Distributed file system

A Distributed File System is an application based on client/server relationship. This application allows clients to access and process objects stored on the server, as if they were stored on their own local hard disk. Whenever a user requests for an object, the server send s a copy of that particular object. The user computer caches the object and sends it back to the server. DFS organizes file and directory services of standalone servers into a global directory. In this way, all servers are interconnected to each other and all files can become available to end-user, truly showing distributed file system. Server has a mechanism installed, whereby many clients may access same data simultaneously. This mechanism also updates files so that the following clients receive most updated version of files and also reduces data conflicts. To protect against data access failures, DFS uses file or database replication. Some of the examples of DFS include:

* Novell Netware

* Microsoft’s Distributed File System

* IBM/Transarc’s Distributed File System

* NFS (we will discuss this example later)

Distributed File System has many common concepts. We will now discuss some of the concepts of Distributed File System. They include Naming and Transparency, File Replication, Remote File Access, Caching, Fault Tolerance, Security and many more. We will briefly go through some of the main common concepts of Distributed File System before embarking on Network File System. Concepts are as follows:

* File Replication

File replication is a useful idleness for improving accessibility of files on different machines. This feature also benefits performance as well. The basic procedure of file replication is that various replications of the same file should sit on independent machines. In this way, one replica is not affected by other replicas sitting on other computers. You can, however, hide the details of replication from users. In other words, they won't know the replication details and it is also advantageous. Naming scheme is responsible to map a replicated file name to a replica. In terms of existence of replicas, they should be invisible at higher level and they should be distinguished from one another by different names. The only problem with file replication is their update.

* Naming and Transparency

A common concept of Distributed File System is Naming and Transparency. Mapping between logical and physical objects is known as naming. We as users refer to the files or object as a textual name and see the files or objects as logical but the system sees the same file as physical blocks of data. Mapping actually provides users with all the information about the whereabouts of a file on the disk. On the other side, a Distributed File System in a transparent mode also tells the whereabouts of a file on the network. There are two categories to name mapping in a Distributed File System. They are Location Transparency and Location Independence.

– Location Transparency: This is whereby the name file's location should not be revealed. I. e. the file name doesn't give any physical storage location of the file.

– Location Independence: This is whereby the file name doesn't need to be changed when the actual location of the file changes in the physical storage location.

Both these categories are related to naming as files have different names at different levels.

* Remote File Access

In a Distributed File System, users can access to remote files by requesting the server for that particular file. This is achieved when the server storing the file is located by the naming scheme. A mechanism called remote-service sends requests to the server and the server replies back by sending the results after processing the requests. The user sends request to the server using the Remote Procedure Call (this will be discussed later).

* Caching

Caching is a very simple concept in Distributed File System. It any of the data is not cached, then a copy of that data is transported from the server to the user. Once the data is on the user computer, accesses are performed to that data. Any recently accessed disk blocks in the cache is preserved so that repeated access does not require any further network to cache; however, there is no direct connection between accesses and traffic to server. Files are seen as one copy sitting on the server but parts of the file

are distributed in different caches. Whenever the cached file is modified or updated, the changes are sent to the main copy of the file to keep it updated.

Many other common concepts of Distributed File System exist. We will now have a look at an example of Distributed File System. It is known as Network File System.

NETWORK FILE SYSTEM

The Network File System is an example of Distributed File System. The Network File System is a client/server application that allows a user to access, view, store and update files on a remote computer as if they were on the user's own computer. In NFS, a client needs to have a NFS client and the corresponding server needs to have a NFS server. Thus, it allows transparent access to files. The main means of communication is done through a communicating protocol known as TCP/IP. Transmission Control Protocol / Internet Protocol provide a reliable communication through establishing a definite connection between local client and host server before transmission of data.

Originally, NFS was developed by Sun Microsystems and it has been nominated as a reliable file server. With the help of NFS, users can mount a part of a file or the file as a whole on top of your local disks. Diagram below shows a typical NFS server sharing with client and server.

The z/OSTM Network File System

Source: http://www-1. ibm. com/servers/eserver/zseries/zos/nfs/index. html

NFS DESIGN AND ARCHITECTURE

As mentioned earlier, NFS aims to provide users with high performance, faster and transparent access to file servers to a vast majority of communities spreaded geographically. Below is a list of some of the most important features and design principles of NFS.

Transparency: This is one of the main features of NFS. This means that users or applications can access files stored on a remote location as if they were local. Users do not come to know whether files being accessed by them are local or on a remote location.

Fast Recovery from Failure: Services are restored very quickly as NFS is designed to overcome quickly from situations such as system failures and network problems. This feature benefits users as they encounter very minimal disruption of services.

Portability: NFS is portable. This means that it is independent from machine and operating system. It can easily be ported onto several hardware and operating systems platforms including mainframes.

Network Protocol Independence: This is also a very important feature of NFS. It basically means that NFS is flexible to run on various transport protocols instead of residing or sticking to one know protocol. This feature also allows NFS to make maximum use of existing protocols and also open doors for new protocols in the future.

Performance: NFS aims to utilize the highest performance standards so that users are able to access remote files just like they access local files. Performance is a key feature of NFS.

Security: NFS supports multiple security mechanisms. It gives users or administrators to choose from various available security mechanisms. They have the flexibility to adopt security mechanisms that suits their environment. Again, this feature also opens doors to new security mechanisms that may be available in the near future.

All these features reduce costs as they share resources across the global enterprise.

NFS PROTOCOL MECHANISM

NFS depends on two different protocols as a medium for communication between sender and receiver. They are known as RPC and XDR. Below is a description of these two protocols.

Remote Procedure Call (RPC)

NFS uses RPC to communicate between client and server. It is a session layer protocol. It implements a client/server link on the hosts that use it. RPC allows hosts to make calls as if they were local but actually they are on remote hosts.

External Data Representation (XDR)

XDR provides a facility to translate data between diverse computers and operating systems. It is a presentation layer protocol. XDR is used by RPC to host a reliable data exchange between clients and severs.

ACHIEVING TRANSPARENCY WITH PROXY PATTERNS

NFS transparency is achieved through RPC and XDR protocols. As mentioned earlier, these protocols are used by NFS to communicate between clients and servers. Below is a diagram showing the client/server relationship.

From the figure above, we can see that the client and the server are communicating with each other using stubs. We can also see that client is accompanied with client stub and server is accompanied with server stub. The client just requests a service through service request and after that only the client and the server interact circuitously with each other. The client and the server stubs use the XDR protocol to make a connection and communicate by utilizing the XDR functions. The figure below gives even a better image of the whole process.

The interaction between the stubs is clearly shown in the above diagram. NFS uses proxy pattern which shows the remote objects as local objects.

THE FOUR MAIN SERVICES OR PROTOCOLS SUPPORTING NFS

The four main services or protocols driving NFS are nfs, mountd, nsm and nlm. All these protocols are explained below in detail.

Nfs

This protocol resides at the base of NFS and it allows manages files, searching, reading and writing, authentication and file statistics.

Mountd

This protocol is responsible for mounting exported files systems so that they can be accessed with nfs. Requests such as mount and unmount are received by servers. The server keeps all the information about the exported file systems.

Nsm – Network Status Monitor

The main job of this protocol is to monitor network status. It monitors nodes to find out the state of a machine. It also informs functions such as reboot and restart.

Nlm – Network Lock Manager

This protocol manages a lock system whereby several clients cannot modify data at the same time. It knows exactly which files are being used so that it can implement a lock on them.

THE NFS ARCHITECTURE

The NFS architecture provides portability using a layers pattern. There are three major layers in NFS. They are UNIX file-system interface, Virtual file system and NFS service layer. The first layer is based on read, write, open and close calls, and file descriptors. The second layer is divided into two important functions.

– First function

By defining a clean VFS interface, the first function separates file-system generic operations from their implementation. Many implementations might co-exist on the same machine, thus, allowing transparent access to different media types mounted locally.

– Second function

The Virtual File System is based on vnode, which contains numerical designator for a particular file that is unique to the network and the kernel maintains this vnode structure for each node.

In this way, Virtual File System distinguishes files that are local from files that are remote. Furthermore, all local files are distinguished depending to their file-system types.

The third and the last layer implement the NFS protocol.

The figure below shows a layered architecture in NFS.

A NFS Layered Architecture

Source: http://www. media. kyoto-u. ac. jp/edu/lec/jnakamu/lecture/y98/miy98/part2/p871bp4. htm

EXPORTING AND MOUNTING IN NFS

Servers make their file systems sharable through a process called exporting and clients gain access to these exported files by adding them onto their local disks through mount process.

EXPORTING

Exporting is a process whereby the files requested by the user are actually sent to the user. Files are then mounted onto user's local hard drive. Server merely processes the user requests and return results to the user. There are some rules, though, for exporting a file in NFS. They are as follows:

– The whole file or part of the file can be exported. Directory /home can be executed as well as directory /home/users.

– If a subdirectory of a exported filesystem is exported then that subdirectory has to sit on another device.

– Parent File System can be exported only if subdirectory is exported and sitting on another device.

– Exporting can be done with local file systems only.

Once the exporting is completed, mounting takes over to complete rest of the transmission.

MOUNTING

As mentioned above, clients can have access to the exported files or objects only after they have been mounted onto their local disk " tree". Only after mounting, files can be accessed. There are three types of Mounts in a mounting process. They are predefined, explicit and automatic.

Predefined mounts are specified in /etc/filesystems file. Each and every entry in file holds characteristics of a mount process. Any entry such as host name,

local and remote path, etc… are defined in the entry. Predefined mount is mainly or specifically used when portions of mounts are required for operation.

Explicit mounts provides user needs at the root level. Explicit mounting is required when there is a need for unintentional mounting. Apart from that, this mounting can also be used for special tasks.

Automatic mounts mount objects according to the request. Whenever a user or program tries to access a directory which is not mounted, then it uses the AutoFS command to seize the request and arrange mounting of that request.

The SCO Network File System (NFS) software allows administrators to mount directories located on remote hosts across the network and then treat those directories as if they were local.

You can administer both local filesystems and locally mounted remote filesystems using the Filesystem Manager.

Below is the FileSystem Manager interface which allows users or system administrators to mount objects such as directories from a remote location and use them as if they were on local host. The interface shows mounted filesystem, unmounted filesystem, root filesystem, etc…

Figure 2 – A File System Manager in NFS

Source: http://osr5doc. ca. caldera. com: 457/NetAdminG/fsD. aboutGUI. html

ROLE OF NFS SERVER AND CLIENT

NFS Server allows disk file system to be accessed. It also performs file system sharing by exporting files to users. On the other hand, NFS Client creates list of files to be mounted by accepting full pathnames. It also authenticates client requests and returns a handle.

NFS SECURITY

NFS can be hacked into using two types of attacks. They are Eavesdropping and Imposter attack. Eavesdropping is an attack whereby a hacker can pick up data (unauthorised) while it is in transmission. An Imposter hacker is a person who actually gains unauthorised access to the network.

ADVANTAGES AND DISADVANTAGES OF NFS

NFS has many advantages. One of the major advantages of NFS is that data can be kept on a central host. All the data accessed by the users can be kept on a central host. For example, you can have all the user accounts mounted on host server 1 and all the other hosts on the network can mount from host server 1.

Data requiring larger disk space can also be kept on one host. Again, clients wishing to access objects will mount the required files onto their local host. For example, programs and files relating to a specific department can be kept and maintained on one host. Likewise, acconting anf finance department can be kept on another host.

A COMPARISON OF NFS AND NTFS

NTFS's scalability is limited as it can support fewer processors as compared to NFS. A file in NTFS is not simple Byte stream as it is in the NFS. In NTFS, a file is a structured object consisting of attributes.

In NTFS every file is stored in an array structure called Master File Table (MFT).

In NTFS, each directory uses B+ tree structure to store an index of the filename.

CONCLUSION

In my conclusion, I have to say that NFS is very simple to install and implement. It is a highly portable file system although sometimes it is inconsistent. The only drawback of NFS is the fact that it doesn't scale to large number of clients.

BIBLIOGRAPHY

Websites:

http://cs. gmu. edu/~menasce/osbook/distfs/sld091. html

http://cs. gmu. edu/~menasce/osbook/distfs/sld096. html

http://osr5doc. ca. caldera. com: 457/NetAdminG/fsD. aboutGUI. html

http://www. media. kyoto-u. ac. jp/edu/lec/jnakamu/lecture/y98/miy98/part2/p871bp4. htm

http://www-1. ibm. com/servers/eserver/zseries/zos/nfs/index. html

http://www. cs. wisc. edu/~sschang/OS-Qual/fs/distributed_file_systems. htm

http://www. cse. ucsc. edu/~darrell/classes/111/slides/silbershatz/mod17. 1. pdf

Books:

Silberschatz , Abraham. Applied Operating System Concepts / Avi Silberschatz, Peter Galvin, Greg Gagne. New York; Chichester: Wiley , 2000.