About integration of rsync and Debian

Martin Pool `mbp@samba.org` $Date: 2002/05/14 03:10:09 $

Contents

1 Introduction **2**

2 Background **2**

2.1 The rsync algorithm 2

2.2 The rsync program 2

2.3 librsync 3

2.4 rproxy 3

2.5 Introduction to Debian 4

2.6 apt-proxy 4

3 Open Issues **5**

3.1 Compressed files cannot be differenced 5

3.2 rsync is too hard on servers 5

3.3 We should throw out the current code and start from scratch 5

3.4 rsync development roadmap 6

3.5 Goswin Brederlow's proposal to use the reverse rsync algorithm over HTTP Range requests . 6

3.6 rsync only compares files with the exact same name 7

3.7 rsync uses too much memory 7

3.8 rsync hangs? 7

3.9 Reduced network usage is not justified by increased CPU usage 7

3.10 rsync can't saturate a fast link 7

3.11 rsync is not actively developed 8

3.12 Servers should cache deltas 8

3.13 Possible patent on rsync? 8

3.14 Debian should store more metadata outside of the package file 8

3.15 Debian should distribute diffs or xdeltas between different versions of their packages 8

3.16 rsync should be used by `apt-get update` 9

A Revision history **9**

1 Introduction

It seems like there is an rsync thread on `debian-devel` every month or two. Rather than going round in circles yet again, I though I would summarise the issues in one place.

By way of background, I have been the maintainer of rsync for about a year, and I wrote the `librsync` and `rproxy` packages. Incidentally, I run Debian on most of the machines I use, so I do care about the two projects working together well.

I have tried to respond to all the issues raised on the list recently, but possibly have missed some. After putting many hours into rsync I suppose I am quite fond of it and may be a little biased, but I think I also know its strengths and problems more than most.

If there are issues ommitted by this document, or if you think the answers are incomplete, unbalanced, or incorrect, then please mail `mbp@samba.org`, `rsync@lists.samba.org` and/or `debian-devel@lists.debian.org`.

2 Background

2.1 The rsync algorithm

rsync is really two independent but related ideas: the *rsync algorithm* for delta compression, and its implementation in a mirroring program.

The rsync algorithm is described in detail in the ANU Technical Report included with the distribution and on the *rsync.samba.org* web site.

Briefly, the rsync algorithm provides an efficient means to transfer a file A from a source machine to a destination machine, when a similar file A' is already present on the destination. The destination machine sends a checksum of each consecutive block (of say 1kB) from A' to the source machine. The source machine searches through A for matching blocks. Whatever does not match must be different, and a description of these differences is sent to the destination.

The algorithm could be embodied in other forms and uses than the rsync program. Indeed, the algorithm has been reimplemented in programs other than rsync itself: in other languages (Rexx, CAML, Java), in a C library (`librsync`) and a derivative in xdelta. xdelta is a specialization of the algorithm for the case where the two files are local and can be directly compared to compute a minimal binary delta.

rsync deltas are much more efficient than diffs, for several reasons: most importantly, useful diff formats include context lines, which are wasted information. rsync gets the same benefit of making sure that the change is applied to a compatible version of the file, but in a much more efficient way. diffs suffer from being human-readable and therefore verbose, and cannot handle binary files. Both can be transparently compressed. diffs are an invaluable tool for software development, but not ideal for distribution of deltas.

2.2 The rsync program

The rsync mirroring program is similar in functionality to other tools such as `wget`, `ftpmirror`, `cvsup` and `rdist`. It has a number of functions that are useful in association with file mirroring, such as bandwidth limiting, access control, recursive mirroring, and selection of files to copy in various ways.

In general, rsync is substantially faster than other tools, because it uses delta compression, and because the protocol is designed to be efficient both in traffic and round trips. rsync can use `zlib` to compress traffic (and introduce double-`free()` security holes :-), at the option of the client. Compression may also be disabled by the server administrator.

rsync has the important property that it is generally idempotent: repeated runs of rsync, even if interrupted, make the contents of the destination machine converge towards those of the source machine.

Unlike wget and ftpmirror, rsync must be installed on both the client and the server. It uses its own protocol, rather than HTTP or FTP. This is probably the biggest practical drawback at the moment: one cannot use rsync from arbitrary web sites, but only from systems that have specifically chosen to support it. On the other hand, many important sites for free software do now support rsync, and the developers see them as an important constituency to support.

rsync can tunnel through a proxy server that supports `HTTP CONNECT`, a SOCKS server (using `socksify`), an ssh tunnel, or various other methods.

rsync can run as a daemon, similar to an `ftpd`, in which it is controlled by the `/etc/rsyncd.conf` file. The archetypal use is to offer public, anonymous, read-only access to a software archive but as with `ftpd` other configurations are possible and common.

rsync can also run across a tunnel program such as `ssh`, in which case it is very similar to `scp`. This mode is commonly used for making network backups, or uploading information to a web server.

rsync may also be used locally, which is a degenerate case with client and server connected across a unix-domain or localhost socket.

rsync runs on many varieties of Unix under both gcc and native compilers, and also under Cygwin on Microsoft platforms. There is somebody working on a VMS port.

2.3 librsync

`librsync` is a library that reimplements the rsync algorithm in a very flexible way, with the goal of allowing any reasonable mode of operation, and integration with any existing program. rsync does not currently link to librsync, but it might do so in the long term.

`librsync` currently uses an encoding format different to that of rsync 2.5, mostly because I did not want to be constrained by historical code.

`librsync` is at version 0.9.5 and roughly as stable as the version number suggests: it is used by Intermezzo and some other projects, but is not yet really mature. The `librsync` distribution comes with a tool `rdiff` that exposes the functionality as a Unix tool to shell scripts, etc. `librsync` is LGPL'd.

For example, you can imagine the server calculating and caching the checksums, or the client calculating the delta once and then sending it over UDP multicast to several destinations, or uuencoded in email. Neither of these would be straightforward to implement in the rsync codebase, but all can be done with rdiff.

2.4 rproxy

`librsync` is used in the *rproxy* `http://rproxy.samba.org/` program, which is a prototype of transparent integration of delta compression into HTTP.

rproxy implements delta compression of arbitrary HTTP content, whether dynamic or static. For example, rproxy gives very good compression on repeated visits to portal sites such as Slashdot, since it can transmit only the sections of the page modified since last time it was viewed. Regular HTTP caches, by contrast, must either hit on the whole page, or reload the whole page, and therefore do poorly on template-based dynamic content.

rproxy adds compatible extensions to HTTP headers, and can pass through HTTP caches. It requires upstream and downstream support, which at the moment means two installations of rproxy, but in the future could conceivably be integrated into Mozilla, Squid, Apache and similar programs.

rproxy is packaged in Debian and is moderately useful for people with slow links.

rproxy is not HTML-specific, but HTML is by far the most common case of dynamic HTTP content that can be delta-compressed. However, HTML documents tend to be fairly small, and as connectivity improves they're becoming of decreasing interest as a compression problem.

(My personal interest in this project declined significantly when I went from a 56k6 modem to ADSL at home. I realize many people in the world still have slow connections.)

There are some Internet Drafts from Jeffrey Mogul and others that add similar delta compression based on the server storing all possible deltas. Last time I looked, there did not seem much interest in wide adoption of these proposals.

Documenting a protocol extension to the standard expected by an RFC can be a lot of work. A beginning on this work has been made for rproxy, but more experiments with the implementation are needed before proposal as a standard.

rproxy is not being actively developed at the moment. Obviously I cannot answer why every programmer in the world does not work on this. Personally I think that developing rsync itself, and then librsync/rdiff, is likely to be more useful; I suspect other people interested in working in this area might have similar thoughts.

I don't think there are any problems in the code or project that would actively prevent anybody from working on it. I'd be happy to hand over maintenance to somebody else. Ben Elliston has expressed interest in looking after it in the last couple of months, and possibly it will be more active in the future.

2.5 Introduction to Debian

Debian is a free operating system developed by the cooperation of people all over the world. The most important relation of rsync to Debian is copying of software packages from developers to distribution servers to end users.

Many Debian users get their software by internet download from a public server, rather than pre-installed on a machine or on CD-ROM.

Debian software archives include both software source, binary packages for various architectures, and index metadata, primarily the `Packages` files.

Because of Debian's community development process, new packages are released very often. On a typical day in Debian's `unstable` release, there might be fifty new uploads.

Debian is quite different from other software distributions in shipping so many packages so frequently. The BSDs (as I understand it) base most of their development out of a CVS or CVSup tree, which inherently distributes coarse-grained deltas. Proprietary systems make binary releases, but much less frequently. Possibly the development branches of other distributions, such as Redhat "Rawhide" and Mandrake "Cooker" are similar.

2.6 apt-proxy

apt-proxy `http://apt-proxy.sourceforge.net/` is a caching proxy for Debian archives. It appears as an HTTP server to apt clients, and uses rsync, http, or ftp to connect to an upstream server.

Because rsync is less efficient than HTTP for transferring compressed files, `apt-proxy` can selectively use rsync for uncompressed Packages and files and HTTP or FTP for `.deb` files.

`apt-proxy`, unlike Squid, has domain knowledge about Debian archives and can therefore perform functions such as purging old packages from the cache.

The original apt-proxy was written by Paul 'Rusty' Russel. The current maintainer is Chris Halls.

3 Open Issues

3.1 Compressed files cannot be differenced

`gzip`, like most compression algorithms has the property that a change in the source file at one point will cause cascading changes in the output file from that point onwards, and therefore make delta compression more or less useless.

Although delta compression is not possible, rsync is still a very useful tool for mirroring compressed files. The efficient network protocol mean that it will generally be slightly faster and use less traffic than HTTP or FTP.

There is a patch called `-rsyncable` for gzip that fixes this behaviour: gzip files are basically broken up into blocks so that changes (including insertion or deletion) in the input file affect only the corresponding blocks in the output file. (The blocks are not of fixed size, but rather delimited by marker patterns at which a checksum hits a particular value, so they move as data is inserted or removed.)

I believe that this will merge into the upstream `zlib` soon and be on by default, at which point `.deb` files will delta-compress well. This patch does seem to be languishing, though, and it would be good to either get it into the upstream, or into Debian's own version. Needless to say it must be extremely thoroughly tested.

The scheme for containing changes is not specific to rsync, and might also be useful with `xdelta`, `rdiff`, or some other binary delta scheme invented in the future. It also does not require a copy of the old file when compressing the new one.

This scheme relies on the output file being determined only by the input file. As far as I know compression schemes like gzip, lzw, and bzip2 are deterministic in this way. (GnuPG, for example, is not, since it uses a random session key.)

Incidentally this would allow more efficient deltas between Debian ISO images.

Alternatively, you could distribute an uncompressed package tree that would rsync efficiently, but since the gzip patch should merge soon this seems unnecessary.

There is also the interesting possibility of using `dpkg-repack` to generate something similar to the previous `.deb` and then use it as the basis of an rsync download.

It could be possible to make an equivalent patch to `bzip2`, but possibly the large block size would cause trouble.

A patch to gzip which implements this behaviour is available from rsync CVS.

3.2 rsync is too hard on servers

If it is, then I think we should fix the problems, rather than invent a new system from scratch. I think the scalability problems are accidents of the current codebase, rather than anything inherent in the design.

3.3 We should throw out the current code and start from scratch

Some projects (Apache 2.0, Mozilla, ...) choose to do this; some concentrate on piecemeal improvement (Linux).

As Joel Spolsky recently observed, throwing the code out to start from scratch always seems like a nice idea, but rarely works out. Essentially you are opting for the devil you don't know, but he definitely exists.

Having dealt with some fairly crufty old code I sometimes feel like doing this in rsync, but I can also see the arguments against it. Starting from a new codebase would bring in plenty of new bugs (including security bugs), orphan existing users, and halt progress until it caught up to the same level. It would require carefully documenting all the existing internal protocols and behaviours, which is a nice idea but not clearly a good use of the developers' time.

If you throw out the code, you have the questions of whether or not to keep the current network protocol, and whether or not to keep the current command-line option.

Not being able to talk to old remote versions would be a real pain, and would certainly slow adoption. (Think of how unpopular SSH2 was when it wanted to be incompatible with SSH1, but use the same TCP port.) On the other hand, keeping the same network protocol limits your ability to redesign things.

There is some cruft in the command line syntax, but throwing it out would also break everybody's scripts, mental maps, and documentation.

Possibly rsync 3.0 will be largely rewritten. `librsync` is a start in that direction. If you want to do this, `librsync` might help you, but please think about it before you begin hacking.

3.4 rsync development roadmap

Debian's goals for rsync have to be balanced against the requirements of other users (including the mirror sites that distribute Debian) and the limited development resources.

As of April 2002, the 2.5.5 release is current and 2.5.6 is pending. These versions seem to work quite well and we encourage people to upgrade from stable 2.4.6 and previous versions. It is very important to the developers to establish a stable base release before starting out on new development.

A number of enhancements are planned for the 2.6 series. Some of them are discussed below, and more details can be found in the `TODO` file in the rsync distribution.

If you want rsync to progress faster, the best thing you can do is find reproducible failure cases and report them well, or to help us write a regression test suite.

Nobody is very actively working on rproxy or librsync as far as I know. Personally at the moment I feel supporting rsync itself is more useful.

3.5 Goswin Brederlow's proposal to use the reverse rsync algorithm over HTTP Range requests

Goswin Brederlow has a nice proposal that allows rsync compression using a regular HTTP server. It depends only upon rsyncable archives, and development of new client and support programs.

The idea, as I understand it, is that the checksums for each package should be pre-calculated and stored on the HTTP server along with the files they describe. (The checksum files should be on the order of a thousand times smaller than the archive file.)

The client program will download the checksum file over HTTP if it exists. The client can then search for common blocks in the local file, and therefore determine what new regions it must fetch from the server. The client then sends an HTTP request using the optional `Range` header for the relevant sections of the new file, and patches them onto the old file. The checksum file ought to also include a whole-file message digest which can be used to ensure that reassembly was successful.

This scheme has the great advantage that the server is entirely passive, and only needs to support standard HTTP. The checksum files can be generated once for each package either during upload, or by the administrator of a particular server.

(This is not the same as rproxy, which uses a different encoding and requires upstream support, but can transparently compress any request without requiring signature files on the server.)

This scheme could be fairly easily built on top of rdiff or librsync.

I think this sounds like a very promising scheme. I think rsync might still be better for people who want to copy large trees, but for users apt-getting single packages this would be a simple way to get delta-compression in, pending only –rsyncable files.

I'm not sure if all HTTP servers currently handle `Range` commands efficiently. This proposal would probably stress them more than is common at the moment.

Because it requires a special client, and special checksum files stored on the server it has a wider impact than just using rsync. It ought to be done in a non-Debian-specific way. Rather than adding knowledge about the deltas into apt-get itself, we ought to make a separate tool which downloads a delta over HTTP.

3.6 rsync only compares files with the exact same name

Even for uncompressed files, rsync will not currently try to use `linux-2.4.17.tar` to do delta-compression against `linux-2.4.18.tar`, because it cannot guess from the names that the two files are related.

Paul Russell has contributed a patch which adds a heuristic to detect this. It will probably be merged in 2.6.

3.7 rsync uses too much memory

rsync traverses the entire directory tree before it begins copying files. On machines with little virtual memory and a lot of files to copy this can be a problem.

Some problems in this area have been fixed in the 2.5 series. Other solutions involving internal restructuring of the `flist` and `hlink` code will be attempted in 2.6, and they should yield a substantial improvement.

We are not likely to change the general approach of traversing the tree up-front in the near future, because it is tightly tied to the network protocol. It might be attempted in the 3.0 timeframe, but it is not entirely clear that it is really a problem.

3.8 rsync hangs?

There were a number of deadlock bugs in the past. We believe they are all fixed in 2.5.5, but would welcome good bug reports to the contrary. Some of them were in fact Linux and Solaris kernel bugs, but they've been fixed by the relevant parties quite some time ago.

3.9 Reduced network usage is not justified by increased CPU usage

The balance point will vary for each administrator. It is hard to answer this universally. The balance is very different in countries other than the US.

I suspect CPU cycles are falling in price faster than network bandwidth, and for many people unused CPU cycles are wasted but network packets cost money.

3.10 rsync can't saturate a fast link

rsync with `-whole-file` (to turn off the delta algorithm and be more comparable to ftp) floods a 100Mbps link. If you have something faster than that to your upstream Debian mirror, you're a lucky person.

It is not inherently impossible to make rsync use `sendfile()` and `mmap()`; it's just that most people don't need them. (Apache did not use them by default either last time I looked, but that doesn't mean it's not fast enough for most people.)

3.11 rsync is not actively developed

This was a pretty fair accusation a year ago, but I don't think it is true anymore. We have made five releases this year, and the `diff -u` since 2.4.6, the last version from the previous maintainer, is 38305 lines.

Mail is answered promptly. The web site has been revised and cleaned. We're working on a regression test suite, code cleanup and documentation, and other quality issues.

I'm the most active developer at the moment, but there are a number of other people with experience in the code who regularly commit changes or send patches.

The stability expectations of our users require a somewhat disciplined approach to accepting patches, but I don't think that's a bad thing. The goal is to iterate through *freeze* and *flow* phases similar to the kernel.

The Debian rsync package maintainer, Phil Hands, has been somewhat inactive, but apparently he's back now. Colin Walters has been helping out.

If you have patches that were dropped, please send them through again.

3.12 Servers should cache deltas

This also smells like premature optimization. It might be useful, but it certainly seems less important than some other tasks.

3.13 Possible patent on rsync?

It has been suggested that some uses of rsync may conflict with a US patent. I am not sure of the current situation so I won't comment. I don't know of any suggestion that the rsync program as it currently exists infringes any patent.

I do know that rsync has been in use for many years by large and small organization with no complaints of patent infringement.

I would point out that linked lists, mark-and-copy garbage collection, and the Tab key are all patented too. Somebody who always carefully checked first for software patents would never write anything at all.

3.14 Debian should store more metadata outside of the package file

I can see some reasons why this might be a good idea, but it doesn't seem particularly relevant to rsync either way.

3.15 Debian should distribute diffs or xdeltas between different versions of their packages

A problem with this is that Debian releases packages very often, and so the number of delta files is large. This will be a problem for mirror sites who cannot carry all of the delta files for history, although of course the client could fall back to directly fetching the new package if the relevant deltas are not present.

Fetching a series of deltas which update the same area is less efficient than calculating the deltas directly.

Although this idea does not seem very practical for people using *unstable*, it could work well for *stable* and other situations where there are relatively few releases, or for people without network access. One could, for example, distribute a CD-ROM of xdeltas. On the other hand, since CD-ROMs are relatively large compared to the distribution it might be simpler to just distribute the new packages directly as is currently done.

3.16 rsync should be used by `apt-get update`

Even if there are few gains from compressing `.debs`, rsync might help in providing a more efficient transfer of the `Packages` file, which has small but frequent changes. This approach is used by `apt-proxy`.

A Revision history

```
$Log: rsync-and-debian.sgml,v $
Revision 1.10  2002/05/14 03:10:09  mbp
Add note about just transferring Package files.

Revision 1.9  2002/05/14 02:35:43  mbp
Update notes on apt-proxy based on mail from Chris Halls.

Revision 1.8  2002/04/12 02:04:15  mbp
More information about Goswin's idea, as I understand it.

Revision 1.7  2002/04/12 01:39:52  mbp
Lots more details about rproxy.  Thanks to Brian May for prompting me
to do this.

Revision 1.6  2002/04/12 00:43:06  mbp
Fix revision history SGML stuff.

Revision 1.5  2002/04/12 00:24:50  mbp
Fix mailing list name.
Add revision history.
```