

# Global File Systems

## Is Red Hat's GFS worthy of the name?

Term Paper by

Douglas McClendon (2006/05/11)

University of Kansas, School of Electrical Engineering and Computer Science

[douglas@mcclendon.org](mailto:douglas@mcclendon.org) / <http://douglas.mcclendon.org/school/ku/eecs645/termpaper>

### 1) Introduction

#### 1.1) Abstract

An easy to use, simple to deploy and manage, high performance Global File System (**GFS**) is something of a holy grail for computer network users and administrators. This is no doubt why Red Hat Inc. chose the name "Global File System" (**RH-GFS**) for one of its key enterprise products. This paper presents an introduction to the broad topic of Global File Systems implemented via the Internet, and a brief evaluation of Red Hat's product bearing the same name.

#### 1.2) The Web as the First Popular GFS

From some perspectives, the World Wide Web, was the first GFS with sufficient performance and simplicity to generate truly widespread global adoption. However, just as computer hardware designers are never content with today's achievements, network software developers continue to strive for a GFS which is even better than the Web.

#### 1.3) Review: Traditional File Systems

At the most basic level, a file system is just a method for storing lots of information, and a protocol for identifying and retrieving individual pieces of that information. In the paper world, this is often done by secretaries with filing cabinets, providing a service to their employers simplifying data retrieval and storage. In the computer world, this is done by software Operating Systems (**OSs**) with digital storage such as electrical, magnetic, and optical media, providing a service to applications and users simplifying data retrieval and storage.

## 1.4) Global File Systems

A filing system becomes global, when the storage locations, and users requesting and retrieving data are distributed across the Earth. To achieve this, a means of transporting or copying the data via long distances must be used. In the paper world, a GFS could be implemented using national and international postal services to transport the requests for data to the physical location of the data, and then to transport the data to the physical location of the requesting user. In the computer world, the communication system required for a GFS can clearly be provided by the global Internet.

Thus the four most important components of a GFS are

- **storage:** the means of data storage
- **communication:** the means of data transportation
- **addressing:** the method of identification of individual pieces of data
- **protocol:** a well defined procedure for requesting and retrieving data in the context of a particular set of storage, communication, and addressing means.

## 1.5) The Web as a Global File System, and It's Limitations

Clearly the Web provides all of these things. Data are stored on disks and RAM in web servers. The Internet is used for communication. Universal Resource Locator (URLs) are used to address specific pieces of data (i.e. web pages, files). Finally the Hypertext Transfer Protocol (HTTP) binds all of these into a coherent useful system.

One aspect that the web lacks however, is the simplicity for users that the traditional computer OS filesystems interface provides. A traditional filesystem from a computer OS perspective is presented to the user as a single directory, rooted at a "Mount Point", e.g. /mnt/data in \*nix terminology, or rooted at a drive letter, e.g. D:\ in DOS/Windows terminology. In such a traditional filesystem, users can search for files, and manage them using command-line tools such as ls/dir, cp/copy, cd, find, or graphical tools which utilize the standard OS interfaces to such "mounted filesystems". It bears noting that the majority of pre-internet and non-networked applications utilize this standard OS interface for file access.

## **1.6) GFS as a "Field of Dreams" [1]**

While current mainstream graphical consumer OS interfaces such as Microsoft Windows do blur the line between the Web and traditional "mounted filesystems", there is still plenty of motivation to provide data distributed over the Internet via the more traditional filesystem interface. One way to view that motivation, is through the "Field Of Dreams"[1] metaphor. The "Field Of Dreams" metaphor is used to describe the aspect of network infrastructure advancement which is developed without specific motivating applications or functionality. Instead, this aspect of advancement is done for the purpose of providing a new more powerful infrastructure, on which it is expected a multitude of new ideas will be explored and ultimately implemented and widely used.

Presenting a global network of files to the application developer, through the existing default filesystem interface, will allow many existing applications to gain profound new functionality, with no, or only slight modifications. For instance, a media player application that already knows how to catalog and present a local library of music and video to the user, could instantly be able to present and catalog a globally shared library of music and video to the user, without changing the user interface at all. The potential to use existing code and applications against global, rather than local filesystems, is one of the key motivations driving the multitude of current research projects relating to GFSs.

## **1.7) Red Hat Inc's GFS Product**

Among the nearly dozens of current products and research projects that loosely fall under the scope of the GFS umbrella, one in particular from Red Hat Inc. stands out. Red Hat's product stands out for for two main reasons. First, their product is literally named "Red Hat's Global File System". Second, in the modern \*nix world, Red Hat still maintains a significant, if not dominant position in enterprise server deployments.

After reviewing the networking issues involved with GFSs, as well as the evolution of network filesystems in general, we will take a close look at the question of how Red Hat's product stands up in the world of GFSs. Finally, I will conclude with an outline of features one might wish to see in an ideal implementation of a GFS.

## 2) Networking Issues with Global/Networked/Distributed Filesystems

Global File Systems, being a rather general user of data transfer over IP networks, are sensitive to most of the traditional issues of IP. These being-

- **Latency**

While GFSs are less sensitive to latency, than say quasi-realtime voice or video communications, users and applications usually do want snappy, rather than sluggish response time. Caching is often used to improve latency. Caching can be done both at the server and client side, and for metadata as well as file data.

- **Bandwidth**

Again, GFSs are a middleware, and the users or applications will dictate how high a relative priority bandwidth is. But as always, more is always better. Bandwidth can be improved by decreasing GFS protocol overhead, as well as utilizing multiple network paths by distributing the data across many systems in the network.

- **Error Detection and Correction**

Dealing with data files, GFSs almost exclusively require faultless fidelity. All errors must be detected and corrected. This may be implemented in various ways, perhaps relying on TCP to provide reliable transport, or perhaps using custom code to deal with UDP's lack of reliability.

- **Security**

Different GFSs offer varying security features, such as-

- **Authentication**

Authentication is often provided to restrict access, perhaps at a high level, or perhaps with a finer granularity, relative to sets of files, or even individual files.

- **Confidentiality**

Confidentiality of transported data, via encryption, is a desirable feature, but seldom implemented due to complexity and performance reasons. Implementations layered on top of a VPN or IPsec can achieve this.

- **Other**

Other security features such as nonrepudiation, anonymity, and digital rights management (DRM) are interesting, but are not yet widely available.

- **Scalability**

Scalability is not exclusively a networking issue, but is usually heavily influenced by network limitations. The primary factors of scale with GFSs are

- number of simultaneous (and non-simultaneous) users/clients
- number of servers
- total size of filesystem, both raw size, and by number of files

- **Ease of Use**

Ease-of-Use is the last major issue to mention, and of these, the least directly influenced by network limitations. But ease of use may in fact be the most important issue impeding more widespread use and adoption of any particular GFS implementation. This issue falls into two main categories-

- Ease of Use for the client application and users
- Ease of Use for the administrators

The Web's initial claim to fame was a tremendous advance in the Ease-of-Use factor for users.

### **3) The Evolution of Network Filesystems**

#### **3.1) Databases as GFSs**

Before embarking on a brief overview of the evolution of network filesystems, it bears noting that networked databases do provide a very similar service. The key difference however is that databases provide their data in systems that are not based on the file and directory tree mechanism. Rather, one can view mounted network filesystems as a particular type of networked database. Many common collections of files, can possibly be better managed in a database context. For instance, email, image, and video libraries all might be very manageable in a database, rather than a filesystem context. Globally Distributed Databases, are certainly as important an area of research

as Globally Distributed File Systems. However, the simplicity and familiarity of the filesystem directory interface, is likely to entrench GFSs in the Internet's future, alongside Globally Distributed Databases.

### 3.2) Early Non-Distributed Networked(Global) Filesystems

While there have no doubt been many network filesystems over that last several decades, the most widespread have probably been these-

- **FTP and the Web**

As mentioned before, while not traditional mounted filesystems, both FTP and HTTP provide a means to read, and write files hosted on the storage of a server located elsewhere on the internet. Unlike the transparency of a mounted network filesystem, a GET or PUT command is required to download a file before reading, or upload a file after writing.

- **NFS**

“Sun's NFS protocol provides transparent remote access to shared file systems across networks.” [2] This open standard, is perhaps the most widely used network filesystem in the \*nix world. An NFS volume is hosted by a single server, and may be mounted by one or more clients, via a command something like-

```
mount -t nfs nfserver.domain.name:/volume_alpha /alpha
```

After issuing a command like the above on a client system, all the files located on the remote server with the Internet IP address of nfserver.domain.name, under the directory /volume\_alpha, would then be transparently visible to the client system under the directory /alpha

- **SMB (Windows Network File Sharing)**

SMB was the defacto filesharing protocol for Microsoft Windows systems. Operationally similar to NFS, a user could access a file, using a syntax like this-

```
\\nfserver.domain.name\alpha\somedir\file.txt
```

Where "alpha" would be the "share name" given to a directory such as c:\volume\_alpha

on the server. Additionally, a client system, for easier reference could "map a network drive", effectively making a shortcut from \\nfserver.domain.name\alpha to a drive letter like N:\, so that the file might be referenced as

N:\somedir\file.txt

### **3.3) RAID Analogies**

While these early networked file systems were and are popular and useful, they suffer the same performance and resiliency bottlenecks as a traditional filesystem housed on a single disk drive. That is, there is only one path to the data, and no redundancy in place to facilitate survival from a disk failure.

These two limitations have been surpassed in the local filesystem world via RAID. RAID based filesystems are stored on multiple disks with multiple channels connecting the host to those disks. If one can afford the extra storage space, and store multiple copies of data on multiple disks, then the data becomes resistant to the failure of any single disk, or a single channel. In addition, by allowing the host to read different parts of the same file from different disks, the multiple channels allow a greater access speed to the data.

Those two RAID methods, are perfectly analogous to how a Globally Distributed Filesystem gains increased performance, and is able to survive server failures. In the GFS case, instead of multiple disks, multiple servers are used. And instead of multiple host buses, multiple network channels are used. Now, if a single server host goes down, either due to fault or maintenance, the clients and their processes, applications, and users needn't be interrupted. As in the RAID case, when all the servers and network channels are functioning, they may be used in parallel to increase performance.

In general terms, nearly all IP networked file systems can be thought of as Global File Systems. However, for my own purposes I will restrict my definition of GFS to those IP networked filesystems which take advantage of RAID-like performance and redundancy features via distributing the filesystem data across multiple servers.

### **3.4) Some GFS Features**

Before listing some examples of existing GFS products and projects, here is a list

of some of the primary features which differentiate the multitude of alternatives available. Note that this list is an extension beyond differing approaches to the aforementioned networking issues.

- **Distribution**

How the filesystem data is distributed across multiple servers in a RAID-like manner. Implementations of this feature can facilitate more robust fault tolerance, and improved performance.

- **Disconnected Operation**

How clients behave when a server or network connection is temporarily down. Traditionally, NFS has had the most painful response in this regard, hanging client processes.

- **Caching**

Whether filesystem caching is done, either on servers or clients. Client caching, in addition to improving performance, can facilitate some levels of disconnected operation.

- **Locking**

When multiple clients are trying to write to the same file, the GFS must manage this in a way that does not lead to confusion. In some cases locking is managed by extra dedicated servers.

- **POSIX Conformity**

The POSIX filesystem standards provide a well known interface to application writers. If the full set of POSIX features are not implemented, some existing applications may not work properly with this type of filesystem.

- **Storage Media**

While data is most often stored on hard disk drives, it is also possible to store data in RAM, flash memory, optical media, and tape drives.

- **Metadata Management**

Metadata is usually stored near file data on disk. Some GFSs store and replicate metadata separately for improved performance. Extra dedicated servers may help

manage metadata.

- **Scalability**

In addition to network scalability of numbers of servers, clients, and bandwidth usage, GFSs may also have scalability limits relating to overall filesystem size, both with respect to total data size, and number of files.

### **3.5) The Vast Array of GFS Current Research Projects and Products**

There is absolutely no shortage of GFS options out there. This is partly because no one product or project has yet been able to satisfy a majority of users to the same degree that HTTP has become ubiquitous. Here is a list of some of the more widely known GFS products and projects, with some of their key differentiating features. Links to further information can be found at the end of this paper after the references.

- **Andrew Filesystem (afs)**

Developed at Carnegie-Mellon University and implemented by IBM, AFS heavily utilizes client caching as well as supporting security features with KERBEROS authentication.

- **Google Filesystem (googlefs/gfs)**

The proprietary in house Google Filesystem is used for distributed applications. It scales to hundred's of terabytes, stored onover a thousand servers, accessed by over a hundred clients. [3]

- **SGI's Clustered XFS (cxfs)**

CXFS is a distributed version of SGI's XFS, meant for SANs. With CXFS, metadata is managed seperately from data. CXFS is POSIX compliant.

- **Parallel Virtual Filesystem Version 2 (pvfs2)**

PVFS2 is funded by NASA, NSF, and DOE. It features distributed metadata management, and striping of file data across multiple nodes.

- **LUSTRE**

LUSTRE is offered by HP and Cray, and targets scalability, i.e. systems with 10,000 nodes, petabytes of storage, and 100GB/s bandwidth. Lustre supports distributed

metadata as well as cryptographic security, via their stackable driver framework.

- **Microsoft Distributed File System (ms-dfs)**

MS-DFS is the official Microsoft solution at the moment. They provide a namespace providing a unified view of files located on multiple servers. This is not a GFS per-se, but rather a hack, which is also available in the unix world known as a "union filesystem" or an "overlay filesystem", which basically presents the user a single view of multiple filesystems. The main problem with union filesystems is that while reads are easy, figuring out which layer of the overlay to write to, is not automatically managed.

- **Radiant Systems PeerFS (peerfs)**

Radiant Data provides a proprietary filesystem for linux, that uses peer to peer replication of data, which is certainly very interesting, and is a move towards the peer-to-peer filesharing functionality that I would like to see integrated into more GFSs.

- **Oracle Cluster Filesystem (ocfs/ocfs2)**

Oracle's entry in the field is targeted at their own Real Application Cluster platform. Though it is also POSIX compliant and a fully functionality general purpose filesystem with an emphasis on high performance via caching. OCFS2 utilizes the JBD interface for journaling, to increase performance.

- **IBM's General Parallel Filesystem (gpfs)**

IBM's GFS has held world records in performance, and offers strong fault tolerance in the event of individual servers failing. GPFS, like Red Hat's GFS however, seems to be targeted at tightly coupled clusters on a LAN or local SAN.

- **NCSA's Teragrid's GPFS-WAN**

NCSA's Teragrid is a distributed computer for scientific research. The Global Parallel Filesystem - Wide Area Network, is actually another GFS that is more of a local cluster filesystem, but with an emphasis on a very large number of globally distributed clients.

- **FUSE "Hacks" : Linux's Filesystem in Userspace**

Even with the above only being a sampling of GFS products, another recent avenue for research has been the advent of a user-space filesystem meta-driver for linux.

Traditionally, filesystems were kernel level code, which developers were very picky about letting researchers add their code to. With the advent of FUSE, a vast multitude of quick hack filesystems have emerged, some with very interesting potential. Among the noteworthy are-

- SSHFS - a filesystem utilizing a generic ssh server for storage
- GMailFS - a filesystem utilizing the google cluster's GMail service for storage
- Clustered Ordinary Raid Network Filesystem - a filesystem taking the RAID analogy to a literal level
- GFarmFS - a filesystem filling the cracks with another grid filesystem which was not in itself fully unix/posix compliant.
- FuseFTP - a filesystem exposing ftp served data as a traditional mounted filesystem.
- Grifi - a filesystem like fuseftp, but exposing the data of multiple ftp servers.
- Bit Torrent FS - perhaps the most intriguing FUSE hack, is a filesystem that exposes a .torrent as a mounted filesystem. Unfortunately, this implementation is only about reading files, not writing them, which I would find vastly more appealing.

#### **4) Is Red Hat's GFS worthy of the name?**

##### **4.1) Expectations for Red Hat's Global File System**

Returning to Red Hat's GFS product, the critical question is this- Will RH-GFS work in an environment of globally distributed server storage nodes? Or is it, like NFS and other GFSs, restricted to merely the case of globally distributed clients, or worse yet, restricted to the case of a tightly clustered system on a LAN or geographically small SAN? By using the name "Global File System", expectations have been set to see a filesystem that performs well when all of the server and client nodes are widely distributed across the global Internet. One would be let down if in fact this "Global" Filesystem was only ever meant to be hosted on a set of tightly coupled servers in a cluster within a single room of a single building.

## **4.2) RH-GFS Is Really Targeted at Tightly Clustered Systems**

A thorough investigation of the RH-GFS documentation suggests that the word "Global" in the name is more of a marketing reference, or a reference to clients, than it is to the appealing feature of data storage servers being globally distributed. In fact, in the user manual, of the 3 pictorial example scenarios, all involve the servers living together on a tightly coupled LAN or SAN. [4]

While SANs can be deployed over WANs, it is not the typical assumption made for most SAN technology. It would have been nice to be able to use RH-GFS with standard PC hardware over a WAN, without having to purchase new SAN hardware. It seems the only option to use RH-GFS over a WAN is to represent the WAN to GFS as a virtual LAN, using VPN technology such as IPsec, PPP over SSH, or bridging.

## **4.3) RH-GFS Limitations**

The RH-GFS documentation also exposes two fairly important limitations. First, the limit of 400 server nodes[4]. While this is fine for many applications, especially in a tightly coupled cluster environment, one can easily see the usefulness of a filesystem spanning thousands of globally distributed server nodes. Second, GFS lacks any network security features, such as authentication, or encryption. Therefore if needed these must be managed at the network infrastructure layer, using other tools.

## **4.4) Potential RH-GFS WAN Experiments**

It had been the intent of this paper to measure the performance of RH-GFS in LAN and WAN environments, even if VPN or other additional infrastructure was required. Unfortunately, time, and the complexity of such additional configuration, namely the "complex animal" that is IPsec [5], in addition to the complexity of RH-GFS itself, did not permit an experimental investigation. There is no reason to believe that such configurations will not work to some level of satisfaction. However, the complexity noted, sheds light on significant room for improvement for future implementations of similar GFSs.

## 5) The Quest Continues...

In conclusion, Red Hat's Global File System does not yet measure up to what I was searching for. The main features I would like to see that it lacks are-

- Ease of Use: Dramatic Simplification of Deployment and Management
- Security: Built in Network Authentication, and Encryption at least
- Peer To Peer Implementation: Current P2P networks provide strong resiliency and scalability, but lack a traditional filesystem interface.
- DRM: Digital Rights Management is an important feature to allow copyrighted material to be hosted on a publicly accessible global file system. One potential DRM feature which intrigues me but has yet to be explored, is to limit the number of accessors of a file. With such a feature, users could contribute their media data to a global library, in the same legal manner that they can currently contribute their physical media to a physical library.

I.e., what I'm truly looking for is something like-

Mount -t truly\_global\_filesystem internet:congressional\_video\_archive /mnt/videos  
(or mapping \\internet\congressional\_video\_archive to V:\)

And then being able to access video files via

ls /mnt/videos/pbs/2005/November/\*/\*Nova\*  
(or dir V:\pbs\2005\November\\*/\*Nova\*)

There is no doubt it'll happen soon enough.

## References

- [1] James P. G. Sterbenz, and Joseph D. Touch. High-Speed Networking. Page 21, John Wiley & Sons, New York, 2001.
- [2] B. Callaghan, B. Pawlowski, and P. Staubach. NFS Version 3 Protocol Specification. RFC1813. Page 3, Sun Microsystems, June 1995
- [3] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. Appearing in ACM SOSP '03, October 19-22, 2003. Page 2. Bolton Landing, New York, USA
- [4] Red Hat Documentation Team. The Red Hat GFS 6.1 Administrator's Guide. Pages 18-20. Raleigh, NC, 2005
- [5] James F. Kurose and Keith W. Ross. Computer Networking. Page 712. Addison Wesley, Boston, MA, 2005

## Filesystem Links

Andrew File System

<http://www.openafs.org>

Google File System

<http://labs.google.com/papers/gfs.html>

Clustered XFS

[http://www.sgi.com/products/storage/tech/file\\_systems.html](http://www.sgi.com/products/storage/tech/file_systems.html)

PVFS2

<http://www.pvfs.org/pvfs2/>

LUSTRE

<http://www.lustre.org>

DFS

<http://www.microsoft.com/windowsserver2003/techinfo/overview/dfs.mspx>

PeerFS

<http://www.radiantdata.com/>

OCFS2

<http://oss.oracle.com/projects/ocfs2/>

GPFS

<http://www-03.ibm.com/servers/eserver/clusters/software/gpfs.html>

Teragrid GFS-WAN

<http://www.teragrid.org>

FUSE based filesystems

<http://fuse.sourceforge.net/>