Content Distribution On Large Scale

10 Things You Might Want To Know About openSUSE Infrastructure

Dr. Peter Poeml

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

Knowledge Sharing – 3rd July, 2008

Introduction

- About
- The Problem
- Approaches

Implementation

- Components
- Mirror Database
- The Mirrorlist Generator / Redirector

Deployment

- Setup
- What We Optimized
- Pros, Cons, Ideas

Outline

Myself:

- With 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 platform (SLES7/8)

This Talk:

- Challenges at openSUSE.org
- How we distribute the traffic
- Things that might be relevant to you
- Demo

Outline

"Everything counts in large amounts"

- Different releases, subprojects, architectures, ...
- Large files (CD/DVD images)
- Ongoing stream of security updates and bugfixes
- Ongoing "Check for updates" by clients (majority of requests)

More downloads than one could ever handle

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

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

About 50% of those are redirected to mirrors.

Outline

Content Delivery Networks (CDN)

- Wide area load distribution by adding intelligence to standard DNS
- "Industry standard" solution – used by Apple, Novell, ...
- Too expensive for open source projects
- openSUSE gets some leftover capacities from Novell

Mirrors Come To Help

- Task: build a "Poor man's CDN"
- Even though we don't control them ourselves

Mirrors are incomplete

- Huge amounts of content: only parts will be mirrored.
- Rapidly changing content: mirrors can never be 100% up to date.

=> we need to deal with **partial** mirrors.

Four Ways To Distribute Traffic To Mirrors

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

Method 1: **Static** Mirror Lists

- Hard to maintain
- Too static
- Hardly ever correct
- Low granularity
- Work for small file trees

Method 2: **Dynamic** Mirror Lists

- Mirror monitoring increases correctness
- Automation -> finer granularity
- Often combined with geolocation of clients
- User gets a suggestion, or needs to chose interactively
- Works for single files (like DVD image, or Samba tarball)
- Can annoy users, or make them all pick the same (good) mirror
- Doesn't work so well for automated downloads

Method 3: Dynamic Mirror Lists, **Transparent Redirects**

- Mirror choice made by server
- Client doesn't see the other mirrors
- User doesn't need to figure out
- But more difficult for user to override choice
- Relies on intensive mirror monitoring
- Good for machine clients

Method 4: **Metalinks**

- *A Metalink is a mirror list in standardized, machine-readable format* (see metalinker.org)
- HTTP, FTP, P2P under one umbrella
- Client can make its own choice, failover possible
- Good for humans and machines

More about Metalinks

- "self-healing downloads" experience
- XML file containing HTTP, FTP, BitTorrent or other P2P URLs
- Segment hashes for transfer integrity checking
- Can include PGP signatures
- Clients: aria2c (commandline), DownThemAll (Firefox extension), KGet, ...

My Christmas Wish For The Future Of Downloading...

- **Transparently negotiated** metalinks
- => no extra link needed
  - Metalink clients will get metalink
  - Other clients will get redirect
- Supported **today** by download.opensuse.org, aria2, DownThemAll, Retriever, Metalink Checker
- Hopefully becoming the standard
- **Goal: native support in web browsers**

My Christmas Wish #2

- Metalink support in libzypp
- GSoC student working on it

Go, Gerard, go!

Outline

openSUSE
Novell

People call it "redirector" – but it rather is sort of a "mirror brain".

- Mirror database
- Monitoring tools
- Mirrorlist generator and redirector
- Communication & documentation

Outline

The Mirror Database

- Keeps inventory of mirrors, on file-level
  - Acquired and updated by crawling the mirror via rsync, FTP or HTTP
- Keeps online status of mirrors
  - Probing at short intervals
- Keeps metadata on mirrors
- Functional tests – does a mirror handle files > 2GB and byte ranges?

Outline

The Mirrorlist Generator / Redirector

- Apache module ("mod_zrkadlo")
- Hooks into request processing phase

The Apache module proceeds like this:

- 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 Apache module proceeds like this:

- 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 Apache module proceeds like this:

- 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 Apache module proceeds like this:

- 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 Apache module proceeds like this:

- 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 Apache module proceeds like this:

- 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

- 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)

- 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)

- 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)

- 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)

- 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)

Example request:

GET /**dist/openSUSE-10.3.iso** HTTP/1.1
Host: **download.opensuse.org**

Server Reply:

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/openSUSE-10.3.iso**

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>
[...]

Outline

openSUSE
Novell

Server Hardware:

- download.opensuse.org:
  - P4 2x 3.4GHz, 4GB RAM, SLE10
  - SAN with 1.4TB XFS filesystem
  - also serves stage.opensuse.org (rsync mirror feed) & drpmsync.opensuse.org & bittorrent tracker & repository pusher
- mirrordb.opensuse.org:
  - Xeon 4x 3.4GHz, 4GB RAM, SLE10
  - mirror database and scan host

- widehat.opensuse.org:
  - Xeon 8x 2.3GHz, 16GB RAM, SLE10
  - SAN with 1.4TB reiserfs filesystem
    - reserve mirror (controlled by us)
    - rsync.opensuse.org (public rsync mirror feed)
    - bittorrent seeder
  - sponsored by our ISP (IPExchange)

History/Timeline

- 11/2006: hotfixing overloaded server during 10.2 release
- 5/2007: redirector replaced
- 8/2007: got widehat.o.o
- 9/2007: openSUSE 10.3 and updates on d.o.o
- 4/2008: metalink support
- 5/2008: automatic checking of mirrors large file support
- 6/2008: openSUSE 11.0 (second release with updates on d.o.o)

ftp.suse.com

- Is being phased out
- /pub/projects tree has a little activity
- 10.2 update tree is the last one

Impressive Numbers

- openSUSE 10.3 release, October 2007:
  - Peak bandwidth "served": 13 GB/s, i.e. 100 TB in a day.
- openSUSE 11.0 release, June 2008:
  - Peak bandwidth "served": 22 GB/s, i.e. 170 TB in a day.

Served By Nearly 100 Active Mirrors

openSUSE
Novell


Internode


IP
eXchange
Top Business
Colocation
Services
Germany


NOVO LOGÓTIPO DO IPB
ipb
INSTITUTO POLITÉCNICO DE BRAGANÇA


IS
internet solutions
A DIVISION OF DIMENSION DATA


Israel Internet Association (ISOC-IL)
איגוד האינטרנט הישראלי ע"ר


사이트맵
KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY


UNIVERSITAS INDONESIA
The university with world class perspectives


upc


KDDI
株式会社KDDI研究所
KDDI R&D LABS


MIRRORS
.KERNEL.ORG


linux.cz


linux
Migratio


irror.in.th
THAI NATIONAL MIRROR


nfsi telecom


NLUUG


NOAA
NATIONAL OCEANIC AND
ATMOSPHERIC ADMINISTRATION


NYI
WHEN YOUR MISSION IS CRITICAL


openSUSE
Novell


OSL
OPEN SOURCE LAB
osuosl.org


PACNET


独立行政法人
理化学研究所


ROMANIAN EDUCATION NETWORK
RoEduNet


RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
RWTH AACHEN


SILEMAN
TWÓJ INTERNET I TELEFON


skynet
Belgacom


SOLNET


SWEDISH UNIVERSITY
COMPUTER NETWORK
Sök · Kontakta SUNET · Nätkarta · Anslutna till SUNET
In english · Nätmöten · Filarkivet · Sunetten · Denna dag
SUNET


SWITCH
Serving Swiss Universities


TDS


TEMPLE
UNIVERSITY


TECHNISCHE UNIVERSITÄT
CHEMNITZ


TECHNISCHE UNIVERSITÄT ILMENAU


TECHNISCHE
UNIVERSITÄT
MÜNCHEN


TU
WIEN


TWAREN


UNIVERSITÄT
BAYREUTH


Friedrich-Alexander-Universität
Erlangen-Nürnberg


openSUSE

Novell


RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG


Westfälische
Wilhelms-Universität
Münster


ulm university
universität
uulm


UNIVERSITY
OF OSLO


UNIVERSITA' degli STUDI di ROMA
TOR VERGATA


UNIXHEADS.ORG


UOL
O MELHOR CONTEÚDO


THE
UNIVERSITY
OF UTAH


UNIVERSIDADE TÉCNICA
DE LISBOA
UTL


ndex
Найдётся всё


元智大學
Yuan Ze University

Outline

Scalability

- Apache needs 50-200 MB
- Load average about 1
- Database fits into memory

Stability

- Solid
- Downtimes limited to human error and hardware issues

Main optimizations were:

- Smaller rsync modules
- New rsync modules for mirroring the most popular 10%
- Refinement of mirror selection
- Cache control headers
- Figure out the critical files **not** to redirect

Files not redirected:

- update and factory tree: files without digit in name
- repositories tree: .xml .xml.gz .xml.asc .repo .ymp
- broken clients (user agents rpm/4.4.2 or APT-HTTP)
- files not present on any mirror

Four Things The Content Creators Should Know

- New content needs to be considered for **mirroring** / mirroring exclusion
- New content needs to be considered for **redirection** / redirection exclusion
- Content which changes **infrequently** needs cache control headers so it **is** cached
- Content which changes **frequently** needs cache control headers so it is **not** cached (or validated)

=> Let openSUSE infra people know about new content

Things that should *not* be mirrored

- Debuginfos
- Sources
- Unpopular architectures
- Older install repos

Our tree needs a split-up, into important and unimportant stuff

Outline

- Open Source
- Generic implementation

File-Level Granularity vs. Directory-Level

- Allows for download statistics
- Makes small & partial mirrors useful
- Maximum control over *how* content is served. (Mirrors don't care about cache control headers)
- If a "broken file" is identified, we can stop redirecting for it, instead of waiting for mirror synchronisation
- If we spread broken URLs, we can work around on the server side

General Disadvantage Of Mirrors:

- They die all the time, and mostly don't tell you
- Time window between failure and detection
- Failures can be *very* hard to detect (think of sporadic firewall quirks)

Client-side failover needed

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.

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): Only suitable for large files. Requires special client

Todo / Ideas

- Promote metalinks
- 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

Todo / Ideas

- Promote metalinks
- 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

Todo / Ideas

- Promote metalinks
- 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

Todo / Ideas

- Promote metalinks
- 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

Todo / Ideas

- Promote metalinks
- 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

Todo / Ideas

- Promote metalinks
- 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

Todo / Ideas

- Promote metalinks
- 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

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?

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?

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?

Your Ideas?

(This space intentially left blank)

We just **love** mirrors...

...because they make us visible :-)

Thanks!

Questions?

poeml at suse.de

For Further Reading

- http://mirrorbrain.org/
- http://www.opensuse.org/Build_Service/Redirector
- https://forgesvn1.novell.com/svn/opensuse/trunk/tools/download-redirector-v2/mod_zrkadlo/mod_zrkadlo.c