Getting the Bits Out: Fedora MirrorManager

Matt Domsch
Dell
Matt_Domsch@Dell.com

Abstract

Fedora is fortunate to have several hundred volunteer mirror organizations globally. MirrorManager tracks all of these mirror servers and automatically directs users to a local, fast, current mirror. It has several unique features, including registration of private mirrors and designation of preferred mirrors by IP address—a great benefit to corporations, ISPs, and their users; and automatic direction of Internet2 clients to Internet2 mirrors. This paper presents the web application architecture that feeds updates to over 200,000 users each day. It provides instructions for setting up local private Fedora or EPEL mirrors for schools, companies, and organizations, and explains how you can volunteer to help distribute Fedora worldwide.

1 Introduction

The Fedora Project (hereafter 'Fedora') is a leading-edge Linux distribution that provides the newest and best Free and Open Source Software to millions of users worldwide. MirrorManager (MM) [9] is the tool developed to get that software out to those users accurately, quickly, and inexpensively.

To assist with this distribution, Fedora is fortunate to have several hundred volunteer mirror organizations globally. These organizations provide manpower (responsive system administrators), servers, storage, and copious bandwidth. Each mirror server carries a subset of the content available on the Fedora master servers. It is often fastest and least expensive for these mirrors to serve users whom are "local" network-wise. MM tracks all of these mirror servers and automatically directs users to a local, fast, current mirror.

We present MM from three aspects. Section 3 shows how end users download software transparently using MM. Section 4 shows how mirror system administrators interact with MM. Section 5 goes behind the scenes into the design of the MM software itself.

2 Background

There are three factors to consider when scoping the size of the distribution channel you need: number of users, size of the software, and available network bandwidth.

By conservative estimates [7], Fedora has nearly 2 million users worldwide. Neglecting the number of users who buy or receive free CDs, at a minimum each user downloads one CD worth of material (about 700MB). This equates to at least 1.4 Exabytes of data to serve for each release. With a single 45 Mbit/second T3 network connection, it would take over 8 years to serve all this content. Security and bugfix updates could easily double this number. At this rate, Fedora releases occur every 6 months, we'd fall behind very quickly (not to mention lose our entire user base!).

As for total disk space, Fedora keeps at least the current release (at time of press, Fedora 9), the previous release (Fedora 8), and the next previous release (Fedora 7) online and available for download. Each Fedora version release, including packages, CD and DVD images, and daily security and bugfix updates, can consume up to 200GB of disk space. In addition, alpha and beta test releases, and the "rawhide" tree (the development tree for what will be the next major release), are posted regularly. These consume a bit less space than a full release. Overall, about 1TB of space is constantly needed on the master servers and for each full mirror.

While our mirror organizations are altruistic, they're also not overly wasteful. Each mirror may choose to carry only a subset of the available content, such as omitting lesser-used architectures and debug data. This means it's not sufficient to know which mirrors exist, but we must also know which content each carries. This precludes using a simple DNS round-robin redirector.

Further complicating matters, due to historical ways in which the content was offered via rsync modules, each

mirror server may publish their tree of the Fedora content at paths of their choosing—often not matching that of the master servers. This makes it even more important that tools can discover the content a mirror carries, and at which URLs that content is served—a naïve redirect would fail miserably.

Organizations have several reasons why they choose to become a Fedora mirror. Generally, they have many Fedora users locally, and for those users, it's faster (and for the organization, less expensive) if they can pull that content from a local mirror rather than across the Internet multiple times. For large Internet Service Providers or organizations, the savings can be quite dramatic.

Organizations that are part of Internet2, or one of the high speed research and educational networks that peer with it, often have significantly lower costs and higher bandwidth when passing traffic over Internet2 than over their commercial links. Fedora itself does not have any public download servers that are accessible via Internet2, but more than half of the Fedora public mirror servers are accessible via Internet2. By directing users to local or Internet2-connected mirrors, they can get the benefit of high speed downloads at a reduced cost.

2.1 Sidebar: Preventing Meltdown

One of the driving forces behind MM is to get the bits to end users as fast as possible. A related goal is to keep Fedora's primary sponsor, Red Hat, online during release week.

In October 2006, Fedora had around 100 active mirrors. During the days leading up to a release, individual mirror admins would report by email that they were synced. However, the list of mirrors was managed manually, included in release announcements manually, and generally quite error-prone (dozens of text files had to be updated correctly, once for each mirror reporting ready).

When Fedora Core 6 was released that month, demand was immense—over 300,000 installs in the first three weeks—larger than ever seen for a Red Hat Linux or Fedora release. A few dozen mirrors were synced in time for the release, but nowhere near sufficient capacity to handle the demand. It didn't help that the web page most users were being directed to in order to begin their download pointed them to use Red Hat's own servers, not mirror servers.

On top of this, an apparent Distributed Denial of Service attack was mounted against Red Hat's own servers on release day. Talk about kicking you when you're down.

The result: for the week following the Fedora Core 6 release, significant portions of Red Hat's network became unusable for anything other than responding to the DDoS attack and serving Fedora content. You can imagine the joy this brought to Red Hat executives. The mirrors were annoyed that they would finally get synced, only to not be listed on the mirror list web pages (the Fedora sysadmins were busy trying to handle the traffic and keep everything running, and were slow getting those manual lists updated). Chaos and confusion.

Thus MM was born, to address the shortcomings of manually updating dozens of text files, and to ensure all known mirrors were accounted for and being put to good use.

Six months later, MM made its debut with the Fedora 7 release. Fortunately, there was no DDoS attack this time, and while there were some growing pains getting all the mirrors listed in the database, it went quite smoothly.

In November 2007, Fedora 8 was released. With every confidence in MM and the mirrors themselves, the Red Hat servers were removed from public rotation—Red Hat served bits to the mirrors, but served very few end users directly. From Red Hat's perspective, the release went so smoothly they didn't even know it happened. Users were able to get their downloads quickly. Life was good.

3 Getting the Bits: End Users

End users have several options for downloading Fedora CDs, DVDs, and packages. Outside the scope of MM, Fedora serves the content via BitTorrent. However, tools such as `yum` do not use BitTorrent, and network restrictions by a user's organization may prevent BitTorrent or other peer-to-peer download methods.

Critical to the goal of delivering mirrored content to users quickly is the *redirector* which automatically redirects user download requests to an up-to-date, close mirror, using several criteria:

- The user's IP address is compared against a list of network blocks as provided by each mirror server.

If a user is on a network served by a listed mirror server, the user is directed to that network-local mirror. This should be the fastest and least expensive way to serve this user.

- If the user is on a network served by Internet2 or its peers, they are redirected to another Internet2-connected mirror in their same country, if available. MaxMind's open source and zero-cost GeoIP database provides country information.
- Users are directed to mirrors in their same country, if any.
- Users are directed to mirrors on their same continent, if any.
- Users are directed to one of the mirrors globally.

This search algorithm, while not always perfect, provides a pretty good approximation of the Internet topology, and in practice has shown to provide acceptable performance for users. In the event a user wants to manually choose a mirror, he or she can look at the list of available up-to-date mirrors [6].

To override this search algorithm in some way (e.g. because GeoIP guesses the country incorrectly, or because the actual network you're on is near a border with another country where there is a faster mirror), users may append flags to the URLs used ([3] or [5]). Table 1 describes the available flags.

4 Hosting the Bits: Mirrors

MM offers several features aimed specifically to assist mirror server administrators most efficiently serve their local users, as well as global users, such as:

- The ability to have "private" mirrors—those which serve only local users and which are not open to the general public.
- The ability to specify the network blocks of their organization. Local users from that organization will be automatically directed to their local mirror.
- The ability to specify the specific countries a mirror should serve.
- The ability to preferentially serve users on Internet2 and related research and educational networks.

These features help help keep down bandwidth costs for serving Fedora users.

4.1 Signing Up

These are the steps involved with registering as a Fedora mirror, either to serve the public, or to serve your own organization.

1. Create yourself a Fedora Account System account [2]. You should have one account per person in your organization who will maintain your mirror. You will be able to list these people as administrators for your mirror site.
2. Log into the MM web administration interface [10].
3. Create a new Site. Sites are the administrative container, and where your organization can get sponsorship credit for running a public mirror. Public mirrors are listed on a `fedoraproject.org` web page with a link to each sponsoring organization.
4. Create a new Host. Hosts are the individual machines, managed under the same Site, which serve content. Sites may have unlimited numbers of Hosts.
5. Add Categories of content for each Host. Most mirrors carry the "Fedora Linux" category (current releases and updates), while some also carry the "Fedora EPEL" (Extra Packages for Enterprise Linux) [1], "Fedora Web" (web site), and "Fedora Secondary Arches" (seconardary architectures such as ia64 and sparc) categories.
6. Add your URLs for each Category. Most mirrors serve content via HTTP and FTP; some also serve via rsync.

In addition, you can set various bits about your Site and Host, including its country, whether it's connected via Internet2 or its peers, whether it's private or public, your local network blocks, etc.

Table 1: mirrorlist flags

Flag	Description
`country=us,ca,jp`	Return the list of mirrors for the specified countries.
`country=global`	Return the global list of mirrors instead of a country-specific list.
`ip=18.0.0.1`	Specify an IP address rather than the one the server believes you are connecting from.

Private Sites or Hosts are those which expect to only serve content to their local organization. As such, they will not appear on the public-list web pages. Hosts default to being "public" unless marked "private" on either the Site (which affects all Hosts), or individually on the Hosts's configuration page. Private Hosts are ideal for universities who have one mirror for internal users, and another they share with the world. Private hosts are returned to download requests based on matching client IP to a Host's netblock.

Netblocks are a feature unique to MM. You may specify all of the IPv4 and IPv6 network blocks, in CIDR format, that your mirror should preferentially serve. Users whose IP addresses fall within one of your netblocks will be directed to your mirror first. There is one security concern, as this could allow a malicious mirror to direct specific users to them. However, as all content served by the mirror system as a whole is GPG-signed by the Fedora signing keys, to be successful the attacker would have to convince the target user to accept their GPG keys as well, which, one hopes, would be unlikely. Mirrors may not set overly large netblocks without MM administrator assistance, further limiting the scope of such possible attack.

Internet2 detection is done by regularly downloading and examining BGP RIB files from the Internet2 log archive server. This data includes all the CIDR blocks visible on Internet2 and its peer research and educational networks worldwide. Clients determined to be on Internet2 will be preferentially directed to a mirror on Internet2 in their same country, if possible. By setting the Internet2 checkbox for the Host, your Host will be included in that preferential list. In addition, private Hosts on Internet2 may be happy to serve clients on Internet2, even if they don't fall within the Host's list of netblocks. MM provides this option as well.

Each Host should list the IP addresses from which they download content from the master servers. These addresses are entered into the rsync Access Control List on the master servers, as well as on the Tier 1 mirrors. This is used to limit the users who may download content from the master mirror servers, so as to not overload them.

4.2 Syncing

Fedora employs a multi-tier system [4] to speed deployments, similar to other Linux distributions. Tier 1 mirrors pull from the Fedora master servers directly, Tier 2 mirrors pull from the Tier 1 servers. Private mirrors pull from one of the Tier 1 or 2 mirrors.

Unique to MM, the tool `report_mirror` is run on each mirror server immediately after each rsync run completes. This tool informs the MM database about the full directory listing of content carried by that mirror. The MM database for each Site contains a password field, used by `report_mirror` to authenticate this upload, so as to not expose an individual user's Fedora Account System username and password.

5 Architecture

The MM software follows a traditional 3-tier architecture of database back-end, application server, and front-end web services. It is written in python, and leverages the TurboGears rapid application development environment. However, some specific design decisions were made to address the memory consumption and multi-threaded locking challenges that python imposes. We split the most often hit web services out from the application server, exactly to address the memory demands.

5.1 Application Server and Database

MM uses TurboGears [13], with the SQLObject [12] object-relational mapper layer for most data, and the SQLAlchemy [11] mapper for integration with the Fedora Account System. The application server provides several entry points:

- The administrative web interface [10], where mirror administrators register their mirrors and can see the perceived status.
- A limited XMLRPC interface used by the `report_mirror` script, run on the mirror servers, to "check in" with the database.
- A web crawler, which detects which mirrors are up-to-date. In conjunction with the `report_mirror` script, this follows the "trust, but verify" philosophy. Mirrors which are unreachable, even temporarily, are removed from the redirector lists.

The database itself can be anything that SQLObject can speak to, including PostgreSQL and MySQL. SQLObject takes care of creating the proper tables and mapping rows into objects. For speed and memory efficiency, some queries are implemented in SQL directly.

5.2 Crawler

The second half of the "trust, but verify" philosophy is the web crawler. This application first updates its record of content found on the master servers. For each public Host, it then scans, using lightweight HTTP HEAD or FTP DIR requests (depending on protocols served by that Host), each file that Host is expected to contain. For large directories full of RPM files, only the most recent 10 files are scanned to cut down on extra unnecessary lookups. Directories where all the files match the master servers are marked up-to-date in the database; unreachable servers or those whose content does not match are marked as not up-to-date, effectively preventing clients from being directed to those Hosts' directories. The crawler can run against several Hosts at once, limited only be available memory on the crawling system.

The crawler extends python's `httplib` to use HTTP keep-alives. This lets it scan about 100 files per server per TCP connection using HTTP HEAD calls which do not download the actual file data, and thus are very fast.

5.3 Web Services

5.3.1 Mirrorlist Redirector

To the end user, the most critical service MM provides is the mirrorlist redirector [5], which directs users to a mirror for the content they request. This service receives all the requests generated by yum looking for package updates, and individuals downloading CD and DVD images from the front page of `fedoraproject.org`. These services operate on a cache of the database, containing pre-computed answers to most queries, for maximum speed.

As this application gets hundreds of hits each second, a pure `mod_python` solution was infeasible—it simply wasn't fast enough, and the memory consumption (upwards of 30MB per httpd process waiting to service a client) overwhelmed the servers. So, we split the application into two parts: a `mod_python mirrorlist_client` app, which marshalls the request and performs basic error checking and HTTP redirection, and a `mirrorlist_server` app, which holds the cache and computes the results for each client request. `mirrorlist_server fork()`s itself on each client connection, keeping the cache read-only (so copy-on-write is never invoked), which eliminates the memory consumption problems and python interpreter startup times. The two communicate over a standard Unix socket. Client requests are answered in about 0.3 seconds on average.

This pair of applications is then replicated on several web servers, distributed globally. This reduces the likelihood of a single server or even data center failure bringing down the service as a whole. In the event of application server or database layer failure, the web services can operate on the cached data indefinitely, until the back ends can be made available again.

5.3.2 Publiclist pages

Aside from the redirector, the second user-visible aspect of MM are the "publiclist pages," web pages that list each up-to-date available public mirror and its properties, including country, sponsoring organization, bandwidth, and URLs to content. These pages are rendered once per hour into static HTML pages and served via HTTP reverse proxy servers, again to make use of caching. This keeps the traffic load manageable, even on very active major release days.

6 Future Work

There are several features MM does not currently provide which would be useful additions.

- MM lists each mirror's available bandwidth, but does not use this information when choosing which mirrors to return in what order. This causes both relatively fast and slow mirrors in the same country to be returned with equal probability. MM should take into account a given Host's available bandwidth, and return a list of mirrors probabilistically favoring the faster mirrors.

- `report_mirror` does not work from behind a HTTP proxy server. Private mirrors need to run this tool, but are often stuck behind such a proxy. This is actually a shortcoming of python's `urllib`.

- Metalink [8] downloads, which would let users pull data from several mirrors in parallel. This is somewhat controversial, as it increases the load on the mirrors (they wind up serving more random read requests, which are much slower than streaming reads). But it might let metalink-aware download tools do a better job of choosing a "close" mirror than MM does.

7 Conclusion

MM has been very effective in getting Fedora content to users quickly and easily. Furthermore, it has decreased the bandwidth burden of Fedora's primary sponsor, Red Hat, by making good use of the contributions from hundreds of volunteer mirror organizations worldwide. Its architecture allows it to serve millions of users, and to scale as demand grows. It's simple and fast for users, and saves money for mirror organizations—a win all around.

8 Acknowledgments

MM is primarily developed for the Fedora Project on behalf of the author and his employer, Dell, Inc. It is licensed under the MIT/X11 license.

MM includes GeoLite data created by MaxMind, available from `http://www.maxmind.com/`.

The Fedora Project is grateful to the hundreds of mirror server administrators and their organizations who help distribute Free and Open Source software globally.

9 About the Author

Matt Domsch is a Technology Strategist in Dell's Office of the CTO. He has served on the Fedora Project Board and as the Fedora Mirror Wrangler since 2006.

References

[1] Extra Packages for Enterprise Linux. `http://fedoraproject.org/wiki/EPEL`.

[2] Fedora Account System. `https://admin.fedoraproject.org/accounts`.

[3] Fedora download site. `http://download.fedoraproject.org`.

[4] Fedora mirror tiering. `http://fedoraproject.org/wiki/Infrastructure/Mirroring/Tiering`.

[5] Fedora mirrorlist used by yum. `http://mirrors.fedoraproject.org/mirrorlist`.

[6] Fedora Project public mirror servers. `http://mirrors.fedoraproject.org`.

[7] Fedora Project statistics. `http://fedoraproject.org/wiki/Statistics`.

[8] Metalink. `http://www.metalinker.org`.

[9] MirrorManager. `http://fedorahosted.org/mirrormanager`.

[10] MirrorManager administrative interface. `https://admin.fedoraproject.org/mirrormanager/`.

[11] SQLAlchemy. `http://sqlalchemy.org`.

[12] SQLObject. `http://sqlobject.org`.

[13] TurboGears. `http://turbogears.org`.