

Dr. Peter Poeml

Novell / SUSE Linux AG <poeml at suse.de>

ApacheCon Europe 2008, Amsterdam





#### Introduction

About The Problem Approaches

#### An Implementation

Building Blocks Mirror Database The Mirrorlist Generator / Redirector

#### Case Study

download.opensuse.org What We Optimized Pros, Cons, Ideas



Introduction

- About



# Outline

#### Introduction About

The Problem Approaches

An Implementation Building Blocks Mirror Database The Mirrorlist Generator / Redirector

Case Study

download.opensuse.org What We Optimized Pros, Cons, Ideas







#### Myself:

- Have been working for SUSE/Novell since 2000
- Working on openSUSE.org download infrastructure
- openSUSE Build service
- Past projects:
  - Maintained Apache, OpenSSL, DHCP
  - Ported SUSE Linux to IBM iSeries





This Talk:

- Popularity of your software -> downloads -> too much traffic
- Ways to deal with the traffic
- How to make use of mirrors
- Show how openSUSE.org does it



-Introduction

- The Problem



# Outline

Introduction About The Problem Approaches

An Implementation Building Blocks Mirror Database The Mirrorlist Generator / Redirector

Case Study

download.opensuse.org What We Optimized Pros, Cons, Ideas



- Introduction

- The Problem



### A Flourishing Open Source Project

- Possibly large files (CD or DVD images)
- Different releases, subprojects, architectures, ...
- More downloads than you could ever handle



-Introduction

- Approaches



# Outline

## Introduction

About The Problem Approaches

An Implementation Building Blocks Mirror Database The Mirrorlist Generator / Redirector

Case Study

download.opensuse.org What We Optimized Pros, Cons, Ideas





Content Delivery Networks (CDN)

- Standard solution to the problem
- They do wide area load distribution, by adding intelligence to standard DNS
- They are expensive
- They hardly fit into the tight budget of an open source project



- Approaches



### Mirrors Come To Help!

- If what you do is popular, then probably somebody is mirroring you.
- They do it for their own benefit (saves their bandwidth)
- Only some do it to help your project
- You have no real control
- You can only facilitate



Introduction

Approaches



### Five Ways To Distribute Traffic To Mirrors

- 1. Static mirror lists
- 2. Dynamic mirror lists
- 3. Dynamic mirror lists, used to redirect transparently
- 4. Metalinks
- 5. Metalinks, used transparently



- Introduction

Approaches



Method 1: Static Mirror Lists

- Can be hard to maintain
- Often too static
- Can hardly ever be correct
- Low granularity
- Work well for small file trees



Approaches



#### Method 2: Dynamic Mirror Lists

- Mirror monitoring to increase correctness
- Automation allows for finer granularity
- Often combined with geolocation of clients
- User gets a suggestion, or needs to chose interactively
- Works well for downloads of single files
- Can be annoying, or lead to all users picking the same mirror
- Doesn't work so well for automated downloads



Approaches



## Method 3: Dynamic Mirror Lists, Transparent Redirects

- Mirror is selected automatically (server makes the choice)
- Client doesn't actually get the list
- User doesn't need to figure out
- More difficult for user to override choice
- Requires intensive mirror monitoring
- Good for machine clients



Approaches



#### Method 4: Metalinks

- Metalink: a mirror list in standardized, machine-readable format (metalinker.org)
- Needs a metalink-capable download client
- Includes hashes for transfer integrity checking
- The client can do automatic failover if one source doesn't work
- This makes downloads robust and fast
- Good for humans and machines



- Introduction

Approaches



### Method 5: Metalinks, Used Transparently

- Interesting, but no standard yet
- Transparent negotiation would be best
- A client which can accept metalinks would get a metalink
- A normal HTTP client would get a redirect







When huge amounts of content change rapidly,

- Mirrors have a hard time catching up
- Thus, you have to deal with partial mirrors

A strong reason for dynamic mirror lists and thourough mirror surveillance.



- Approaches



Now, I will show you an implementation which combines method 2, 3, and 4. It does

- transparent redirection
- dynamic mirror lists
- metalinks



- An Implementation

- Building Blocks

# ApacheCon Europe '08

# Outline

ntroduction About The Problem Approaches

## An Implementation

## **Building Blocks**

Mirror Database The Mirrorlist Generator / Redirector

Case Study

download.opensuse.org What We Optimized Pros, Cons, Ideas



Building Blocks



The building blocks of the framework are:

- Mirror database
- Mirrorlist generator and redirector
- Monitoring tools

I like to call the whole thing "mirror brain".



- An Implementation

-Building Blocks



- Other building blocks are the mirrors a heterogenous clique.
- If the mirroring machines are owned and controlled by yourself, all the better.



-Building Blocks



Technology

- Apache HTTP server 2.2
- DBD framework
- libGeoIP
- libapr\_memcache (from APR trunk)
- MySQL server
- Mirror monitoring tools in Python and Perl



- An Implementation

- Mirror Database

# ApacheCon Europe '08

# Outline

ntroduction About The Problem Approaches

## An Implementation

**Building Blocks** 

## Mirror Database

The Mirrorlist Generator / Redirector

Case Study

download.opensuse.org What We Optimized Pros, Cons, Ideas



-Mirror Database



The mirror database keeps an *inventory* of the mirrors, on file-level.

- It is acquired and updated by crawling the mirror via rsync, FTP or HTTP
- Mirrors are frequently probed for availability
- For large files, functional tests are useful (e.g., whether the mirror correctly sends files > 2GB and handles byte ranges)
- A "strength index" is assigned to each mirror, according to its capabilities
- Database design is such that a single SQL query is enough to retrieve the list of mirrors for a file



- An Implementation

- The Mirrorlist Generator / Redirector

# ApacheCon Europe '08

# Outline

ntroduction About The Problem Approaches

## An Implementation

Building Blocks Mirror Database

#### The Mirrorlist Generator / Redirector

ase Study download.opensuse.org What We Optimized Pros, Cons, Ideas



- The Mirrorlist Generator / Redirector



The Mirrorlist Generator / Redirector

- mod\_zrkadlo (*zrkadlo* = Slovakian for *mirror*)
- implemented as an Apache module in C
- hooks in as handler into the request processing phase
- thus fully integratable into other "jobs" of the webserver
- relies on the awesome, new DBD framework for database access
- (and thus needs Apache HTTP server 2.2.x)



- The Mirrorlist Generator / Redirector



- check if the requested file qualifies for redirection
- if not, the handler quits and lets the file be served directly
- canonicalize filename
- geolocate the client through its IP address
- search for possible mirrors in the database
- if no mirror was found, quit and let the file be served directly



- The Mirrorlist Generator / Redirector



- check if the requested file qualifies for redirection
- if not, the handler quits and lets the file be served directly
- canonicalize filename
- geolocate the client through its IP address
- search for possible mirrors in the database
- if no mirror was found, quit and let the file be served directly



- The Mirrorlist Generator / Redirector



- check if the requested file qualifies for redirection
- if not, the handler quits and lets the file be served directly
- canonicalize filename
- geolocate the client through its IP address
- search for possible mirrors in the database
- if no mirror was found, quit and let the file be served directly



- The Mirrorlist Generator / Redirector



- check if the requested file qualifies for redirection
- if not, the handler quits and lets the file be served directly
- canonicalize filename
- geolocate the client through its IP address
- search for possible mirrors in the database
- if no mirror was found, quit and let the file be served directly



- The Mirrorlist Generator / Redirector



- check if the requested file qualifies for redirection
- if not, the handler quits and lets the file be served directly
- canonicalize filename
- geolocate the client through its IP address
- search for possible mirrors in the database
- if no mirror was found, quit and let the file be served directly



- The Mirrorlist Generator / Redirector



- check if the requested file qualifies for redirection
- if not, the handler quits and lets the file be served directly
- canonicalize filename
- geolocate the client through its IP address
- search for possible mirrors in the database
- if no mirror was found, quit and let the file be served directly



- An Implementation

The Mirrorlist Generator / Redirector



- sort mirrors by closeness, strength and randomize a bit
- return one of the following:
  - a redirect (HTTP status code 302 Found and a Location: header)
  - sorted mirror list (if requested)
  - metalink (if requested)



- An Implementation

- The Mirrorlist Generator / Redirector



- sort mirrors by closeness, strength and randomize a bit
- return one of the following:
  - a redirect (HTTP status code 302 Found and a Location: header)
  - sorted mirror list (if requested)
  - metalink (if requested)



- An Implementation

- The Mirrorlist Generator / Redirector



- sort mirrors by closeness, strength and randomize a bit
- return one of the following:
  - > a redirect (HTTP status code 302 Found and a Location: header)
  - sorted mirror list (if requested)
  - metalink (if requested)



- An Implementation

- The Mirrorlist Generator / Redirector



- sort mirrors by closeness, strength and randomize a bit
- return one of the following:
  - a redirect (HTTP status code 302 Found and a Location: header)
  - sorted mirror list (if requested)
  - metalink (if requested)



- An Implementation

- The Mirrorlist Generator / Redirector



- sort mirrors by closeness, strength and randomize a bit
- return one of the following:
  - a redirect (HTTP status code 302 Found and a Location: header)
  - sorted mirror list (if requested)
  - metalink (if requested)



- An Implementation

- The Mirrorlist Generator / Redirector



#### Example request:

#### GET /dist/openSUSE-10.3.iso HTTP/1.1

Host: download.opensuse.org

#### The Server Replies With A Redirect:

HTTP/1.1 302 Found Date: Sun, 02 Mar 2008 10:14:58 GMT Server: Apache/2.2.8 (Linux/SUSE) Location: http://ftp5.gwdg.de/opensuse/dist/openSUSE10.3.iso



- An Implementation

- The Mirrorlist Generator / Redirector



#### Example metalink reply (shortened):

- <?xml version="1.0" encoding="UTF-8"?> <metalink version="3.0" xmlns="http://www.metalinker.org/" origin="http://download.opensuse.org/dist/openSUSE-10.3.iso">
- <files>

```
<file name="openSUSE-10.3.iso">
```

<resources>

```
<url location="de" preference="100"> http://... </url>
<url location="de" preference="100"> http://... </url>
<url location="us" preference="99"> http://... </url>
[...]
```



- An Implementation

- The Mirrorlist Generator / Redirector



#### Log Example Of A Redirect:

85.84.25.24 - - [07/Feb/2008:15:30:24 +0200] "GET /update/10.3/repodata/patch-kernel-4943.xml HTTP/1.1" 302 356 "-" "Novell ZYPP Installer" uminho.pt 137 741 EU:ES size:51940

> 302 uminho.pt EU:ES

HTTP status code mirror identifier continent:country



- An Implementation

- The Mirrorlist Generator / Redirector



#### Now I'll talk about experiences with the deployment.



Case Study

-download.opensuse.org

## ApacheCon Europe '08

### Outline

- ntroduction About The Problem Approaches
- An Implementation Building Blocks Mirror Database The Mirrorlist Generator / Redirector

#### Case Study

#### download.opensuse.org

What We Optimized Pros, Cons, Ideas



-download.opensuse.org



download.opensuse.org - Download Server For:

- An operating system, and thousands of components that ship with it
- Different releases, architectures, ...
- Ongoing stream of security updates and bugfixes
- Ongoing "Check for updates" by clients (majority of requests)



- Case Study

-download.opensuse.org



- Number of files: > 700.000
- Size: 864 GB
- High turnover rate

Quote of a mirror:

That sounds onerous - a full ubuntu mirror (including ISO's) is 260GB, debian without ISO's is 320GB



-download.opensuse.org



Human users

- Download mostly large files (CD/DVD images)
- 0.5 to 35 req/s

Machine clients

- Variety of "installer tools"
- Smaller files
- 200 to 400 req/s

Altogether, 15,000,000 to 40,000,000 requests per day



-download.opensuse.org



The Hardware Is Mediocre:

- Web server:
  - P4 2x 3.4GHz, 4GB RAM, SLE10
  - SAN with 1.4TB XFS filesystem
  - is stage.opensuse.org (rsync server) at the same time
- Database Server:
  - Xeon 4x 3.4GHz, 4GB RAM, SLE10
  - also serves as "scan host"



-download.opensuse.org



But The Numbers Are Good!

openSUSE 10.3 release, October 2007:

- Peak bandwidth "served": 13 GB/s, i.e. 100 TB in a day.
- Memory usage of httpd: 50-200 MB (sum of RSS minus SHARED of all processes)
- Insignificant load (about 1)





-download.opensuse.org



#### **On The Shoulders Of Giants**





- Case Study

-download.opensuse.org











- Case Study

-download.opensuse.org







-download.opensuse.org



Result: The described approach works well for us.

- Lots of headroom
- Rock-solid



- Case Study

-download.opensuse.org



# The Apache HTTP server and the APR are really an excellent infrastructure to build upon!



Case Study

- Optimizations



### Outline

ntroduction About The Problem Approaches

An Implementation Building Blocks Mirror Database The Mirrorlist Generator / Redirector

#### Case Study

download.opensuse.org What We Optimized

Pros, Cons, Ideas





The main optimization work was:

- Database tuning
- Improvement of the rsync modules for mirror feeding
- Enable mirrors to mirror the most popular 10% of the content
- Cache control headers (needed regardless of mirrors)
- Figure out the critical files not to redirect



Optimizations



- Integration with "real" CDN (catch-all mirror with country='\*\*')
- Send "weak" mirrors only regional requests (critical feature for them)
- Permit a "fragile" mirror in a remote region if it is the only one
- Respect special network topology of countries and their connectivity (e.g. New Zealand).
- Circadian variation of selection probability for certain mirrors



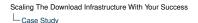


Optimizations



- Integration with "real" CDN (catch-all mirror with country='\*\*')
- Send "weak" mirrors only regional requests (critical feature for them)
- Permit a "fragile" mirror in a remote region if it is the only one
- Respect special network topology of countries and their connectivity (e.g. New Zealand).
- Circadian variation of selection probability for certain mirrors



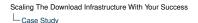


Optimizations



- Integration with "real" CDN (catch-all mirror with country='\*\*')
- Send "weak" mirrors only regional requests (critical feature for them)
- Permit a "fragile" mirror in a remote region if it is the only one
- Respect special network topology of countries and their connectivity (e.g. New Zealand).
- Circadian variation of selection probability for certain mirrors



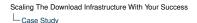


Optimizations

ApacheCon Europe '08

- Integration with "real" CDN (catch-all mirror with country='\*\*')
- Send "weak" mirrors only regional requests (critical feature for them)
- Permit a "fragile" mirror in a remote region if it is the only one
- Respect special network topology of countries and their connectivity (e.g. New Zealand).
- Circadian variation of selection probability for certain mirrors





Optimizations



- Integration with "real" CDN (catch-all mirror with country='\*\*')
- Send "weak" mirrors only regional requests (critical feature for them)
- Permit a "fragile" mirror in a remote region if it is the only one
- Respect special network topology of countries and their connectivity (e.g. New Zealand).
- Circadian variation of selection probability for certain mirrors



Case Study

- Pros, Cons, Ideas

### ApacheCon Europe '08

### Outline

ntroduction About The Problem Approaches

An Implementation Building Blocks Mirror Database The Mirrorlist Generator / Redirector

#### Case Study

download.opensuse.org What We Optimized

Pros, Cons, Ideas



-Case Study

- Pros, Cons, Ideas



Good:

- Open Source
- The implementation is not tied to openSUSE
- You can use it!





File-Level Granularity, Rather Than Directory-Level

- Makes download statistics possible
- Makes small & partial mirrors useful
- Maximum control over *how* content is served. (Mirrors don't care about cache control headers, but you might depend on them)
- If a "broken file" is identified, you can stop redirecting for it, instead of waiting for mirror synchronisation





General Disadvantage Of Mirrors That You Don't Control:

- Mirrors die all the time, and don't hardly ever give you notice about it
- There is a time window of some minutes between the failure, and detecting it and automatically disabling the mirror
- Some failures very hard to detect (just think sporadic firewall quirks)

Client-side failover can help a lot here.



- Pros, Cons, Ideas



- Transparent metalink support
- Client feedback could trigger reactive mirror probing
- Hack the rsync daemon to directly update the database
- Find automated way to mirror files based on popularity
  - ad-hoc rsync modules?
  - massive space-savings on mirrors conceivable
- External api for mirror admins, to disable hosts, change priority or trigger re-scan



- Pros, Cons, Ideas



- Transparent metalink support
- Client feedback could trigger reactive mirror probing
- Hack the rsync daemon to directly update the database
- Find automated way to mirror files based on popularity
  - ad-hoc rsync modules?
  - massive space-savings on mirrors conceivable
- External api for mirror admins, to disable hosts, change priority or trigger re-scan



- Pros, Cons, Ideas



- Transparent metalink support
- Client feedback could trigger reactive mirror probing
- Hack the rsync daemon to directly update the database
- Find automated way to mirror files based on popularity
  - ad-hoc rsync modules?
  - massive space-savings on mirrors conceivable
- External api for mirror admins, to disable hosts, change priority or trigger re-scan



- Pros, Cons, Ideas



- Transparent metalink support
- Client feedback could trigger reactive mirror probing
- Hack the rsync daemon to directly update the database
- Find automated way to mirror files based on popularity
  - ad-hoc rsync modules?
  - massive space-savings on mirrors conceivable
- External api for mirror admins, to disable hosts, change priority or trigger re-scan



- Pros, Cons, Ideas



- Transparent metalink support
- Client feedback could trigger reactive mirror probing
- Hack the rsync daemon to directly update the database
- Find automated way to mirror files based on popularity
  - ad-hoc rsync modules?
  - massive space-savings on mirrors conceivable
- External api for mirror admins, to disable hosts, change priority or trigger re-scan



- Pros, Cons, Ideas



- Transparent metalink support
- Client feedback could trigger reactive mirror probing
- Hack the rsync daemon to directly update the database
- Find automated way to mirror files based on popularity
  - ad-hoc rsync modules?
  - massive space-savings on mirrors conceivable
- External api for mirror admins, to disable hosts, change priority or trigger re-scan



- Pros, Cons, Ideas



- Transparent metalink support
- Client feedback could trigger reactive mirror probing
- Hack the rsync daemon to directly update the database
- Find automated way to mirror files based on popularity
  - ad-hoc rsync modules?
  - massive space-savings on mirrors conceivable
- External api for mirror admins, to disable hosts, change priority or trigger re-scan



- Pros, Cons, Ideas



#### Other Ideas

- Finer geolocation would be good for "Internet countries" like Germany
- Send mirrors their local clients (by network prefix?)
- Stickyness of (large) files to certain mirrors, to make better use of buffer caches?



- Pros, Cons, Ideas



Other Ideas

- Finer geolocation would be good for "Internet countries" like Germany
- Send mirrors their local clients (by network prefix?)
- Stickyness of (large) files to certain mirrors, to make better use of buffer caches?



- Pros, Cons, Ideas



Other Ideas

- Finer geolocation would be good for "Internet countries" like Germany
- Send mirrors their local clients (by network prefix?)
- Stickyness of (large) files to certain mirrors, to make better use of buffer caches?



- Case Study

- Pros, Cons, Ideas



Your Ideas?

This space intentially left blank





Summary

- Mirrors can be used to build a poor man's CDN (Content Delivery Network).
- Mirrors out of your control, and partial mirrors can still be useful.
- The more complex and voluminous the content gets, the more mirror monitory is needed.
- Outlook
  - Transparent integration of metalinks: a great plan.





#### We just love mirrors...



...because they make us visible :-)





## Thanks!





## **Questions?**

poeml at mirrorbrain.org



- Appendix

- For Further Reading



### For Further Reading

- http://mirrorbrain.org/
- http://www.poeml.de/users/poeml/talks/apachecon08mirrors.pdf (this talk)
- http://www.opensuse.org/Build\_Service/Redirector
- https://forgesvn1.novell.com/svn/opensuse/trunk/ tools/download-redirector-v2/mod\_zrkadlo/mod\_zrkadlo.c



- Appendix

- For Further Reading

### ApacheCon Europe '08

#### Other Existing Approaches

- Bouncer: (Mozilla project) essentially similar approach, but different implementation (PHP script); (I think) more specialized to Mozilla software download structure
- Fedora MirrorManager / Yum: principally a very similar approach, but done differently ;) They evolved from static lists to dynamic mirror lists. Works with less granularity (directory-wise).
- geomcfly: on-the-fly generator of metalinks based on clients' geographical location. No mirror management (I think)
- mirmon: more a monitoring framework, but can be used with a redirector. Implementation is quite different. Doesn't keep inventory of mirror, but checks a timestamp.

For Further Reading



Other Existing Approaches (continued)

- Web caches (squid): could work fine, but requires people to set up squids ;)
- Coral CDN, uses standard DNS but is not transparent
- mod\_offload: requires script on mirror, which makes it act as "active" cache. Files are mirrored on demand. Practical if you control all mirrors
- BitTorrent (and other P2P): requires special client

