GlusterFS 1.3 User Guide [DRAFT]

Nov 5, 2007

Vikas Gorur

Z Research

This is the user manual for GlusterFS 1.3.

Table of Contents

Acknowledgements

GlusterFS continues to be a wonderful and enriching experience for all of us involved. Anand Babu conceived GlusterFS, and leads its development. The development team consists of Anand Avati, Amar Tumballi, Basavanagowda Kanur, Harshavardhana Ranganath, Krishna Srinivas, Raghavendra G, and myself. Our CEO, Hitesh Chellani, ensures we get paid for hacking on GlusterFS.

GlusterFS development would not have been possible at this pace if not for our enthusiastic users. People from around the world have helped us with bug reports, performance numbers, and feature suggestions. A huge thanks to them all.

Matthew Paine - for RPMs & general enthu

Leonardo Rodrigues de Mello - for DEBs

Julian Perez & Adam D'Auria - for multi-server tutorial

Paul England - for HA spec

Brent Nelson - for many bug reports

Jacques Mattheij - for Europe mirror.

Patrick Negri - for TCP non-blocking connect.

Vikas Gorur (vikas@zresearch.com)
Z Research

1 Introduction

GlusterFS is a distributed filesystem. It works at the file level, not block level.

A network filesystem is one which allows us to access remote files. A distributed filesystem is one that stores data on multiple machines and makes them all appear to be a part of the same filesystem.

Need for distributed filesystems

- Scalability: A distributed filesystem allows us to store more data than what can be stored on a single machine.
- Redundancy: We might want to replicate crucial data on to several machines.
- Uniform access: One can mount a remote volume (for example your home directory) from any machine and access the same data.

1.1 Contacting us

You can reach us through the mailing list **gluster-devel** (gluster-devel@nongnu.org).

You can also find many of the developers on IRC, on the `#gluster` channel on Freenode (`irc.freenode.net`).

For commercial support, you can contact Z Research at:

Z Research Inc.,
3194 Winding Vista Common
Fremont, CA 94539
USA.
Phone: +1-510-5346801
Toll free: +18888136309

You can also email us at support@zresearch.com.

2 Installation and Invocation

2.1 Pre requisites

Before installing GlusterFS make sure you have the following components installed.

2.1.1 FUSE

You'll need FUSE version 2.6.0 or higher to use GlusterFS. You can omit installing FUSE if you want to build *only* the server. Note that you won't be able to mount a GlusterFS filesystem on a machine that does not have FUSE installed.

FUSE can be downloaded from: `http://fuse.sourceforge.net/`

To get the best performance from GlusterFS, however, it is recommended that you use our patched version of FUSE. See Section 6.2 [Patched FUSE], page 34 for details.

2.1.2 libibverbs (optional)

This is only needed if you want GlusterFS to use InfiniBand as the interconnect mechanism between server and client. You can get it from:

`http://www.openfabrics.org/downloads.htm`.

2.1.3 Bison and Flex

These should be already installed on most Linux systems. If not, use your distribution's normal software installation procedures to install them. Make sure you install the relevant developer packages also.

2.2 Getting GlusterFS

There are many ways to get hold of GlusterFS. For a production deployment, the recommended method is to download the latest release tarball. Release tarballs are available at: `http://gluster.org/download.php`.

If you want the bleeding edge development source, you can get them from the GNU Arch[1] repository. First you must install GNU Arch itself. Then register the GlusterFS archive by doing:

```
$ tla register-archive http://arch.sv.gnu.org/archives/gluster
```

Now you can check out the source itself:

```
$ tla get -A gluster@sv.gnu.org glusterfs--mainline--2.5
```

If you are on an RPM based system, you can also try RPMs contributed by Matthew Paine (`matt@mattsoftware.com`), for CentOS 5, available at:

`http://www.mattsoftware.com/msw_repo/centos/5/`

Leonardo Rodrigues de Mello (`l@lmello.eu.org`) has created Debian packages of GlusterFS. They are available at:

`http://lmello.virt-br.org/debian`

[1] `http://www.gnu.org/software/gnu-arch/`

2.3 Building

You can skip this section if you're installing from RPMs or DEBs.

GlusterFS uses the Autotools mechanism to build. As such, the procedure is straight-forward. First, change into the GlusterFS source directory.

```
$ cd glusterfs-1.3
```

If you checked out the source from the Arch repository, you'll need to run `./autogen.sh` first. Note that you'll need to have Autoconf and Automake installed for this.

Run `configure`.

```
$ ./configure
```

The configure script accepts the following options:

`--disable-ibverbs`
: Disable the InfiniBand transport mechanism.

`--disable-fuse-client`
: Disable the FUSE client.

`--disable-server`
: Disable building of the GlusterFS server.

Build and install GlusterFS.

```
# make install
```

The binaries (`glusterfsd` and `glusterfs`) will be by default installed in `/usr/local/sbin/`. Translator, scheduler, and transport shared libraries will be installed in `/usr/local/lib/glusterfs/<version>/`. Sample volume specification files will be in `/usr/local/etc/glusterfs/`. This document itself can be found in `/usr/local/share/doc/glusterfs/`. If you passed the `--prefix` argument to the configure script, then replace `/usr/local` in the preceding paths with the prefix.

2.4 Running GlusterFS

2.4.1 Server

The GlusterFS server is necessary to export storage volumes to remote clients (See Section 4.2.3 [Server protocol], page 15 for more info). This section documents the invocation of the GlusterFS server program and all the command-line options accepted by it.

`-f, --spec-file=<path>`
: Use the specified file as the volume specification.

`-l, --log-file=<path>`
: Specify the path for the log file.

`-L, --log-level=<level>`
: Set the log level for the server. Log level should be one of DEBUG, WARNING, ERROR, CRITICAL, or NONE.

`-N, --no-daemon`
: Run glusterfsd as a foreground process.

`-p, --pidfile=<path>`
: Path for the PID file.

`-?, --help`
: Show this help text.

`--usage`
: Display a short usage message.

`-V, --version`
: Show version information.

2.4.2 Client

The GlusterFS client process is necessary to access remote storage volumes and mount them locally using FUSE. This section documents the invocation of the client process and all its command-line arguments.

```
# glusterfs [options] <mountpoint>
```

The `mountpoint` is the directory where you want the GlusterFS filesystem to appear. Example:

```
# glusterfs -f /usr/local/etc/glusterfs-client.vol /mnt
```

The command-line options are detailed below.

`-a, --attr-timeout=<n>`
: Attribute timeout for inodes in the kernel, in seconds. Defaults to 1 second.

`-d, --direct-io-mode=[ENABLE|DISABLE]`
: Whether to force direct I/O on the FUSE file descriptor. Defaults to ENABLE.

`-e, --entry-timeout=<n>`
: Entry timeout for directory entries in the kernel, in seconds. Defaults to 1 second.

`-f, --spec-file=<path>`
: Use the specified file as the volume specification file. One of `--spec-file` or `--server` is mandatory.

`-l, --log-file=<path>`
: Specify the path for the log file.

`-L, --log-level=<level>`
: Set the log level for the server. Log level should be one of DEBUG, WARNING, ERROR, CRITICAL, or NONE.

`-n, --volume-name=<volume name>`
: Volume name in client spec to use. Defaults to the root volume.

`-N, --no-daemon`
: Run glusterfs as a foreground process.

`-p, --port=<n>`
: Connect to the specified port on the server. Default is 6996.

`-s, --server=<hostname or IP address>`
: Server to connect to fetch the client volume specification file. One of `--server` or `--spec-file` is mandatory.

`-t, --transport=<transport type>`
: Transport type to use to get the spec from server. See Section 4.2.1 [Transport modules], page 13, for more on transport modules.

`-?, --help`
: Give this help list

`-V, --version`
: Show version information.

2.5 A Tutorial Introduction

This section will show you how to quickly get GlusterFS up and running. We'll configure GlusterFS as a simple network filesystem, with one server and one client. In this mode of usage, GlusterFS can serve as a replacement for NFS.

We'll make use of two machines; call them *server* and *client* (If you don't want to setup two machines, just run everything that follows on the same machine). In the examples that follow, the shell prompts will use these names to clarify the machine on which the command is being run. For example, a command that should be run on the server will be shown with the prompt:

```
[root@server]#
```

Our goal is to make a directory on the *server* (say, `/export`) accessible to the *client*.

First of all, get GlusterFS installed on both the machines, as described in the previous sections. Make sure you have the FUSE kernel module loaded. You can ensure this by running:

```
[root@server]# modprobe fuse
```

Before we can run the GlusterFS client or server programs, we need to write two files called *volume specifications*. The volume spec describes the *translator tree* on a node. The next chapter will explain the concepts of 'translator' and 'volume specification' in detail. For now, just assume that the volume spec is like an NFS `/etc/export` file.

On the server, create a text file somewhere (we'll assume the path `/tmp/glusterfs-server.vol`) with the following contents.

```
volume colon-o
  type storage/posix
  option directory /export
end-volume

volume server
  type protocol/server
  subvolumes colon-o
  option transport-type tcp/server
  option auth.ip.colon-o.allow *
end-volume
```

A brief explanation of the file's contents. The first section defines a storage volume, named "colon-o" (the volume names are arbitrary), which exports the `/export` directory. The second section defines options for the translator which will make the storage volume accessible remotely. It specifies `colon-o` as a subvolume. This defines the *translator tree*, about which more will be said in the next chapter. The two options specify that the TCP protocol is to be used (as opposed to InfiniBand, for example), and that access to the storage volume is to be provided to clients with any IP address at all. If you wanted to restrict access to this server to only your subnet for example, you'd specify something like `192.168.1.*` in the second option line.

On the client machine, create the following text file (again, we'll assume the path to be `/tmp/glusterfs-client.vol`). Replace *server-ip-address* with the IP address of your server machine. If you are doing all this on a single machine, use `127.0.0.1`.

```
volume client
  type protocol/client
  option transport-type tcp/client
  option remote-host server-ip-address
  option remote-subvolume colon-o
end-volume
```

Now we need to start both the server and client programs. To start the server:

```
[root@server]# glusterfsd -f /tmp/glusterfs-server.vol
```

To start the client:

```
[root@client]# glusterfs -f /tmp/glusterfs-client.vol /mnt/glusterfs
```

You should now be able to see the files under the server's `/export` directory in the `/mnt/glusterfs` directory on the client. That's it; GlusterFS is now working as a network file system.

3 Concepts

3.1 Filesystems in Userspace

A filesystem is usually implemented in the kernel. Kernel development is much harder than userspace development. FUSE is a kernel module/library that allows us to write a filesystem completely in userspace.

FUSE consists of a kernel module which interacts with the userspace implementation using a device file `/dev/fuse`. When a process makes a syscall on a FUSE filesystem, VFS hands the request to the FUSE module, which writes the request to `/dev/fuse`. The userspace implementation polls `/dev/fuse`, and when a request arrives, processes it and writes the result back to `/dev/fuse`. The kernel then reads from the device file and returns the result to the user process.

3.2 Translator

The *translator* is the most important concept in GlusterFS. In fact, GlusterFS is nothing but a collection of translators working together, forming a translator *tree*.

The idea of a translator is perhaps best understood using an analogy. Consider the VFS in the Linux kernel. The VFS abstracts the various filesystem implementations (such as EXT3, ReiserFS, XFS, etc.) supported by the kernel. When an application calls the kernel to perform an operation on a file, the kernel passes the request on to the appropriate filesystem implementation.

For example, let's say there are two partitions on a Linux machine: `/`, which is an EXT3 partition, and `/usr`, which is a ReiserFS partition. Now if an application wants to open a file called, say, `/etc/fstab`, then the kernel will internally pass the request to the EXT3 implementation. If on the other hand, an application wants to read a file called `/usr/src/linux/CREDITS`, then the kernel will call upon the ReiserFS implementation to do the job.

The "filesystem implementation" objects are analogous to GlusterFS translators. A GlusterFS translator implements all the filesystem operations. Whereas in VFS there is a two-level tree (with the kernel at the root and all the filesystem implementation as its children), in GlusterFS there exists a more elaborate tree structure.

We can now define translators more precisely. A GlusterFS translator is a shared object (`.so`) that implements every filesystem call. GlusterFS translators can be arranged in an arbitrary tree structure (subject to constraints imposed by the translators). When GlusterFS receives a filesystem call, it passes it on to the translator at the root of the translator tree. The root translator may in turn pass it on to any or all of its children, and so on, until the leaf nodes are reached. The result of a filesystem call is communicated in the reverse fashion, from the leaf nodes up to the root node, and then on to the application.

So what might a translator tree look like?

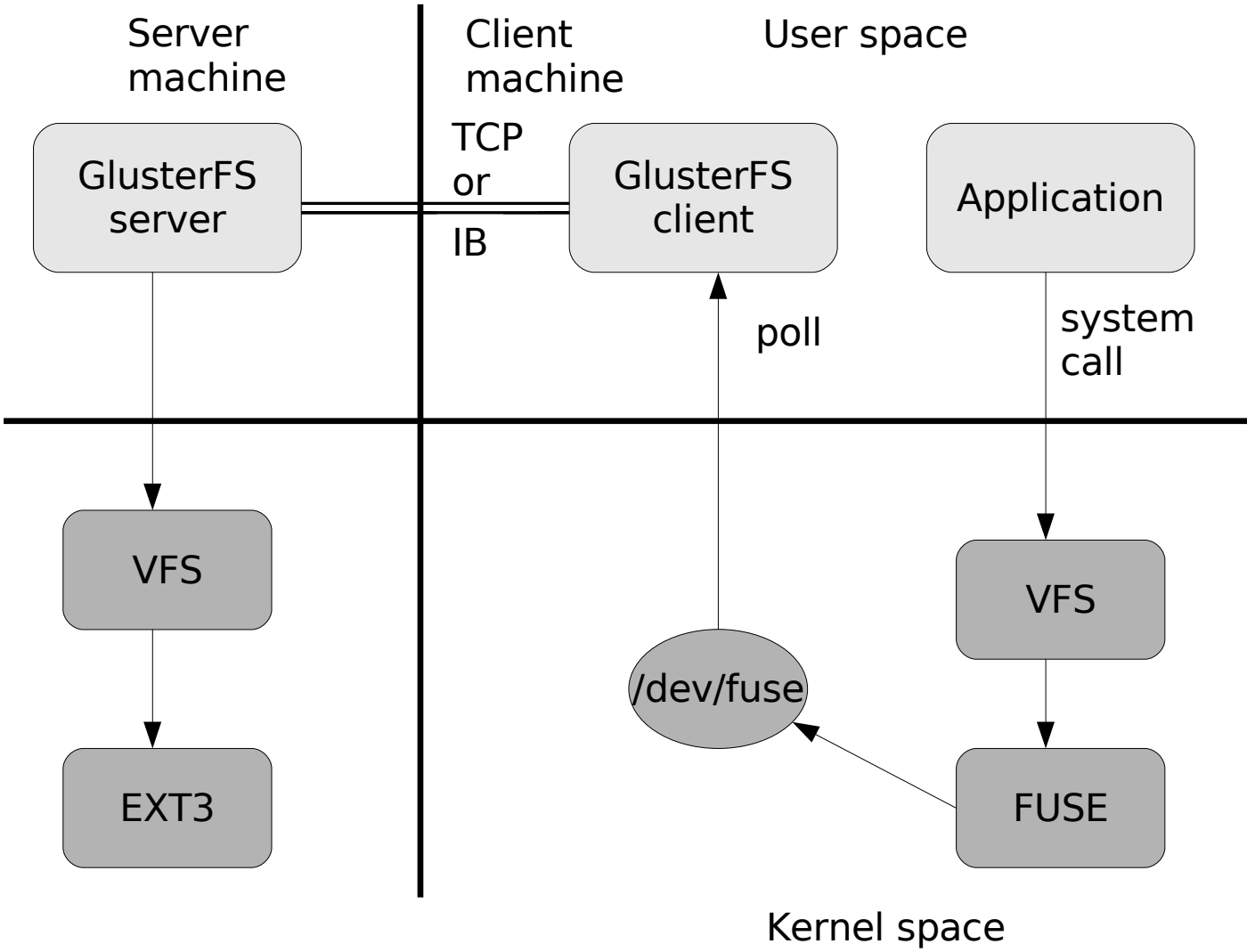


Fig 1. Control flow in GlusterFS

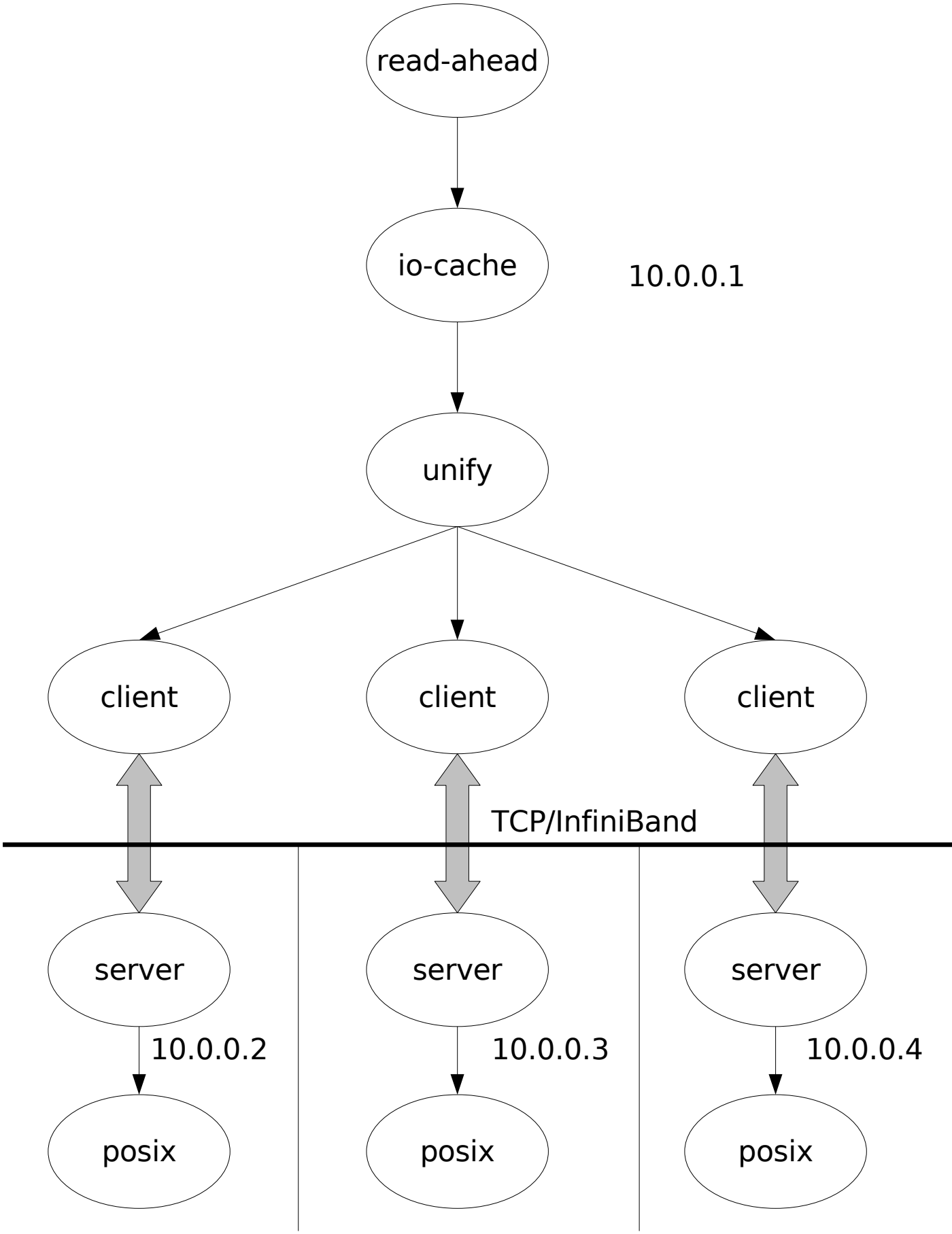

read-ahead
io-cache
10.0.0.1
unify
client
client
client
TCP/InfiniBand
server
server
server
10.0.0.2
10.0.0.3
10.0.0.4
posix
posix
posix

Fig 2. A sample translator tree

3.3 Volume specification file

The volume specification file describes the translator tree for both the server and client programs.

A volume specification file is a sequence of volume definitions. The syntax of a volume definition is explained below:

```
volume volume-name
  type translator-name
  option option-name option-value
  ...
  subvolumes subvolume1 subvolume2 ...
end-volume
  ...
```

volume-name
: An identifier for the volume. This is just a human-readable name, and can contain any alphanumeric character. For instance, "storage-1", "colon-o", or "forty-two".

translator-name
: Name of one of the available translators. Example: `protocol/client`, `cluster/unify`.

option-name
: Name of a valid option for the translator.

option-value
: Value for the option. Everything following the "option" keyword to the end of the line is considered the value; it is up to the translator to parse it.

subvolume1, *subvolume2*, . . .
: Volume names of sub-volumes. The sub-volumes must already have been defined earlier in the file.

There are a few rules you must follow when writing a volume specification file:

- Everything following a '`#`' is considered a comment and is ignored. Blank lines are also ignored.
- All names and keywords are case-sensitive.
- The order of options inside a volume definition does not matter.
- An option value may not span multiple lines.
- If an option is not specified, it will assume its default value.
- A sub-volume must have already been defined before it can be referenced. This means you have to write the specification file "bottom-up", starting from the leaf nodes of the translator tree and moving up to the root.

A simple example volume specification file is shown below:

```
# This is a comment line
volume client
 type protocol/client
 option transport-type tcp/client
 option remote-host localhost      # Also a comment
 option remote-subvolume brick
# The subvolumes line may be absent
end-volume

volume iot
 type performance/io-threads
 option thread-count 4
 subvolumes client
end-volume

volume wb
 type performance/write-behind
 subvolumes iot
end-volume
```

4 Translators

This chapter documents all the available GlusterFS translators in detail. Each translator section will show its name (for example, `cluster/unify`), briefly describe its purpose and workings, and list every option accepted by that translator and their meaning.

4.1 Storage Translators

The storage translators form the "backend" for GlusterFS. Currently, the only available storage translator is the POSIX translator, which stores files on a normal POSIX filesystem. A pleasant consequence of this is that your data will still be accessible if GlusterFS crashes or cannot be started.

Other storage backends are planned for the future. One of the possibilities is an Amazon S3 translator. Amazon S3 is an unlimited online storage service accessible through a web services API. The S3 translator will allow you to access the storage as a normal POSIX filesystem.[1]

4.1.1 POSIX

```
type storage/posix
```

The `posix` translator uses a normal POSIX filesystem as its "backend" to actually store files and directories. This can be any filesystem that supports extended attributes (EXT3, ReiserFS, XFS, ...). Extended attributes are used by some translators to store metadata, for example, by the AFR and stripe translators. See Section 4.3.2 [Automatic File Replication], page 20 and Section 4.3.3 [Stripe], page 22, respectively for details.

`directory <path>`
: The directory on the local filesystem which is to be used for storage.

4.2 Client and Server Translators

The client and server translator enable GlusterFS to export a translator tree over the network or access a remote GlusterFS server. These two translators implement GlusterFS's network protocol.

4.2.1 Transport modules

The client and server translators are capable of using any of the pluggable transport modules. Currently available transport modules are `tcp`, which uses a TCP connection between client and server to communicate; `ib-sdp`, which uses a TCP connection over InfiniBand, and `ibverbs`, which uses high-speed InfiniBand connections.

Each transport module comes in two different versions, one to be used on the server side and the other on the client side.

4.2.1.1 TCP

The TCP transport module uses a TCP/IP connection between the server and the client.

```
option transport-type tcp/client
option transport-type tcp/server
```

The TCP client module accepts the following options:

[1] Some more discussion about this can be found at:
http://developer.amazonwebservices.com/connect/message.jspa?messageID=52873

`non-blocking-connect [no|off|on|yes] (on)`
Whether to make the connection attempt asynchronous.

`remote-port <n> (6996)`
Server port to connect to.

`remote-host <hostname> *`
Hostname or IP address of the server.

The TCP server module accepts the following options:

`bind-address <address> (0.0.0.0)`
The local interface on which the server should listen to requests. Default is to listen on all interfaces.

`listen-port <n> (6996)`
The local port to listen on.

4.2.1.2 IB-SDP

```
option transport-type ib-sdp/client
option transport-type ib-sdp/server
```

kernel implements socket interface for ib hardware. SDP is over ib-verbs. This module accepts the same options as `tcp`

4.2.1.3 ibverbs

InfiniBand is a scalable switched fabric interconnect mechanism primarily used in high-performance computing. InfiniBand can deliver data throughput of the order of 10 Gbit/s, with latencies of 4-5 ms.

The `ib-verbs` transport accesses the InfiniBand hardware through the "verbs" API, which is the lowest level of software access possible and which gives the highest performance. On InfiniBand hardware, it is always best to use `ib-verbs`. Use `ib-sdp` only if you cannot get `ib-verbs` working for some reason.

If you are familiar with InfiniBand jargon, the mode is used by GlusterFS is "reliable connection-oriented channel transfer".

The `ib-verbs` transport module accepts the following options:

`ib-verbs-work-request-send-count <n> (64)`
Length of the send queue in datagrams. [Reason to increase/decrease?]

`ib-verbs-work-request-recv-count <n> (64)`
Length of the receive queue in datagrams. [Reason to increase/decrease?]

`ib-verbs-work-request-send-size <size> (128KB)`
Size of each datagram that is sent. [Reason to increase/decrease?]

`ib-verbs-work-request-recv-size <size> (128KB)`
Size of each datagram that is received. [Reason to increase/decrease?]

`ib-verbs-port <n> (1)`
Port number for ib-verbs.

`ib-verbs-mtu [256|512|1024|2048|4096] (2048)`
The Maximum Transmission Unit [Reason to increase/decrease?]

`ib-verbs-device-name <device-name> (first device in the list)`
InfiniBand device to be used.

For maximum performance, you should ensure that the send/receive counts on both the client and server are the same.

ib-verbs is preferred over ib-sdp.

4.2.2 Client

`type procotol/client`

The client translator enables the GlusterFS client to access a remote server's translator tree.

`transport-type [tcp,ib-sdp,ib-verbs] (tcp/client)`
The transport type to use. You should use the client versions of all the transport modules (`tcp/client`, `ib-sdp/client`, `ib-verbs/client`).

`remote-subvolume <volume_name> *`
The name of the volume on the remote host to attach to. Note that this is *not* the name of the `protocol/server` volume on the server. It should be any volume under the server.

`transport-timeout <n> (120- seconds)`
Inactivity timeout. If a reply is expected and no activity takes place on the connection within this time, the transport connection will be broken, and a new connection will be attempted.

4.2.3 Server

`type protocol/server`

The server translator exports a translator tree and makes it accessible to remote GlusterFS clients.

`client-volume-filename <path> (<CONFDIR>/glusterfs-client.vol)`
: The volume specification file to use for the client. This is the file the client will receive when it is invoked with the `--server` option (Section 2.4.2 [Client], page 5).

`transport-type [tcp,ib-verbs,ib-sdp] (tcp/server)`
: The transport to use. You should use the server versions of all the transport modules (`tcp/server`, `ib-sdp/server`, `ib-verbs/server`).

`auth.ip.<volume name>.allow <IP address wildcard pattern>`
: IP addresses of the clients that are allowed to attach to the specified volume. This can be a wildcard. For example, a wildcard of the form `192.168.*.*` allows any host in the `192.168.x.x` subnet to connect to the server.

4.3 Clustering Translators

The clustering translators are the most important GlusterFS translators, since it is these that make GlusterFS a cluster filesystem. These translators together enable GlusterFS to access an arbitrarily large amount of storage, and provide RAID-like redundancy and distribution over the entire cluster.

There are three clustering translators: **unify**, **afr**, and **stripe**. The unify translator aggregates storage from many server nodes. The afr translator provides file replication. The stripe translator allows a file to be spread across many server nodes. The following sections look at each of these translators in detail.

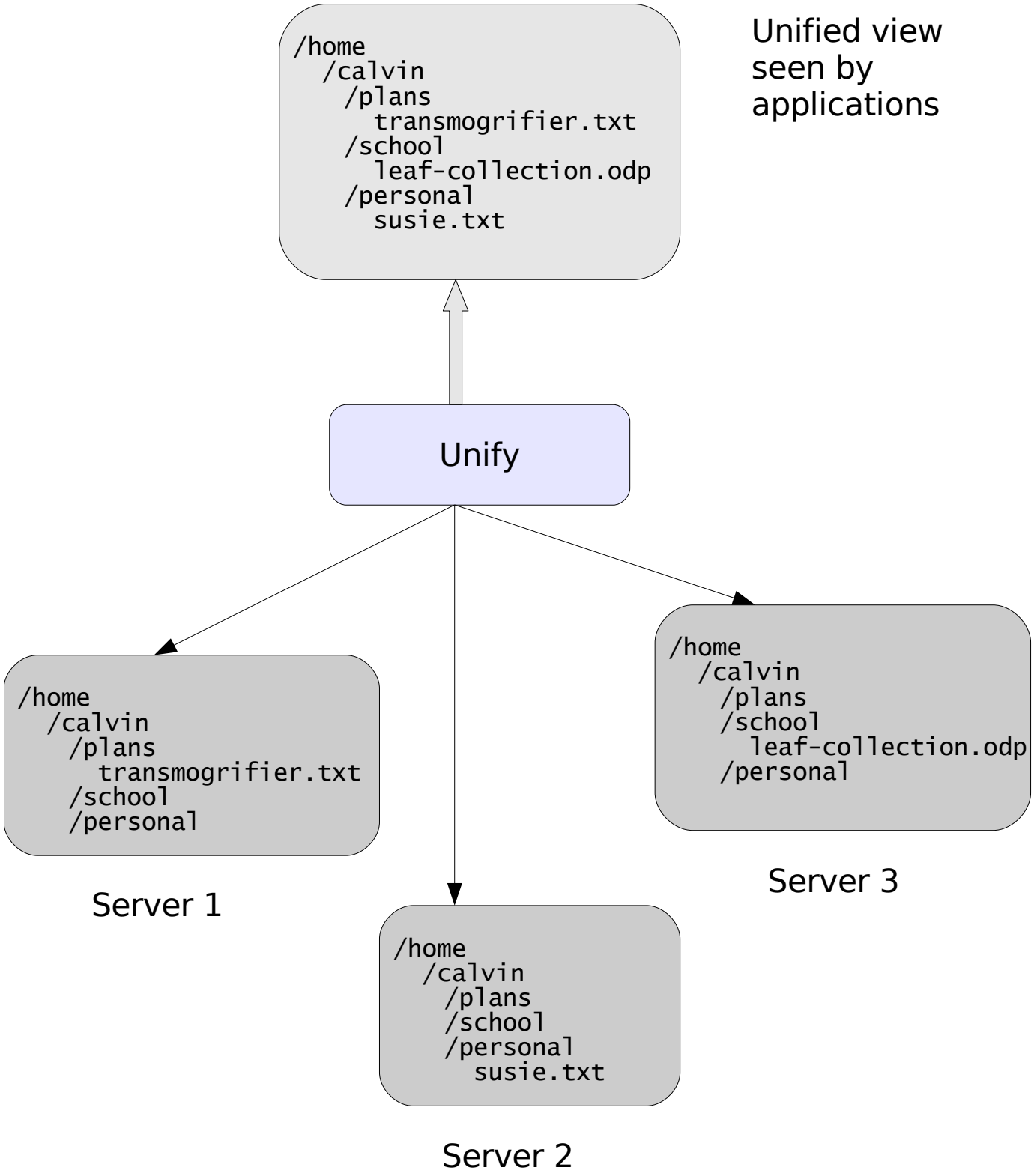
4.3.1 Unify

```
type cluster/unify
```

The unify translator presents a 'unified' view of all its sub-volumes. That is, it makes the union of all its sub-volumes appear as a single volume. It is the unify translator that gives GlusterFS the ability to access an arbitrarily large amount of storage.

For unify to work correctly, certain invariants need to be maintained across the entire network. These are:

- The directory structure of all the sub-volumes must be identical.
- A particular file can exist on only one of the sub-volumes. Phrasing it in another way, a pathname such as `/home/calvin/homework.txt`) is unique across the entire cluster.



Looking at the second requirement, you might wonder how one can accomplish storing redundant copies of a file, if no file can exist multiple times. To answer, we must remember that these invariants are from *unify's perspective*. A translator such as AFR at a lower level in the translator tree than unify may subvert this picture.

The first invariant might seem quite tedious to ensure. We shall see later that this is not so, since unify's *self-heal* mechanism takes care of maintaining it.

The second invariant implies that unify needs some way to decide which file goes where. Unify makes use of *scheduler* modules for this purpose.

When a file needs to be created, unify's scheduler decides upon the sub-volume to be used to store the file. There are many schedulers available, each using a different algorithm and suitable for different purposes.

The various schedulers are described in detail in the sections that follow.

4.3.1.1 ALU

```
option scheduler alu
```

ALU stands for "Adaptive Least Usage". It is the most advanced scheduler available in GlusterFS. It balances the load across volumes taking several factors in account. It adapts itself to changing I/O patterns according to its configuration. When properly configured, it can eliminate the need for regular tuning of the filesystem to keep volume load nicely balanced.

The ALU scheduler is composed of multiple least-usage sub-schedulers. Each sub-scheduler keeps track of a certain type of load, for each of the sub-volumes, getting statistics from the sub-volumes themselves. The sub-schedulers are these:

- disk-usage: The used and free disk space on the volume.
- read-usage: The amount of reading done from this volume.
- write-usage: The amount of writing done to this volume.
- open-files-usage: The number of files currently open from this volume.
- disk-speed-usage: The speed at which the disks are spinning. This is a constant value and therefore not very useful.

The ALU scheduler needs to know which of these sub-schedulers to use, and in which order to evaluate them. This is done through the `option alu.order` configuration directive.

Each sub-scheduler needs to know two things: when to kick in (the entry-threshold), and how long to stay in control (the exit-threshold). For example: when unifying three disks of 100GB, keeping an exact balance of disk-usage is not necesary. Instead, there could be a 1GB margin, which can be used to nicely balance other factors, such as read-usage. The disk-usage scheduler can be told to kick in only when a certain threshold of discrepancy is passed, such as 1GB. When it assumes control under this condition, it will write all subsequent data to the least-used volume. If it is doing so, it is unwise to stop right after the values are below the entry-threshold again, since that would make it very likely that the situation will occur again very soon. Such a situation would cause the ALU to spend most of its time disk-usage scheduling, which is unfair to the other sub-schedulers. The exit-threshold therefore defines the amount of data that needs to be written to the least-used disk, before control is relinquished again.

In addition to the sub-schedulers, the ALU scheduler also has "limits" options. These can stop the creation of new files on a volume once values drop below a certain threshold. For example, setting `option alu.limits.min-free-disk 5GB` will stop the scheduling of files to volumes that have less than 5GB of free disk space, leaving the files on that disk some room to grow.

The actual values you assign to the thresholds for sub-schedulers and limits depend on your situation. If you have fast-growing files, you'll want to stop file-creation on a disk much earlier than when hardly any of your files are growing. If you care less about disk-usage balance than about read-usage balance, you'll want a bigger disk-usage scheduler entry-threshold and a smaller read-usage scheduler entry-threshold.

For thresholds defining a size, values specifying "KB", "MB" and "GB" are allowed. For example: `option alu.limits.min-free-disk 5GB`.

```
alu.order <order> *
("disk-usage:write-usage:read-usage:open-files-usage:disk-speed")
alu.disk-usage.entry-threshold <size> (1GB)
alu.disk-usage.exit-threshold <size> (512MB)
alu.write-usage.entry-threshold <%> (25)
alu.write-usage.exit-threshold <%> (5)
alu.read-usage.entry-threshold <%> (25)
alu.read-usage.exit-threshold <%> (5)
alu.open-files-usage.entry-threshold <n> (1000)
alu.open-files-usage.exit-threshold <n> (100)
alu.limits.min-free-disk <%>
alu.limits.max-open-files <n>
```

4.3.1.2 Round Robin (RR)

```
option scheduler rr
```

Round-Robin (RR) scheduler creates files in a round-robin fashion. Each client will have its own round-robin loop. When your files are mostly similar in size and I/O access pattern, this scheduler is a good choice. RR scheduler checks for free disk space on the server before scheduling, so you can know when to add another server node. The default value of min-free-disk is 5% and is checked on file creation calls, with atleast 10 seconds (by default) elapsing between two checks.

Options:

`rr.limits.min-free-disk <%> (5)`
: Minimum free disk space a node must have for RR to schedule a file to it.

`rr.refresh-interval <t> (10 seconds)`
: Time between two successive free disk space checks.

4.3.1.3 Random

```
option scheduler random
```

The random scheduler schedules file creation randomly among its child nodes. Like the round-robin scheduler, it also checks for a minimum amount of free disk space before scheduling a file to a node.

`random.limits.min-free-disk <%> (5)`
: Minimum free disk space a node must have for random to schedule a file to it.

`random.refresh-interval <t> (10 seconds)`
: Time between two successive free disk space checks.

4.3.1.4 NUFA

```
option scheduler nufa
```

It is common in many GlusterFS computing environments for all deployed machines to act as both servers and clients. For example, a research lab may have 40 workstations each with its own storage. All of these workstations might act as servers exporting a volume as well as clients accessing the entire cluster's storage. In such a situation, it makes sense to store locally created files on the local workstation itself (assuming files are accessed most by the workstation that created them). The Non-Uniform File Allocation (NUFA) scheduler accomplishes that.

NUFA gives the local system first priority for file creation over other nodes. If the local volume does not have more free disk space than a specified amount (5% by default) then NUFA schedules files among the other child volumes in a round-robin fashion.

NUFA is named after the similar strategy used for memory access, NUMA[2].

`nufa.limits.min-free-disk <%> (5)`
: Minimum disk space that must be free (local or remote) for NUFA to schedule a file to it.

`nufa.refresh-interval <t> (10 seconds)`
: Time between two successive free disk space checks.

`nufa.local-volume-name <volume>`
: The name of the volume corresponding to the local system. This volume must be one of the children of the unify volume. This option is mandatory.

4.3.1.5 Namespace

Namespace volume needed because: - persistent inode numbers. - file exists even when node is down.

namespace files are simply touched. on every lookup it is checked.

`namespace <volume> *`
: Name of the namespace volume (which should be one of the unify volume's children).

`self-heal [on|off] (on)`
: Enable/disable self-heal. Unless you know what you are doing, do not disable self-heal.

4.3.1.6 Self Heal

* When a 'lookup()/stat()' call is made on directory for the first time, a self-heal call is made, which checks for the consistancy of its child nodes. If an entry is present in storage node, but not in namespace, that entry is created in namespace, and vica-versa. There is an writedir() API introduced which is used for the same. It also checks for permissions, and uid/gid consistencies.

* This check is also done when an server goes down and comes up.

* If one starts with an empty namespace export, but has data in storage nodes, a 'find .>/dev/null' or 'ls -lR >/dev/null' should help to build namespace in one shot. Even otherwise, namespace is built on demand when a file is looked up for the first time.

NOTE: There are some issues (Kernel 'Oops' msgs) seen with fuse-2.6.3, when someone deletes namespace in backend, when glusterfs is running. But with fuse-2.6.5, this issue is not there.

4.3.2 Automatic File Replication (AFR)

```
type cluster/afr
```

AFR provides RAID-1 like functionality for GlusterFS. AFR replicates files and directories across the subvolumes. Hence if AFR has four subvolumes, there will be four copies of all files and directories. AFR provides high-availability, i.e., in case one of the subvolumes go down (e. g. server crash, network disconnection) AFR will still service the requests using the redundant copies.

[2] Non-Uniform Memory Access: `http://en.wikipedia.org/wiki/Non-Uniform_Memory_Access`

AFR also provides self-heal functionality, i.e., in case the crashed servers come up, the outdated files and directories will be updated with the latest versions. AFR uses extended attributes of the backend file system to track the versioning of files and directories and provide the self-heal feature.

```
volume afr-example
 type cluster/afr
 subvolumes brick1 brick2 brick3
end-volume
```

This sample configuration will replicate all directories and files on brick1, brick2 and brick3.

All the read operations happen from the first alive child. If all the three sub-volumes are up, reads will be done from brick1; if brick1 is down read will be done from brick2. In case read() was being done on brick1 and it goes down, AFR transparently falls back to brick2.

The next release of GlusterFS will add the following features:

- Ability to specify the sub-volume from which read operations are to be done (this will help users who have one of the sub-volumes as a local storage volume).
- Allow scheduling of read operations amongst the sub-volumes in a round-robin fashion.

The order of the subvolumes list should be same across all the AFRs as they will be used for locking purposes.

4.3.2.1 Self Heal

AFR has self-heal feature, which updates the outdated file and directory copies by the most recent versions. For example consider the following config:

```
volume afr-example
 type cluster/afr
 subvolumes brick1 brick2
end-volume
```

4.3.2.2 File self-heal

Now if we create a file foo.txt on afr-example, the file will be created on brick1 and brick2. The file will have two extended attributes associated with it in the backend filesystem. One is trusted.afr.createtime and the other is trusted.afr.version. The trusted.afr.createtime xattr has the create time (in terms of seconds since epoch) and trusted.afr.version is a number that is incremented each time a file is modified. This increment happens during close (incase any write was done before close).

If brick1 goes down, we edit foo.txt the version gets incremented. Now the brick1 comes back up, when we open() on foo.txt AFR will check if their versions are same. If they are not same, the outdated copy is replaced by the latest copy and its version is updated. After the sync the open() proceeds in the usual manner and the application calling open() can continue on its access to the file.

If brick1 goes down, we delete foo.txt and create a file with the same name again i.e foo.txt. Now brick1 comes back up, clearly there is a chance that the version on brick1 being more than the version on brick2, this is where createtime extended attribute helps in deciding which the outdated copy is. Hence we need to consider both createtime and version to decide on the latest copy.

The version attribute is incremented during the close() call. Version will not be incremented in case there was no write() done. In case the fd that the close() gets was got by create() call, we also create the createtime extended attribute.

4.3.2.3 Directory self-heal

Suppose brick1 goes down, we delete foo.txt, brick1 comes back up, now we should not create foo.txt on brick2 but we should delete foo.txt on brick1. We handle this situation by having the createtime and version attribute on the directory similar to the file. when lookup() is done on the directory, we compare the createtime/version attributes of the copies and see which files needs to be deleted and delete those files and update the extended attributes of the outdated directory copy. Each time a directory is modified (a file or a subdirectory is created or deleted inside the directory) and one of the subvols is down, we increment the directory's version.

lookup() is a call initiated by the kernel on a file or directory just before any access to that file or directory. In glusterfs, by default, lookup() will not be called in case it was called in the past one second on that particular file or directory.

The extended attributes can be seen in the backend filesystem using the `getfattr` command. (`getfattr -n trusted.afr.version <file>`)

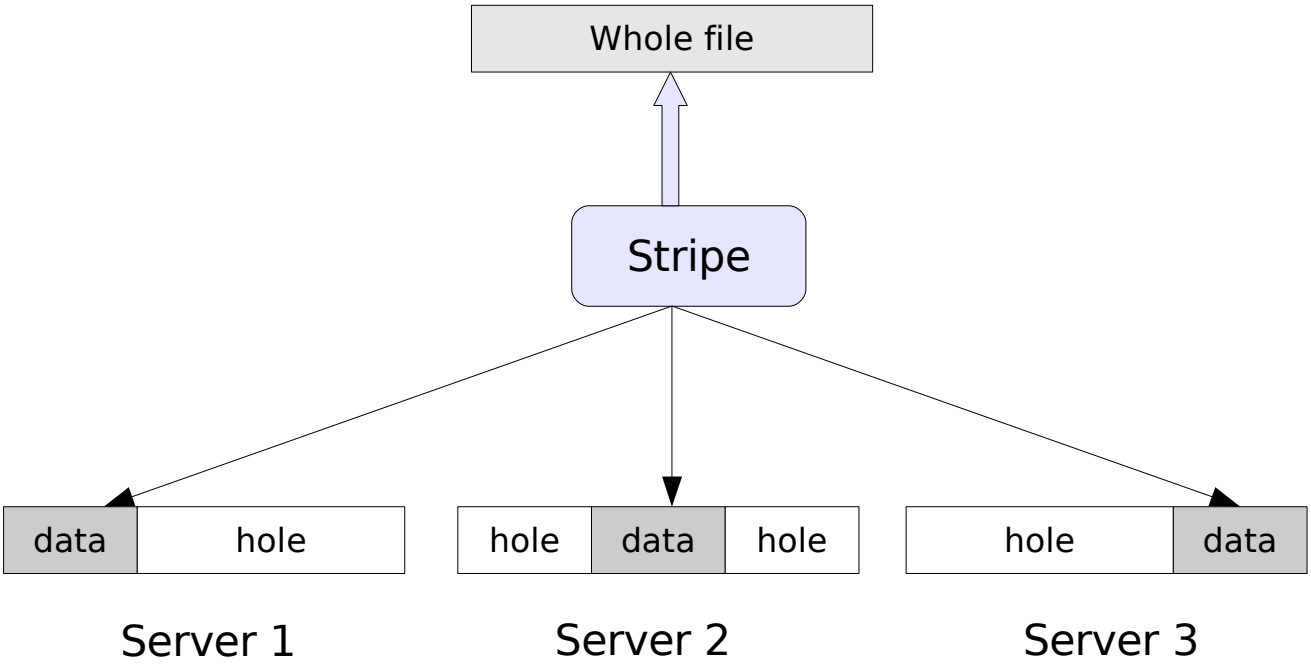
```
debug [on|off] (off)
self-heal [on|off] (on)
replicate <pattern> (*:1)
lock-node <child_volume> (first_child)
```

4.3.3 Stripe

`type cluster/stripe`

The stripe translator distributes the contents of a file over its sub-volumes. It does this by creating a file equal in size to the total size of the file on each of its sub-volumes. It then writes only a part of the file to each sub-volume, leaving the rest of it empty. These empty regions are called 'holes' in Unix terminology. The holes do not consume any disk space.

The diagram below makes this clear.



You can configure stripe so that only filenames matching a pattern are striped. You can also configure the size of the data to be stored on each sub-volume.

`block-size <pattern>:<size> (*:0 no striping)`
: Distribute files matching `<pattern>` over the sub-volumes, storing at least `<size>` on each sub-volume. For example,

```
option block-size *.mpg:1M
```

distributes all files ending in `.mpg`, storing at least 1 MB on each sub-volume.

Any number of `block-size` option lines may be present, specifying different sizes for different file name patterns.

4.4 Performance Translators

4.4.1 Read Ahead

`type performance/read-ahead`

The read-ahead translator pre-fetches data in advance on every read. This benefits applications that mostly process files in sequential order, since the next block of data will already be available by the time the application is done with the current one.

Additionally, the read-ahead translator also behaves as a read-aggregator. Many small read operations are combined and issued as fewer, larger read requests to the server.

Read-ahead deals in "pages" as the unit of data fetched. The page size is configurable, as is the "page count", which is the number of pages that are pre-fetched.

Read-ahead is best used with InfiniBand (using the ib-verbs transport, see Section 4.2.1.3 [ibverbs], page 14). On FastEthernet and Gigabit Ethernet networks, GlusterFS can achieve the link-maximum throughput even without read-ahead, making it quite superflous.

Note that read-ahead only happens if the reads are perfectly sequential. If your application accesses data in a random fashion, using read-ahead might actually lead to a performance loss, since read-ahead will pointlessly fetch pages which won't be used by the application.

Options:

`page-size <n> (256KB)`
: The unit of data that is pre-fetched.

`page-count <n> (2)`
: The number of pages that are pre-fetched.

`force-atime-update [on|off|yes|no] (off|no)`
: Whether to force an access time (atime) update on the file on every read. Without this, the atime will be slightly imprecise, as it will reflect the time when the read-ahead translator read the data, not when the application actually read it.

4.4.2 Write Behind

`type performance/write-behind`

The write-behind translator improves the latency of a write operation. It does this by relegating the write operation to the background and returning to the application even as the write is in progress. Using the write-behind translator, successive write requests can be pipelined. This mode of write-behind operation is best used on the client side, to enable decreased write latency for the application.

The write-behind translator can also aggregate write requests. If the `aggregate-size` option is specified, then successive writes upto that size are accumulated and written in a single operation. This mode of operation is best used on the server side, as this will decrease the disk's head movement when multiple files are being written to in parallel.

The `aggregate-size` option has a default value of 128KB. Although this works well for most users, you should always experiment with different values to determine the one that will deliver maximum performance. This is because the performance of write-behind depends on your interconnect, size of RAM, and the work load.

`aggregate-size <n> (128KB)`
: Amount of data to accumulate before doing a write

`flush-behind [on|yes|off|no] (off|no)`

4.4.3 IO Threads

`type performance/io-threads`

The IO threads translator is intended to increase the responsiveness of the server to metadata operations by doing file I/O (read, write) in a background thread. Since the GlusterFS server is single-threaded, using the IO threads translator can significantly improve performance. This translator is best used on the server side, loaded just below the server protocol translator.

IO threads operates by handing out read and write requests to a separate thread. The total number of threads in existence at a time is constant, and configurable. You can also set a maximum limit on the amount of data to be queued with IO threads, using the `cache-size`

option. If more than `cache-size` worth of operations are pending, any further requests will block.

`thread-count <n> (1)`
: Number of threads to use.

`cache-size <n> (64MB)`
: Maximum amount of data allowed to be pending inside io-threads.

4.4.4 IO Cache

```
type performance/io-cache
```

The IO cache translator caches data that has been read. This is useful if many applications read the same data multiple times, and if reads are much more frequent than writes (for example, IO caching may be useful in a web hosting environment, where most clients will simply read some files and only a few will write to them).

The IO cache translator reads data from its child in `page-size` chunks. It caches data upto `cache-size` bytes. The cache is maintained as a prioritized least-recently-used (LRU) list, with priorities determined by user-specified patterns to match filenames.

When the IO cache translator detects a write operation, the cache for that file is flushed.

The IO cache translator periodically verifies the consistency of cached data, using the modification times on the files. The verification timeout is configurable.

`page-size <n> (128KB)`
: Size of a page.

`cache-size (n) (32MB)`
: Total amount of data to be cached.

`force-revalidate-timeout <n> (1)`
: Timeout to force a cache consistency verification, in seconds.

`priority <pattern> (*:0)`
: Filename patterns listed in order of priority.

4.4.5 Booster

```
type performance/booster
```

The booster translator gives applications a faster path to communicate read and write requests to GlusterFS. Normally, all requests to GlusterFS from applications go through FUSE, as indicated in Section 3.1 [Filesystems in Userspace], page 8. Using the booster translator in conjunction with the GlusterFS booster shared library, an application can bypass the FUSE path and send read/write requests directly to the GlusterFS client process.

The booster mechanism consists of two parts: the booster translator, and the booster shared library. The booster translator is meant to be loaded on the client side, usually at the root of the translator tree. The booster shared library should be `LD_PRELOAD`ed with the application.

The booster translator when loaded opens a Unix domain socket and listens for read/write requests on it. The booster shared library intercepts read and write system calls and sends the requests to the GlusterFS process directly using the Unix domain socket, bypassing FUSE. This leads to superior performance.

Once you've loaded the booster translator in your volume specification file, you can start your application as:

```
$ LD_PRELOAD=/usr/local/bin/glusterfs-booster.so your_app
```

The booster translator accepts no options.

4.5 Features Translators

4.5.1 POSIX Locks

`type features/posix-locks`

This translator provides storage independent POSIX record locking support (fcntl locking). Typically you'll want to load this on the server side, just above the POSIX storage translator. Using this translator you can get both advisory locking and mandatory locking support. `flock()` locks are currently not supported.

Caveat: Consider a file that does not have its mandatory locking bits (+setgid, -group execution) turned on. Assume that this file is now opened by a process on a client that has the write-behind xlator loaded. The write-behind xlator does not cache anything for files which have mandatory locking enabled, to avoid incoherence. Let's say that mandatory locking is now enabled on this file through another client. The former client will not know about this change, and write-behind may erroneously report a write as being successful when in fact it would fail due to the region it is writing to being locked.

There seems to be no easy way to fix this. To work around this problem, it is recommended that you never enable the mandatory bits on a file while it is open.

`mandatory [on|off] (on)`
: Turns mandatory locking on.

4.5.2 Fixed ID

`type features/fixed-id`

The fixed ID translator makes all filesystem requests from the client to appear to be coming from a fixed, specified UID/GID, regardless of which user actually initiated the request.

`fixed-uid <n> [if not set, not used]`
: The UID to send to the server

`fixed-gid <n> [if not set, not used]`
: The GID to send to the server

4.6 Miscallaneous Translators

4.6.1 ROT-13

`type encryption/rot-13`

ROT-13 is a toy translator that can "encrypt" and "decrypt" file contents using the ROT-13 algorithm. ROT-13 is a trivial algorithm that rotates each alphabet by thirteen places. Thus, 'A' becomes 'N', 'B' becomes 'O', and 'Z' becomes 'M'.

It goes without saying that you shouldn't use this translator if you need *real* encryption (a future release of GlusterFS will have real encryption translators).

```
encrypt-write [on|off] (on)
          Whether to encrypt on write

decrypt-read [on|off] (on)
          Whether to decrypt on read
```

4.6.2 Trace

`type debug/trace`

The trace translator is intended for debugging purposes. When loaded, it logs all the system calls received by the server or client (wherever trace is loaded), their arguments, and the results. You must use a GlusterFS log level of DEBUG (See Section 2.4 [Running GlusterFS], page 4) for trace to work.

Sample trace output (lines have been wrapped for readability):

```
2007-10-30 00:08:58 D [trace.c:1579:trace_opendir] trace: callid: 68
(*this=0x8059e40, loc=0x8091984 {path=/iozone3_283, inode=0x8091f00},
 fd=0x8091d50)

2007-10-30 00:08:58 D [trace.c:630:trace_opendir_cbk] trace:
(*this=0x8059e40, op_ret=4, op_errno=1, fd=0x8091d50)

2007-10-30 00:08:58 D [trace.c:1602:trace_readdir] trace: callid: 69
(*this=0x8059e40, size=4096, offset=0 fd=0x8091d50)

2007-10-30 00:08:58 D [trace.c:215:trace_readdir_cbk] trace:
(*this=0x8059e40, op_ret=0, op_errno=0, count=4)

2007-10-30 00:08:58 D [trace.c:1624:trace_closedir] trace: callid: 71
(*this=0x8059e40, *fd=0x8091d50)

2007-10-30 00:08:58 D [trace.c:809:trace_closedir_cbk] trace:
(*this=0x8059e40, op_ret=0, op_errno=1)
```

5 Usage Scenarios

5.1 High Availability Setup

This section is based on the High Availability tutorial written by Paul England on the GlusterFS wiki.

Introduction

The configuration examples in this article will show how to configure three servers, each with a single storage device, with the aim of creating a completely Redundant Server-Side GlusterFS Storage Cluster.

Basically we're trying to create a traditional SAN where all of the redundancy is handled on the server side. The clients are not concerned with maintaining data integrity and focus on their other primary roles.

Basic Overview

Here you can see that Automatic File Replication and the Unify Translators happen on a separate server-side network. Each file is replicated three times which will give us 3 exact copies of the entire Namespace and Data volumes. Also in the event that a server is removed and added to the cluster the Self-Heal ability of the Unify Translator will ensure data remains intact.

This configuration aims to ensure the maximum amount of Redundancy and High Availability of Data. It is possible to lose 2/3 Storage Servers and yet the cluster will still operate. When the servers come back online all Self-Heal traffic will occur on a separate network and will not effect the Storage Clients.

Possible Problems

We are still investigating mount point redundancy

This is not entirely scalable, however for High Availability this is not really a problem.

There is a lot of Network and Disk IO load placed on the Server-Side.

Version

The configuration files below were designed for use with glusterfs-1.3-pre5.2

The Configuration Files

Each of the servers configurations are almost identical except for the IP Addresses change...

/etc/glusterfs/glusterfs-server.vol

Each server has a slightly different configuration to this as they will refer to a local posix volume and two remote client volumes. Essentially the only thing that changes are the names of the volumes and the IP Addresses.

```
volume mailspool-ds
        type storage/posix
        option directory /home/export/mailspool
end-volume

volume mailspool-ns
        type storage/posix
        option directory /home/export/mailspool-ns
end-volume

volume mailspool-santa2-ds
        type protocol/client
        option transport-type tcp/client
        option remote-host 192.168.252.2
        option remote-subvolume mailspool-ds
end-volume

volume mailspool-santa2-ns
        type protocol/client
        option transport-type tcp/client
        option remote-host 192.168.252.2
        option remote-subvolume mailspool-ns
end-volume

volume mailspool-santa3-ds
        type protocol/client
        option transport-type tcp/client
        option remote-host 192.168.252.3
        option remote-subvolume mailspool-ds
end-volume

volume mailspool-santa3-ns
        type protocol/client
        option transport-type tcp/client
        option remote-host 192.168.252.3
        option remote-subvolume mailspool-ns
end-volume

volume mailspool-ns-afr
        type cluster/afr
        subvolumes mailspool-ns mailspool-santa2-ns mailspool-santa3-ns
        option replicate *:3
end-volume

volume mailspool-ds-afr
        type cluster/afr
        subvolumes mailspool-ds mailspool-santa2-ds mailspool-santa3-ds
        option replicate *:3
end-volume

volume mailspool-unify
        type cluster/unify
        subvolumes mailspool-ds-afr
        option namespace mailspool-ns-afr
        option scheduler rr
end-volume
```

/etc/glusterfs/glusterfs-client.vol

Currently we are using DNS Round Robin to connect to the cluster until the High Availability Translator in version 1.4 is available.

```
volume santa
  type protocol/client
  option transport-type tcp/client             # for TCP/IP transport

  # DNS Round Robin pointing towards all 3 GlusterFS servers
  option remote-host roundrobin.gluster.local
  option remote-subvolume mailspool            # name of the remote volume
end-volume

volume writeback
  type performance/write-behind
  option aggregate-size 131072 # unit in bytes
  subvolumes santa1
end-volume

volume readahead
  type performance/read-ahead
  option page-size 65536     # unit in bytes
  option page-count 16       # cache per file  = (page-count x page-size)
  subvolumes writeback
end-volume
```

Starting it up

On each of the servers...

```
# glusterfsd -L NORMAL -l /var/log/glusterfs/glusterfsd.log
```

And on the client server...

```
# glusterfs -L DEBUG -l /var/log/glusterfs/glusterfsd.log
            -f /etc/glusterfs/glusterfs-client.vol /mnt/happy
```

5.2 Advanced Striping

5.2.1 Mixed Storage Requirements

There are two ways of scheduling the I/O. One at file level (using unify translator) and other at block level (using stripe translator). Striped I/O is good for files that are potentially large and require high parallel throughput (for example, a single file of 400GB being accessed by 100s and 1000s of systems simultaneously and randomly). For most of the cases, file level scheduling works best.

In the real world, it is desirable to mix file level and block level scheduling on a single storage volume. Alternatively users can choose to have two separate volumes and hence two mount points, but the applications may demand a single storage system to host both.

This document explains how to mix file level scheduling with stripe.

5.2.2 Configuration Brief

This setup demonstrates how users can configure unify translator with appropriate I/O scheduler for file level scheduling and strip for only matching patterns. This way, GlusterFS chooses appropriate I/O profile and knows how to efficiently handle both the types of data.

A simple technique to achieve this effect is to create a stripe set of unify and stripe blocks, where unify is the first sub-volume. Files that do not match the stripe policy passed on to first unify sub-volume and inturn scheduled arcoss the cluster using its file level I/O scheduler.

5.2.3 Preparing GlusterFS Envoronment

Create the directories /export/namespace, /export/unify and /export/stripe on all the storage bricks.

Place the following server and client volume spec file under /etc/glusterfs (or appropriate installed path) and replace the IP addresses / access control fields to match your environment.

```
## file: /etc/glusterfs/glusterfs-server.vol
 volume posix-unify
         type storage/posix
         option directory /export/for-unify
 end-volume

 volume posix-stripe
         type storage/posix
         option directory /export/for-stripe
 end-volume

 volume posix-namespace
         type storage/posix
         option directory /export/for-namespace
 end-volume

 volume server
         type protocol/server
         option transport-type tcp/server
         option auth.ip.posix-unify.allow 192.168.1.*
         option auth.ip.posix-stripe.allow 192.168.1.*
         option auth.ip.posix-namespace.allow 192.168.1.*
         subvolumes posix-unify posix-stripe posix-namespace
 end-volume
```

```
## file: /etc/glusterfs/glusterfs-client.vol
  volume client-namespace
    type protocol/client
    option transport-type tcp/client
    option remote-host 192.168.1.1
    option remote-subvolume posix-namespace
  end-volume

  volume client-unify-1
    type protocol/client
    option transport-type tcp/client
    option remote-host 192.168.1.1
    option remote-subvolume posix-unify
  end-volume

  volume client-unify-2
    type protocol/client
    option transport-type tcp/client
    option remote-host 192.168.1.2
    option remote-subvolume posix-unify
  end-volume

  volume client-unify-3
    type protocol/client
    option transport-type tcp/client
    option remote-host 192.168.1.3
    option remote-subvolume posix-unify
  end-volume

  volume client-unify-4
    type protocol/client
    option transport-type tcp/client
    option remote-host 192.168.1.4
    option remote-subvolume posix-unify
  end-volume

  volume client-stripe-1
    type protocol/client
    option transport-type tcp/client
    option remote-host 192.168.1.1
    option remote-subvolume posix-stripe
  end-volume

  volume client-stripe-2
    type protocol/client
    option transport-type tcp/client
    option remote-host 192.168.1.2
    option remote-subvolume posix-stripe
  end-volume

  volume client-stripe-3
    type protocol/client
    option transport-type tcp/client
    option remote-host 192.168.1.3
    option remote-subvolume posix-stripe
  end-volume
```

Bring up the Storage

Starting GlusterFS Server: If you have installed through binary package, you can start the service through init.d startup script. If not:

```
[root@server]# glusterfsd
```

Mounting GlusterFS Volumes:

```
[root@client]# glusterfs -s [BRICK-IP-ADDRESS] /mnt/cluster
```

Improving upon this Setup

Infiniband Verbs RDMA transport is much faster than TCP/IP GigE transport.

Use of performance translators such as read-ahead, write-behind, io-cache, io-threads, booster is recommended.

Replace round-robin (rr) scheduler with ALU to handle more dynamic storage environments.

6 Performance

- effect of direct_io mode.

6.1 Patched FUSE

The GlusterFS project maintains a patched version of FUSE meant to be used with GlusterFS. The patches increase GlusterFS performance. It is recommended that all users use the patched FUSE.

The patched FUSE tarball can be downloaded from:

`ftp://ftp.zresearch.com/pub/gluster/glusterfs/fuse/`

The specific changes made to FUSE are:

- The communication channel size between FUSE kernel module and GlusterFS has been increased to 1MB, permitting large reads and writes to be sent in bigger chunks.
- The kernel's read-ahead boundry has been extended upto 1MB.
- Block size returned in the `stat()`/`fstat()` calls tuned to 1MB, to make cp and similar commands perform I/O using that block size.
- flock() locking support added (although some rework in GlusterFS is needed for perfect compliance).

6.2 DNS failover

GlusterFS client will try connecting to all the IP addresses of a hostname in round-robin fashion. Using this you can implement failover.

7 Troubleshooting

7.1 GlusterFS error messages

7.1.1 Server errors

```
glusterfsd: FATAL: could not open specfile:
'/etc/glusterfs/glusterfs-server.vol'
```

glusterfsd server requires volume specification file under /etc/glusterfs/glusterfs-server.vol. Default installation will only provide /etc/glusterfs/glusterfs-server.vol.sample sample file. You need to copy it to /etc/glusterfs/glusterfs-server.vol actual file.

```
gf_log_init: failed to open logfile "/usr/var/log/glusterfs/glusterfsd.log"
             (Permission denied)
```

Check if you are running as root user. Use 'whoami' utility.

7.1.2 Client errors

```
fusermount: failed to access mountpoint /mnt:
            Transport endpoint is not connected
```

umount /mnt and mount again.

Error "Transport endpoint is not connected"

GlusterFS mount failed. Start glusterfs client in DEBUG mode and look out for descriptive error messages.

connect to server failed

SERVER-ADDRESS: Connection refused

GluserFS Server is not running or dead. Also check your network connections or firewall settings. To check if the server is reachable, try the following command:

telnet IP-ADDRESS 6996

If server is accessible, your 'telnet' command should connect and block. If not you will see an error message like telnet: Unable to connect to remote host: Connection refused. '6996' is the default GlusterFS port. If you have customized it, then use the corresponding port instead.

gf_log_init: failed to open logfile "/usr/var/log/glusterfs/glusterfs.log" (Permission denied)

glusterfs: failed to open logfile "/usr/var/log/glusterfs/glusterfs.log"

Check if you are running as root user. Use 'whoami' utility.

7.2 FUSE error messages

`modprobe` fuse fails with "Unknown symbol in module, or unknown parameter".

If you are using fuse-2.6.x on Redhat Enterprise Linux Work Station 4 and Advanced Server 4 with 2.6.9-42.ELlargesmp, 2.6.9-42.ELsmp, 2.6.9-42.EL kernels and get this error while loading fuse kernel module, you need to apply the following patch.

For fuse-2.6.2:

`http://ftp.zresearch.com/pub/gluster/glusterfs/fuse/fuse-2.6.2-rhel-build.patch`

For fuse-2.6.3:

`http://ftp.zresearch.com/pub/gluster/glusterfs/fuse/fuse-2.6.3-rhel-build.patch`

7.3 AppArmour and GlusterFS

Under OpenSuSE GNU/Linux, AppArmour security feature did not allow GlusterFS to create temporary files or network socket connection even as root user. You will see the error messages like Unable to open log file: Operation not permitted or Connection refused error. Disabling apparmour using YaST tool or properly configuring apparmour to recognize glusterfsd or glusterfs / fusermount should solve the problem.

GlusterFS log files.

7.4 Reporting a bug

If you encounter a bug in GlusterFS, please follow the below guidelines when you report it to the mailing list. Be sure to report it! User feedback is crucial to the health of the project and we value it highly.

7.4.1 General instructions

When running GlusterFS in a non-production environment, be sure to build it with the following command:

```
$ make CFLAGS='-g -O0 -DDEBUG'
```

This includes debugging information which will be helpful in getting backtraces (see below) and also disable optimization. Enabling optimization can result in incorrect line numbers being reported to gdb.

7.4.2 Volume specification files

Attach all relevant server and client spec files you were using when you encountered the bug. Also tell us details of your setup, i.e., how many clients and how many servers.

7.4.3 Log files

Set the loglevel of your client and server programs to DEBUG (by passing the -L DEBUG option) and attach the log files with your bug report. Obviously, if only the client is failing (for example), you only need to send us the client log file.

7.4.4 Backtrace

If GlusterFS has encountered a segmentation fault or has crashed for some other reason, include the backtrace with the bug report. You can get the backtrace using the following procedure.

Run the GlusterFS client or server inside gdb.

```
$ gdb ./glusterfs
(gdb) set args -f client.spec -N -l/path/to/log/file -LDEBUG /mnt/point
(gdb) run
```

Now when the process segfaults, you can get the backtrace by typing:

```
(gdb) bt
```

If the GlusterFS process has crashed and dumped a core file (you can find this in / if running as a daemon and in the current directory otherwise), you can do:

```
$ gdb /path/to/glusterfs /path/to/core.<pid>
```

and then get the backtrace.

If the GlusterFS server or client seems to be hung, then you can get the backtrace by attaching gdb to the process. First get the PID of the process (using ps), and then do:

```
$ gdb ./glusterfs <pid>
```

Press Ctrl-C to interrupt the process and then generate the backtrace.

7.4.5 Reproducing the bug

If the bug is reproducible, please include the steps necessary to do so. If the bug is not reproducible, send us the bug report anyway.

7.4.6 Other information

If you think it is relevant, send us also the version of FUSE you're using, the kernel version, platform.

Appendix A GNU Free Documentation Licence

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible.

You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled

"Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified

version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.0.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index