File Systems

Justin Groce
Systems Programming
jbgroce21@tnech.edu

ABSTRACT

File Systems (FS) are fundamental to modern computing. A FS is the way a computer or network stores files. This involves, among other things, how a file is stored, how a file is altered, and how a file is retrieved. Obviously, a single computer would have a different FS than a network of multiple computers. While it would likely have a different FS implementation, the concept remains very similar regardless of scale. What happens if the machine the files are stored on is unexpectedly shutdown? Who has permission to access or change files? These are some of the questions that must be answered for any FS.

There are many different implementations of FS. These implementations differ in the delivery methods of getting the file from its location in physical memory to its destination. The major implementations of FS discussed in this paper are the Network File System (NFS), the Andrew File System (AFS), the Google File System (GFS), and ZFS. Core procedures and conventions will also be discussed. For simplicity's sake, distributed or network FS will be focused on.

Keywords

FS, File System, Network, DFS, AFS, GFS, ZFS.

1. How does it work

A file system (FS) is a blueprint of how to organize files. Initially, FS relied heavily on specific system calls straight from FS calls [5]. This approach hinders the effort to expand upon the system with more FS.

Abstraction was used with FS by introducing the virtual node (vnode) to the FS. A vnode provides distance from the operation system from the FS. This abstraction allows for operating systems to support many different file systems. Linux 2.6, for example, can support over 30 FS [5].

When implementing vnodes, it is possible to layer multiple FS on top of each other. This is achieved by one vnode calling another until the operating system is reached to do the desired operation. This layered approach also give the advantage of adding functionality in the middle layers without affecting the overall outcome of an operation. Designing vnodes to do extra operations is much simpler than trying to change the operating system in some nontrivial way [5].

A commonly supported FS is the Network File System (NFS). NFS like other FS provides the possibility to share files and directories with multiple clients over a network. This is beneficial for users because local machines use less of their own disk space when they store commonly used files on a central machine that is accessible by those that use the file. That way a user can even move from machine to machine and still have all the files they need. The user does not need a directory on each machine with copies of their files in multiple locations.

NFS operates on a client server model. Thus, a client connects to a server and requests data. Then, the server responds to the client with the requested data.

Another specific FS is the Andrew File System (AFS). AFS is a popular FS that was designed at Carnegie Mellon University. The Distributed File System (DFS) is simply one version of AFS [3].

In terms of architecture, AFS is similar to NFS in that they both provide a method of connecting client and server machines in a shared system. However, AFS is specially designed to provide reliable file services on a large scale. This is accomplished with cells. In AFS, a cell is a group of client systems and file servers. A cell is managed by a single authority [3].

AFS cells have a privacy policy as well. A user can access files easily within the cell, but a user much have permission to access the files in another cell. The purpose of having cells is to take the operating system out of the equation. Much like vnodes, cells can be in between the user and the operation system [3].

AFS is set up to use the client server paradigm. That is to say that a client makes a request to the server for a file service and the server subsequently responds with the information. The server also keeps track of the structure of the FS by monitoring the files, the status of the FS, and verification of clients [3].

AFS has some features that NFS does not have. There features include using file names that are not tied directly to physical addresses, caching in the clients to lower the load of the network, and data encryption for security purposes [3].

One of the biggest FS is one of the most secretive (kind of). That FS is the Google File System (GFS). Google is a huge company, and GFS is used by many people. This being said, the GFS does not use fancy, state of the art computers in its implementation. While there is a lot that non-Google employees don't know, it is known that GFS is implemented by inexpensive machines using the Linux operating system. So the computers used in GFS are not the intriguing part of the FS. The way Google uses these machines is intriguing [4].

GFS is organized into networks of machines called clusters. Clusters can have hundreds of machines in them. Clusters could even have thousands of machines in them. The clusters are organized into three different classifications. Those classifications are client, master server and chunkserver [4].

A client with respect to GFS refers to a machine or application that makes a request. A client can request to retrieve a file, to manipulate an existing file, or to create a new file [4].

A master server with respect to GFS is the server that coordinates the cluster. The master logs all the activities that happen in the cluster. The purpose of keeping the log is to minimize downtime. However, if a master server crashes, then there is another server ready to take its place. It is important to note that there is only one master server active in a cluster at a time. The master also keeps track of the location of the chunks throughout the cluster. A chunk is a portion of a file. Because GFS typically works with large files, the files are broken up into chunks. In order to do all of the duties that a master server must do, the master server doesn’t actually handle file data. The master server sends and receives small messages and leaves to hard work to the chunkservers. This keeps network traffic to a minimum [4].

A chunkserver with respect to GFS is the server that does all the real work. Chunkservers store all the chunks for all the files in the cluster. They actually store copies of the chunks. At one time, there are about three copies of each chunk spread out to different chunkservers. This is to prevent loss of data in the event of a chunkserver going offline for any reason [4].

To discuss ZFS, this paper will focus on the Oracle Solaris ZFS. The main concept in ZFS is the idea of virtual memory. This idea removes the FS from keeping data in physical storage. This allows the machines used in the implementation to be used more efficiently. With virtual memory, the actually physical storage device can be added to the virtual memory pool with no interruption of service [2].

ZFS is constantly checking its data to make sure it is up to date and correct. If an error is found, there are procedures in place that correct the error behind the scenes, again without interrupting service [2].

Another method of error checking is journaling. Almost any implementation of a FS will use some sort of journaling. Journaling is simply keeping a record of events that happen in the FS. In the event of an error or complete system failure, the system can be brought back to a working state by replaying the journal [1].

2. Conclusion

A File System (FS) is essential for any reasonable network of computers. A FS allows the sharing of data between machines in an efficient way. The possible implementations vary differently and as seen in this paper, there are many. Depending on the size of the network and the amount of data being transmitted, one could choose from multiple different implementations.

Most implementations of FS are open source to the public at least on an abstract basis. This allows anyone to implement a FS similar to the mainstream one. To pick which FS is right for a particular network, one must simply compare the resources in the network and the desired speed of data transmission to the descriptions of an existing FS. Once a FS is chosen, it is simple to implement it by following the steps laid out in the description of the FS.

3. REFERENCES

[1] Frost, Christopher and Mike Mammarella, Eddie Kohler, Andrew de los Reyes, Shant Hovsepian, Andrew Matsuoka, Lei Zhang "Generalized File System Dependencies" (Nov 2010) http://delivery.acm.org.ezproxy.tntech.edu/10.1145/1300000/1294291/p307-frost.pdf?key1=1294291&key2=4183940921&coll=DL&dl=ACM&CFID=112200312&CFTOKEN=93840110

[2] "ORACLE SOLARIS ZFS" (Nov 2010) http://www.oracle.com/us/products/servers-storage/solaris/034779.pdf

[3] Sheldon, Tom "AFS (Andrew File System)" (Nov 2010) http://www.linktionary.com/a/afs.html

[4] Strickland, Jonathan "How the Google File System Works" (Nov 2010) http://communication.howstuffworks.com/google-file-system.htm

[5] Zadok, Erez and Rakesh Iyer, Nikolai Joukov, Gopalan Sivathan, Charles P. Wright "On Incremental File System Development" (Nov 2010) http://delivery.acm.org.ezproxy.tntech.edu/10.1145/1150000/1149979/p161-zadok.pdf?key1=1149979&key2=5333940921&coll=DL&dl=ACM&CFID=112200312&CFTOKEN=93840110