Through the Looking-Glass: the technology behind the UK Mirror Service

http://www.mirrorsservice.org/

Adam Sampson
atsl@mirrorsservice.org

University of Kent
Introduction
What is UKMS?

- Fast local copies (mirrors) of popular Internet resources for the UK academic community

Some approximate numbers for September 2004:

- 201 mirrored sites
- 4 million files
- 6 terabytes of disk space
- 0.4 terabytes of data shipped per day
- Average bandwidth usage 42Mbit/sec (peaking at 100Mbit/sec)
- 120,000 distinct user IP addresses per month
- 28% outgoing traffic is to UK academic users on JANET
- 100% availability from 1999 to 2004
What we carry

- Open Source software (the vast majority)
  e.g. Debian GNU/Linux, OpenOffice.org

- Academic sites
  e.g. Project Gutenberg, Duke Papyrus Archive

- Some commercial software
  e.g. MATLAB, Netscape

- Legacy content (HENSA/Micros, etc.)

- Official mirror site for many mirrors
History

1987 netlib mirror at UKC; access by email
(18 years later, we still mirror netlib)

1992 UKC gets first Internet link;
ISSC-funded HENSA/Unix at UKC, HENSA/Micros at Lancaster
(hensa.ac.uk)

1999 HENSA merges to become the JISC-funded UK Mirror Service with staff at UKC and Lancaster
(mirror.ac.uk)

2003 Added third UKMS site at Reading C-POP

2004 JISC contract expires;
UK Mirror Service now operates from UKC Computer Science
(mirrorbservice.org)
Architecture
From the outside

connections to source sites via FTP, HTTP or rsync

user connections via FTP, HTTP or rsync

The UK Mirror Service
Removing the lid via FTP, HTTP or rsync user connections
connections to source sites via FTP, HTTP or rsync
connections to source sites via FTP, HTTP or rsync
user connections via FTP, HTTP or rsync
internal network
SCSI disk disk disk disk disk disk disk disk disk
Explaining the roles

- Frontend and backend hosts
- Users make FTP, HTTP or rsync connections to a randomly-selected frontend host (DNS round-robin entries)
- Frontends act as smart caching proxies to reduce load on backends and disks
- Frontends fetch the data from the backends
- Backends have disks attached, each with several mirrors on
- Backends periodically fetch data from source sites to disks
Hardware at UKC

- **4× Sun V120s** as frontends
- A V120 as “slow” backend with disk arrays:
  - a 400-gigabyte Sun 3300
  - a 4-terabyte Transtec (cheap!)
- A Sun E450 as “fast” backend with Sun A1000 disk arrays:
  - **2× 160-gigabyte**
  - **2× 280-gigabyte**
  - **2× 400-gigabyte**
UKMS backend traffic levels

Bytes/second

compton en0 out
compton hme0 out
palomar hme0 out

A fortnight in the life
Serving content
Serving in general

- Making a directory on a backend disk available to users
- Must support a standard directory layout across all protocols
- Must transparently select the right backend for different mirrors
- Must minimise the backend load where possible
- Must respect the source site’s presentation instructions
Zooming in...
An aside: CFTP

- A C++ library and a set of tools
- Used by nearly every part of the UKMS software
- The result of a late-90s UKC research project
- Provides a virtual Unix-like filesystem tree with mountable filesystems
  - **posix-fs** connects to a real local directory
  - **ftp-fs** connects to another FTP server
  - **remote-fs** connects to a directory on another host (using its own protocol)
Serving content with FTP

- **ftp-server** is a custom FTP server based on CFTP
- Limit on the number of concurrent connections to each frontend (client should try next FE if full)
- Can generate tar archives on the fly
- The FTP server exports the CFTP virtual filesystem
- Each mirror is mounted using **remote-fs** from the appropriate backend in the right place under /sites/
- **remote-fsd** server runs on backends

Through the Looking-Glass – p. 16
Serving content with rsync

- ukms-rsync is a patched version of the stock rsync server
- Uses libqfs, a C interface to the CFTP virtual filesystem (looks like Unix system calls)
- Uses nearly the same config for CFTP as the FTP server
CFTP frontend caching

- Initial approach used existing CFTP cache – not good enough!
- Final-year project implemented a better cache filesystem for CFTP
- Cooperative data and metadata caching between all FTP/rsync processes on a frontend
- Tested on real logs – 30 gigabyte data cache reduced data pulled from backend by 50%
Serving content with HTTP

- We use the Apache web server (twice)
- Frontend Apache acts as a caching proxy, forwarding requests on to the appropriate backends
- Apache’s cache behaves poorly for large files, though, so we redirect requests for CD images to the FTP server
- Most virtual hosts handled directly by frontends
- “Special” virtual hosts handled on backends using extra ports
- Doesn’t use CFTP yet, but we’re working on it
The browser

- Generates our web interface under /sites/
- A (C!) CGI program that runs on backends
- Supports browsing and extracting from archive files
- Displays likely README files and mirror descriptions
The search engine

- Uses a separate machine and a huge PostgreSQL database
- Indexing scripts examine newly-mirrored data
- Web frontend does queries against the database
- Much more complex than it sounds – good searching is difficult!
Mirroring
Mirroring in general

- Making a local directory look like one on the source site
- May need to exclude some content, or add extra content
- Need to cope with source site being down
- Need to update search engine
- Need to copy mirrored content to other UKMS sites
FTP mirroring with syncfs

- `syncfs` is a general mirroring tool based on CFTP
- Uses `ls-lR` files if available on source site
- Can use multiple connections
- Can resume partial downloads
- Updates mirroring status files (hidden from users)
- Lots of special behaviour to deal with broken FTP servers
Peer mirroring

- Syncfs mirroring technique using two UKMS sites (UKC and Lancs)
- Both UKMS sites connect to source site
- One mirrors in alphabetical order, the other in reverse alphabetical order
- Each checks the other UKMS site to see whether it’s got the file they want before going to the source site
- On average, each pulls half the content from the source site
- Works when one UKMS site is down too!
Web mirroring

- Actually a general term for anything that’s not FTP
- Wrapper around off-the-shelf tools
  - rsync for rsync mirroring
  - pavuk for HTTP mirroring
  - tucopy for Tucows mirrors
- Has workarounds for broken tools
  - Detect empty files and remirror
  - Fix up bad links in HTML
  - Detect “stuck” mirroring processes and restart
- Push mirroring via a modified writable rsync server
Tying it all together
Metaconf

- Configuring all our software by hand would be impractical.
- Metaconf takes descriptions of mirrors in a standard format (MDF).
- Provides a uniform interface that scripts can use to get at the descriptions.
- Generates per-site configuration files for software that can’t use the Perl interface.
- Copes with remapping disks when faults occur.
MDF

- An application of RDF (yes, the Semantic Web is useful!)
- An MDF file describes a single mirror:
  - The name, description and classification of the mirror
  - The logical disk upon which it’s stored
  - The host that performs mirroring
  - How and when to mirror it (including any special options)
  - How it can be accessed (protocols, virtual hosts)
What we’ve learnt
...about mirroring

- Source sites are usually broken
  - The most important source sites are always overloaded
  - With FTP listings, anything that can go wrong will
- ...but FTP is still the best protocol for mirroring
  - HTTP mirroring is basically guesswork
  - rsync works badly for very large mirrors
- Off-the-shelf mirroring software is unreliable
- Source site maintainers don’t have the time to listen to mirror maintainers
...about serving

- Client software is usually broken
  - “Download accelerators”
  - Incorrect HTTP Redirect handling
  - Large file handling (Fedora DVDs)
- Malicious users exist
- Overzealous indexing bots also exist
- Frontend caching is a really good idea
...about protocol design

- FTP has some serious problems
  - Poor takeup of standardised directory listings
  - The whole active/passive mess – NAT/firewall unfriendly

- HTTP has some serious problems
  - No directory listings (WebDAV isn’t there yet)
  - It’s not good for mirroring

- rsync has some serious problems
  - Huge latency when transferring initial tree
  - Random failures during transfers
  - Backwards compatibility makes code messy

- All protocols suck!
...about software design

- Don’t! Big design up front doesn’t work for us
- Our most reliable systems are those that have evolved slowly
- Clear code is easier to modify than a well-documented mess
- Use the right language for the job
- Simple file formats are best
- Being self-healing is useful
- Version control software is invaluable
Future plans
Making our software Open Source

- UKC owns copyright on most of the code
- Need to tidy up build systems and package for release
- Patches to existing software (rsync) are easier
- Package as “Mirror Service In A Box” for others to use
Reducing cost

- Sun kit is reliable but (very!) expensive
- PCs and SATA disks are cheap (free in some cases) and fast
- Our plan is:
  - 6 PCs, each with 4 300-gigabyte SATA disks
  - Mirrored pairs of machines for redundancy
  - Each machine is both a frontend and a backend
- We aren’t the only people who’ve noticed this – Google and archive.org take the same approach on a much larger scale!
The End
Any questions?

Find us at:
http://www.mirrorservice.org/