu. This fixes issue 7.
  - The `MirrorBrainHandleDirectoryIndexLocally` directive has been removed. It was never actually useful, because we never did (and could) redirect to directory listings. For one, a listing might not be available at each URL that we might redirect to. What's more, since the database only stores file paths and not directories, we can't actually look up directories. Thus, the directive is now removed, and a warning issued where it is still found in the config.
  - The default of `MirrorBrainHandleHEADRequestLocally` has been changed to `Off`, and it has been made clearer (in the Apache-internal help text) what the default is. This change mainly has the effect that the directive does *not* need to be given anymore, in most scenarios.
  - The default setting of the `MirrorBrainMinSize` directive has been documented in its help text.
- The documentation for installation on Debian Lenny was tested and corrected where needed. Thanks, TheUni! Minor issues in the Debian packages have been improved, to further simplify the installation. Ubuntu benefits from this as well.
- **mb**
  - Parse errors in the configuration file are not caught and and reported nicely.
  - Special characters occurring in the password are escaped before passing them to SQLObject/psycopg2, thus fixing issue 27. A remaining issue is that double quotes can't be used; a warning is issued if it's attempted.
- **mb scan**:
  - A warning that appeared since the last release has been removed. It was caused by the removal of obsolete code, and purely cosmetic.
- **null-rsync**
  - An `--exclude` commandline option has been implemented, to be passed through to **rsync**.
  - Control over the program output can now be exerted by the two new options `--quiet` and `--verbose`.

   - Usage info is implemented (`--help` etc.).
   - Interruptions by `Ctrl-C` and similar signals are intercepted now.

- **`metalink-hasher`**
   - When comparing the modification time of a saved metalink hash with that of a source file, the sub(sub-)second portion of the value could be different from the value that has just been set by `os.utime()`. (Quite surprisingly.) So now, we compare only the `int()` portion of the value. This fixed issue 24.

12.18 Release 2.10.3 (r7871, Nov 28, 2009)

This release adds a new script, which hopefully opens up interesting new use cases, called **`null-rsync`**. This is a special rsync wrapper which creates a local file tree from a mirror, where all files contain only zeroes instead of real data. The files are created as *sparse files*, so only the metadata occupies actual space in the filesystem. Modification times and sizes are fully copied, so that even (native) rsync thinks that the file tree is identical.

This script should allow to create a pseudo mirror of arbitrary size (or several mirrors), in order to host MirrorBrain instances which run under the precondition that they *always* redirects. (This scenario hasn't tested yet, but should work.) At any rate, it is a good basis for experimentation.

Then, this release fixes some usability issues in the **`mb`** tool:

- **`mb new`**:
   - when creating a new mirror, and detecting that the hostname resolved to multiple addresses (round-robin DNS), a warning about this fact was issued. Now, (short of documentaion in the manual) a reference to http://mirrorbrain.org/archive/mirrorbrain/0042.html is added, where the issue has been discussed in depth.
   - A proper error message is now shown if an identifier is chosen that already exists.
- **`mb mirrorlist`** / **`mb marker`**:
   - The order in which mirrorlist columns are presented is now kept unchanged, so it appears as it was entered into the database.
   - The sort order of mirrorlist entries has been improved. Instead of the priority, the mirror operator name is now given precendence in order, which results in a mirror list that actually *looks* sorted.

12.19 Release 2.10.2 (r7853, Nov 9, 2009)

Some non-code changes that should be mentioned:

- The documentation was updated in various places. Notably, there are now instructions for *Installation on Debian/Ubuntu Linux*, which David Farning deserves credits for.
- Ubuntu (and Debian) packages have been created. The Ubuntu packages have been tested successfully. (See download page.)
- A bug tracking system has been set up: http://mirrorbrain.org/issues/

In the code, the following bugs were fixed:

- The **`mirrorprobe`** could crash when the sender domain of a configured mail log handler wasn't resolvable (issue #9). This has been fixed.
- When scanning a subdirectory, the mirror scanner (**`mb scan`**) could accidentally delete files from the database outside of that directory. This was caused by lack of terminatation (with a slash) of the path expression that is used to grab the list of known files before the scan. Herewith fixing issue #19.

- A misleading error message in the `mb` tool was improved, which was issued when encountering config file with missing sections.

12.20 Release 2.10.1 (r7798, Sep 9, 2009)

- The implementation of the hash cache created by the `metalink-hasher` tool has been revised again. The reason is that some filesystems (at least the VirtualBox Shared Folder) don't implement stable inode numbers. Instead of the inode number, now the file size (plus filename and modification time) is used to identify file hashes. (These are the same criteria that rsync uses, by the way.)

  Existing hashes are migrated, so that the files don't need to be hashed again (which could potentially be time-consuming).

  The modification time of files is now copied to the hash file, so it is available for comparison when checking if a hash file is up to date.

  `mod_mirrorbrain` has been adapted for the new cache scheme. Also, it is now required that the modification time of the hash file matches the modification time of the file. (For backwards compability, the module still also checks for files matching the old scheme.)

  To ease the migration, and since it doesn't matter otherwise, non-existance of files to be unlinked is ignored now. This occurs for instance in the above mentioned migration scenario, where the hash files are renamed to a different name.

- New features in the `metalink-hasher` tool:

  - Per-directory locking was implemented: directories where already a job is running will be skipped. This allows for hassle-free parallel runs of more than one job.

    Note that simultaneous spawning of the script still needs to be controlled, to avoid consuming too much I/O or CPU bandwidth for a machine.

  - Ctrl-C key presses and common interrupting signals are now handled properly.

12.21 Release 2.10.0 (r7789, Sep 4, 2009)

- The cache of metalink hashes, as created by the `metalink-hasher`, was changed to more reliably detect changes in the origin files. So far, the file modification time was the criterion to invalidate cached hashes. When files were replaced with *older* versions (version with smaller mtime), this wasn't detected, and a cached hash would not be correctly invalidated. https://bugzilla.novell.com/536495 reports this of being an issue.

  To fix this, the cache now also uses the file inode as criterion.

  `mod_mirrorbrain` was updated to use the new inode-wise metalink hashes. At the same time, it still knows how to use the previous scheme as fallback. If the new-style hash isn't found, it looks for the old-style hash file.

  Thus, the transition should be seamless, and no special steps should be required when upgrading. Note however that all hashes are regenerated, which could take a while for large file trees, and which could lead to cron jobs stacking up.

- There were a number of enhancements, and small bug fixes, in the `mb` tool (and accompanying Python module):

  - `mb new`:

    * When adding new mirrors, the hostname part in the HTTP base URL might contain a port number. This is now recognized correctly, so the DNS lookup, GeoIP lookup and ASN lookup for the hostname string can work.

    * The commandline options `--region-only`, `--country-only`, `--as-only`, `--prefix-only` were added, each setting the respective flag.
    * The commandline options `--operator-name` and `--operator-url` were added.
    * The `--score` option is depreciated, since it has been renamed it to `--prio`.
  - **`mb scan`**:
    * The passing of arguments to the scanner script was fixed in the case where the `-j` (`--jobs`) option was used together with mirror identifier specified on the commandline.
  - **`mb list`**:
    * Command line options to display the boolean flags were added: `--region-only`, `--country-only`, `--as-only` and `--prefix-only`.
  - **`mb scan`** and **`mb file ls --probe`**:
    * the lookup whether the `multiprocessing` or `processing` module exist was fixed: it could print a false warning that none of them was installed.
- The **`mirrorprobe`** program no longer logs to the console (stderr). This allows for running the script without redirection its output to `/dev/null` — which could mean swallowing important errors in the end.

  A scenario was documented where the mirrorprobe could fail on machines with little memory and many mirrors to check. The fix is to properly set ulimits to allow a large enough stack size.

  Error handling was cleaned up; more errors are handled (e.g. socket timeouts during response reading) and logged properly; and for exceptions yet unhandled, info about the mirror that caused them is printed.

12.22 Release 2.9.2 (Aug 21, 2009)

- Most work happened on the documentation, which includes
  - more installation instructions,
  - directions for upgrading,
  - some tuning hints,
  - a quite complete walkthrough through the usage of the **`mb`** commandline tool to maintain the mirror database,
  - instructions how to set up change notifications (*Exporting to a Version Control System (VCS)*)
  - list of known problems, and these release notes.

  The documentation is in the `docs` subdirectory, as well as online at http://mirrorbrain.org/docs/.

  Notably, there is a new section *How to improve this documentation*, which explains *how* to work on the docs.
- New features:
  - **`mb export`** can now generate a mirmon mirror list. Thus, it is easy to deploy mirmon, automatically scanning the mirrors that are in the database. See *Exporting in mirmon format* for usage info.
  - In **`mod_autoindex_mb`**, displaying the "Mirrors" and "Metalink" links was implemented for configurations with Apache's `IndexOptions HTMLTable` configured.
- Two minor bugs were fixed:
  - Missing slash added in **`mod_autoindex_mb`** to terminate the XHTML `br` element in the footer.
  - The scanner now ignores rsync temp directories (`.~tmp~`) also when they occur at the top level of the tree, and not below.

12.23 Release 2.9.1 (Jul 30, 2009)

- **`mb new`**
  - Now an understandable error message is printed when the geoiplookup_continent couldn't be executed. Thanks to Daniel Dawidow for providing helpful information to track this down.
- **`mod_mirrorbrain`**
  - Under unusual circumstances it may happen that mod_mirrorbrain can't retrieve a prepared SQL statement. This occurs when an identical database connection string is being used in different virtual hosts. To ease tracking down this special case, the module now logs additional information that could be useful for debugging. Also, it logs a hint noting that connection strings defined with DBDParams must be unique, and identical strings cannot be used in two virtual hosts.
- The **`mod_mirrorbrain`** example configuration files were updated to reflect several recent (or not so recent) changes:
  - the switch to PostgreSQL
  - the now disabled memcache support
  - the updated GeoIP database path (/var/lib/GeoIP instead of /usr/share/GeoIP)

12.24 Release 2.9.0 (Jul 28, 2009)

- A very hindering restriction in the **`mb`** tool which made it require mod_asn to be installed alongside MirrorBrain has been removed. MirrorBrain can now be installed without installing mod_asn.
- The Subversion repository was moved to http://svn.mirrorbrain.org/svn/mirrorbrain/trunk/.
- rsync authentication was fixed. Credentials given in rsync URLs in the form of `rsync://<username>:<password>@<host>/<module>` now work as expected. Patch by Lars Vogdt.
- The documentation has been moved into a docs subdirectory, and is rewritten in reStructured Text format, from which HTML is be generated via Sphinx (http://sphinx.pocoo.org/). Whenever the documentation is changed in subversion, the changes automatically get online on http://mirrorbrain.org/docs/
- Parallelized mirror probing. Note: for this new feature, the Python modules `processing` or `multiprocessing` need to be installed. If none of them is found, the fallback behaviour is to probe serially, like it was done before. This new feature affects the **`mb probefile`** and **`mb file`** commands, and not actually the mirrorprobe, which has always ran threaded. It also affects the scanner (**`mb scan`**) to speed up the checks done when only a subdirectory is scanned.
- Various new features were implemented in the **`mb`** tool:
  - **`mb probefile`**
    - Implemented downloading (and displaying) of content.
    - A `--urls` switch was added, to select the kind of URLs to be probed.
      - `--urls=scan` probes the URLs that would be used in scanning.
      - `--urls=http` probes the (HTTP) base URLs used in redirection.
      - `--urls=all` probes all registered URLs.
    - The usual proxy environment variables are unset before probing (`http_proxy`, `HTTP_PROXY`, `ftp_proxy`, `FTP_PROXY`)
    - Report the mirror identifier for FTP socket timeouts

    - `mb scan`
        * Logging output was considerably improved, avoiding lots of ugly messages which look like real errors (and tend to cover real ones)
        * The time that a scan took is now shown.
    - `mb new`
        * while looking up a mirror's location when a new mirror is added, try different geoip database locations (GeoIP database was moved around on openSUSE…).
        * prefer the larger city lite database, if available, and prefer updated copies that were fetched with the **geoip-lite-update** tool.
    - `mb list`
        * add `--other-countries` option to allow displaying the countries that a mirror is configured to handle in addition to its own country

- **mod_mirrorbrain**: in the `generator` tag of metalinks, include mod_mirrorbrain's version string
- The **metalink-hasher** tool has been revised to implement a number of lacking features:
    - Automatic removal of old hashes, which don't have a pendant in the file tree anymore, is implemented now.
    - A summary of deletions is printed after a run.
    - A number of things were optimized to run more efficiently on huge trees, mainly by eliminating all redundant `stat()` calls.
    - sha256 was added to the list of digests to generated.
    - The need to specify the `-b` (`--base-dir`) option was eliminated, which makes the command easier to use.
    - The order in which the tool works through the todo list of directories was changed to be alphabetical.
    - Using a Python `set()` builtin type instead of a list can speed up finding obsolete files in the destination directory by 10 times, for huge directories.
    - The program output and program help was improved generally.
    - Various errors are caught and/or ignored, like vanishing directories and exceptions encountered when recursively removing ignored directories.
    - The indentation of verification containers was corrected, so it looks sane in the metalink in the end.
    - The version was bumped to 1.2.
- **geoip-lite-update**: This tool to fetch GeoIP databases has been updated to use the path that's used in the openSUSE package since recently (`/var/lib/GeoIP`), and which complies better to the Linux Filesystem Hierarchy Standard. It still tries the old location (`/usr/share/GeoIP`) as well, so to continue to work in a previous setup.
- **mirrorprobe**
    - A logrotate snippet was added.
    - The mirrorprobe logfile was moved to the `/var/log/mirrorbrain/` directory.
- The openSUSE RPM package now creates a user and group named *mirrorbrain* upon installation. Also, it packages a runtime directory `/var/run/mirrorbrain` (which is cleaned up upon booting) and a log directory `/var/log/mirrorbrain`. Some additional Requires have been added, on the perl-TimeDate, metalink and libapr-util1-dbd-pgsql packages.

12.25 Release 2.8.1 (Jun 5, 2009)

- Python 2.6 compatibility fixes:
  - **`mb file ls`** `--md5` now uses the `hashlib` module, if available (this fixes a DepracationWarning given by Python 2.6 when importing the `md5` module).
  - **`mb list`**: The `--as` option had to be renamed to `--asn`, because `as` is a reserved keyword in Python, and Python 2.6 is more strict about noticing this also in cases where just used as an attribute.
  - The `b64_md5` function was removed, which was no longer used since a while.
- **`mb file ls`**
  - make the `--md5` option imply the `--probe` option
- **`mb export`**
  - when exporting metadata for import into a VCS (version control system), handle additions and deletions
- The docs were updated to point to new RPM packages in the openSUSE build service (in a repository named Apache:MirrorBrain). The formerly monolithic package has been split up into subpackages.
- perl-Config-IniFiles was added to the list of perl packages required by the scanner (**`mb scan`**)

12.26 Release 2.8 (Mar 31, 2009)

- Improvements in the scanner, mainly with regard to the definition of patterns for files (and directories) that are to be included from scanning. Old, hardcoded stuff from the scanner has been removed. Now, excludes can be defined in `/etc/mirrorbrain.conf` by the `scan_exclude` and `scan_exclude_rsync` directives. The former takes regular expressions and is effective for FTP and HTTP scans, while the latter takes rsync patterns, which are passed directly to the remote rsync daemon. See http://mirrorbrain.org/archive/mirrorbrain-commits/0140.html for details. This can decrease the size of the database (>20% for openSUSE), and for many mirrors it considerably shortens the scan time.
- Fixed a bug where the scanner aborted when encountering filenames in (valid or invalid) UTF-8 encoding. See https://bugzilla.novell.com/show_bug.cgi?id=490009
- Improved the implementation of exclusions as well as the top-level-inclusion pattern, which were not correctly implemented to work in subdir scans.
- The documentation was enhanced in some places.
- mod_autoindex_mb (which is based on mod_autoindex) was rebased on httpd-2.2.11.
- **`mb dirs`**: new subcommand for showing directories that the database contains, useful to tune scan exclude patterns.
- **`mb export`**: implement a new output format, named `vcs`. Can be used to commit changes to a subversion repository and get change notifications from it. See http://mirrorbrain.org/archive/mirrorbrain-commits/0152.html
- Partial deletions (for subdir scans) have been implemented.
- **`mb list`** accept `--country` `--region` `--prefix` `--as` `--prio` options to influence which details are output by it.
- **`mb file`**: support for probing files, with optional md5 hash check of the downloaded content.
- The latter three changes have already been described in more detail at http://mirrorbrain.org/news_items/2.7_mb_toolchain_work

12.27 Release 2.7 (Mar 4, 2009)

- Completely reworked the file database. It is 5x faster and one third the size. Instead of a potentially huge relational table including timestamps (48 bytes per row), files and associations are now in a single table, using smallint arrays for the mirror ids. This makes the table 5x faster and 1/3 the size. In addition, we need only a single index on the path, which is a small and very fast b-tree. This also gives us a good search, and the chance to do partial deletions (e.g. for a subtree).
- With this change, MySQL is no longer supported. The core, mod_mirrorbrain, would still work fine, but the toolchain around is quite a bit specific to the PostgreSQL database scheme now. If there's interest, MySQL support in the toolchain can be maintained as well.
- many little improvements in the toolchain were made.
- Notably, the scanner has been improved to be more efficient and give better output.
- mirror choice can be influenced for testing with a query parameter (`as=`), specifying the autonomous system number.

12.28 Release 2.6 (Feb 13, 2009)

- supports additional, finer mirror selection, based on network topological criteria, network prefix and autonomous system number, using mod_asn and global routing data.
- updated database schemes and toolchain – PostgreSQL support is solid now
- work on installation documentation for both MySQL and PostgreSQL (the latter is recommended now, because it allows for nifty features in the future. The **`mb`** tool has an **`mb export`** subcommand now, perfect to migrate the database.)
- toolchain work

12.29 Release 2.5 (Feb 3, 2009)

- working on PostgreSQL support
- working on the INSTALL documentation
- scanner: 0.22
  - more efficient SQL statement handling
  - output much improved
  - added SQL logging option for debugging
- **`mb`** (mirrorbrain tool):
  - bugfix in the **`mb file`** command: make patterns work which have a wildcard as first character.
  - extend **`mb scan`** to accept `-v` and `--sql-debug` and pass it to the scanner

12.30 Release 2.4 (Jan 23, 2009)

- rename **`mod_zrkadlo`** to **`mod_mirrorbrain`**
- use mod_geoip for GeoIP lookups, instead of doing it ourselves. We can now use the GeoIP city database for instance
- handle satellite "country" called `A2`
- auto-reenable dead mirrors
- **`geoiplookup_city`** added, new tool to show details from GeoIP city databases
- **`geoip-lite-update`** tool updated, with adjusted URL for GeoLite databases. It also downloads the city database now.
- deprecate `clientip` query parameter, which can no longer work once we use mod_geoip. Implement `country` parameter that can be used instead.
- make memcache support optional at compile time

12.31 Release 2.3 (Dec 13, 2008)

- add commandline tool to edit marker files. (Marker files are used to generate mirror lists. Each marker file is used to determine whether a mirror mirrors a certain subtree.)
- improvements and few features in the toolchain:
  - the mirrorprobe now does GET requests instead of HEAD requests.
  - **`mb`**, the mirrorbrain tool, has a powerful **`mb probefile`** command now that can check for existance of a file on all mirrors, probing all URLs. This is especially useful for checking whether the permission setup for staged content is correct on all mirrors.
- new database fields: `public_notes`, `operator_name`, `operator_url`
- new database tables: `country`, `region`
- generate mirror lists

12.32 Release 2.2 (Nov 22, 2008)

- simplified database layout, with additional space save.

12.33 Release 2.1 (Nov 9, 2008)

- simplified the Apache configuration: It is no longer needed to configure a database query. At the same time it's less error-prone and avoids trouble if one forgets to update the query, when the database schema changes.
- specific mirrors can be now configured to get only requests for files < n bytes

12.34 Release 2.0 (Nov 3, 2008)

- implement better fallback mirror selection
- add **`mb file`** tool to list/add/rm files in the mirror database

12.35 Release 1.9 (Oct 26, 2008)

- add bittorrent links (to all .torrent files that are found) into metalinks
- embed PGP signatures (.asc files) into metalinks
- add configurable CSS stylesheet to mirror lists
- **`mod_zrkadlo`**:
  - implement the redirection exceptions (file too small, mime type not allowed to be redirected etc) for transparently negotiated metalinks.
  - add `Vary` header on all transparently negotiated resources.
  - allow to use the apache module and all tools with multiple instances of the mirrorbrain. Now, one machine / one Apache can host multiple separate instances, each in a vhost.
- new, better implementation of rsyncusers tool
- bugfixes in the scanner, mainly for scanning via HTML
- installation instructions updated
- a number of small bugs in the tools were fixed and several improvements added.
- added "mirrordoctor", a commandline tool to maintain mirror entries in the database. Finally!

12.36 Release 1.8 (Jun 2, 2008)

- mod_zrkadlo now uses mod_memcache for the configuration and initialization of memcache
- **`metalink-hasher`** script added, to prepare hashes for injection into metalink files
- **`rsyncusers`** analysis tool added
- **`rsyncinfo`** tool added
- scanner bugfix regarding following redirects for large file checks
- failover testbed for text mirrorlists implemented
- metalinks: switch back to **RFC 822** format
- new `ZrkadloMetalinkPublisher` directive
- fix issue with <`size`> element
- now there is another (more natural) way to request a metalink: by appending `.metalink` to the filename.
- change metalink negotiation to look for *`application/metalink+xml`* in the `Accept` header (keep `Accept-Features` for now, but it is going to be removed probably)

12.37 Release 1.7 (Apr 21, 2008)

- new terse text-based mirrorlist
- allow clients to use **RFC 2295** Accept-Features header to select variants (metalink or mirrorlist-txt)
- metalink hash includes can now be out-of-tree
- **`mod_autoindex_mb`** added
- adding a `content-disposition` header

12.38 Older changes

Please refer to the subversion changelog: http://svn.mirrorbrain.org/svn/mirrorbrain/trunk respectively http://svn.mirrorbrain.org/viewvc/mirrorbrain/trunk/

CHAPTER

THIRTEEN

INDICES AND TABLES

- genindex
- modindex

INDEX