Unix File Systems

Ahmed Almohsin
amalmohsin21@tntech.edu

ABSTRACT

In this paper we will discuss about Unix file systems, such as NFS, AFS, ZFS, GFS.

General Terms

Design, version, hierarchical. structure, interface.

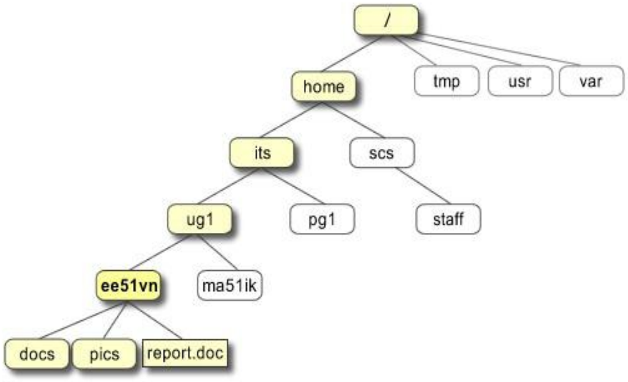
Keywords

NFS, AFS, ZFS, GFS.

1. INTRODUCTION

UNIX keeps track of directories and files using a file system. Once you log in to your UNIX account, you are automatically placed in your "home" directory and so home directory becomes your "present working directory" when you log in. In your home directory, you can create files and directories and subdirectories.

UNIX file system has an upside down tree structure. At the very top of the tree is the root directory, named "/". It is maintained by UNIX system administrator. Under the root directory, subdirectories organize the files and its subdirectories on the system.



There are five types of UNIX files: Regular, Directories, Symbolic Links, IPC Endpoints, and Device Files. Regular files are the most common and contain basic data. Directories contain files and other directories. Symbolic links is an alias for another directory or file name. IPC (Inter-Process Communications) Endpoints allow processes to communicate with each other while running on the same machine. Device files are use to access hardware. They are typically found in the “/dev” directory.

2. NFS

Network File System (NFS) version 3 was introduced by Sun Microsystems in the mid 1990s. The NFS protocol is designed to be independent from network architecture and transport protocol. This independence is achieved through the use of Remote Procedure Call (RPC). In the NFS version 3 protocols, servers do not need to maintain the state of its clients in order to function correctly and so it is called as stateless server implementation. Stateless servers have an advantage in the event of a crash over state-ful servers. In stateless servers, a client only need to retry a request until the server responds; the client does not need to know that the server has crashed.

NFS version 3 could be hacked easily if it is not on a secure dedicated network, because security is enforced at only client level instead of server level. Like HTTP and FTP it is also a stateless protocol, so it is lack of performance since it must assert its present state with all operations, unlike version 4 it does not define Open and Close operations. It only supports Read and Write operations. File locking was added later but still it could be hacked on a network.

NFS version 4 features improved access and better performance over the Internet, strong security with negotiation built into the protocol, good cross-platform interoperability and was designed for protocol extensions. It has ability to migrate or replicate file systems by the use of a special file attributes i.e. file system locations attribute which provides a method for the client to probe the server about the location of a file system.

During the migration of a file system, the client will receive an error when operating on the file system and it can then query as to the new file system location. Similarly replication is done; the client is capable to query the server for the multiple available locations of a particular file system. From this information, the client can use its own policies to access the appropriate file system location. Version 4 also introduces OPEN and CLOSE operations. The OPEN operation provides a single point where file lookup, creation, and share semantics can be combined. The CLOSE operation provides for the release of the state accumulated by OPEN.

3. AFS

In 1980s Carnegie Mellon University researchers introduced. The main goal of AFS was to design a distributed file system so that server can support as many clients as possible. The first version of AFS was called ITC distributed file system(S+85). The AFS is a distributed network file system which enables access to files from any machine located at far distances like they are stored locally.

AFS is composed of cells and each cell represents an independent portion of the file space. Cells are connected to form UNIX file system under root “/afs” directory. An AFS cell is a collection of servers grouped together administratively which presents a single, cohesive file system. Some advantages of using AFS are caching facility, security features, simplicity of addressing, scalability, single system image, replicated AFS volume, communications protocol, and ease of networking. AFS volume(s) which constitute

"/home" can be simply moved between the servers. It could be achieved without any intervention to the users' work, while users are using files actively in "/home" directory.

AFS client machine runs a Cache Manager for caching which maintains information about the identities of the users who are logged into the machine, and it finds and requests data on their behalf, and keeps chunks of retrieved files on local disk. As a result of this as soon as a remote file is accessed, a portion of that file gets copied to local disk. Thus subsequent accesses are as fast as local disk, and considerably faster than a cold read (across the network). Caching locally reduces the amount of network traffic, which improves performance when cold reading across the network.

AFS uses Kerberos network authenticating protocol to authenticate users which improves security by not passing plaintext passwords on the network, and by making encrypted passwords invisible. AFS also uses Access Control Lists (ACLs) to enable users to restrict access to their own directories. User does not need to know which fileserver has the requested file; the user only needs to know the pathname of a file and the name of the AFS cell where the file is stored. When a user authenticates using the "klog" command he/she will be prompted for a password. If the password is accepted Kerberos Authentication Server (KAS) will provide the user with an encrypted token, which will expire after the limited lifetime. The "log" and "unlog" commands are used to make authentication more convenient. Log re-authenticates a user if the tokens have timed out, and unlog discards the tokens.

4. ZFS

A third UNIX file system is the ZFS file system. ZFS was designed by Sun Microsystems. It is mainly used by Sun's operating system, Solaris, but can be implemented on many operating systems. ZFS allows for end to end data integrity and pooled storage. ZFS is a 128-bit file system. This allows there to be no limit to how much can be on the system including files, snapshots, links, or directories.

Pooled storage is a model that doesn't use the concept of volumes. Many of the file systems can draw from one storage pool and only take up the space that is needed to complete its task. The combined I/O bandwidth of every device that is in the pool is available to every file system at all times.

A big part of ZFS is data integrity. Every block of storage is checksummed to prevent data corruption. Using mirror or a RAID configuration, the data can actually be self-healing. If a copy becomes damaged, ZFS will repair this copy with another one. ZFS also introduces a new replication model called RAID-Z. RAID-Z uses variable stripes. ZFS also offers disk scrubbing that traverses the entire storage pool and validates the data with a 256-bit checksum and repairs where needed. It can do disk scrubbing while the system is live. ZFS also allows the use of unlimited snapshots. A snapshot is a read-only, point-in-time copy of a file system. Any of the snapshots can be a full backup to the file system while any number of pairs of snapshots can be made into an incremental backup.

In ZFS device driver exports a block device to the Storage Pool Allocator (SPA). The SPA handles block and I/O allocation; exports virtually addressed, explicitly allocated freed blocks to the Data Management Unit (DMU). The DMU turns the virtually addressed blocks from the SPA into a transactional object interface for the ZFS POSIX Layer (ZPL). Finally, the ZPL implements a POSIX file system on top of DMU objects, and exports vnode operations to the system call layer.

The SPA allocates blocks from all the devices in a storage pool. One system can have multiple storage pools, although most systems will only need one pool. Unlike a volume manager, the SPA does not present itself as a logical block device. Instead, it presents itself as an interface to allocate and free virtually addressed blocks - basically, malloc() and free() for disk space. We call the virtual addresses of disk blocks data virtual addresses (DVAs). Using virtually addressed blocks makes it easy to implement several of our design principles. It allows dynamic addition and removal of devices from the storage pool without interrupting service.

5. GFS

Google File System was invented by Google Inc to meet the rapidly growing demand of their data processing needs and their goal was to build a huge storage network by using inexpensive hardware and system must constantly monitor itself on a routine basis to detect, tolerate, and recover promptly from component failures. Their goal was to include constant monitoring, error detection, fault tolerance, and automatic recovery features as an integral part of the file system. After deletion of a file, GFS does not immediately reclaim the storage. It does reclaiming during regular garbage collection at both the file and chunk levels which makes the system simple and reliable.

GFS provides a common file system interface, but does not implement a standard API such as POSIX. Files are organized hierarchically in directories and are identified by pathnames. It also supports the usual operations to create, delete, open, close, read, and write files. GFS supports creating copy of a file or a directory tree at low cost. It also supports multiple clients to append data to the same file along with guaranteeing atomicity of each individual client's append. It is useful for implementing multi-way merge results and queues that many clients can simultaneously append, without additional locking.

A GFS cluster consists of a single master and multiple chunk servers and can be accessed by multiple clients and these are typically Linux machines running user level server process. Also it is easy to run both chunk server and client on the same machine as long as machine configuration allows. Files are divided into fixed-size chunks. Each chunk is identified by an immutable and globally unique 64 bit which is much larger than typical file system block sizes chunk handle assigned by the master at the time of chunk creation. Chunk servers store chunks on local disks as Linux files and read or write chunk data specified by a chunk handle and byte range. For reliability, chunks are replicated on multiple chunk servers.

The master stores three types of metadata. The file and chunk name spaces, the mapping from files to chunks and the location of replica of each chunk. Name spaces and file-to-chunk mapping are kept persistent by logging mutations to an operation log and is stored on the master's local disk and replicated on remote machines.

6. ACKNOWLEDGMENTS

My thanks to ACM, University of California-Santa Cruz, and Carnegie Mellon University for sharing their resources.

7. REFERENCES

[1] Bonwick, Jeff, Matt Ahrens, Val Henson, Mark Maybee, and Mark Shellenbaum. "The Zettabyte File System." n. pag. Web. 3 Nov 2010.
http://users.soe.ucsc.edu/~sbrandt/221/zfs_overview.pdf

[2] "File Systems Support." FreeBSD Handbook. FreeBSD, Web. 3 Nov 2010.
http://www.freebsd.org/doc/en/books/handbook/filesystems-zfs.html

[3] Howard, John. "An Overview Of Andrew File System." n. pag. Web. 2 Nov 2010.
http://reports-archive.adm.cs.cmu.edu/anon/itc/CMU-ITC-062.pdf

[4] "NFS: Network File System." Sun Microsystems (1994): n. pag. Web. 4 Nov 2010.
http://www.connectathon.org/nfsv3.pdf

[5] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "The Google File System." Web. 8 Nov 2010.
http://static.googleusercontent.com/external_content/untrusted_dlcp/labs.google.com/en/us/papers/gfs-sosp2003.pdf