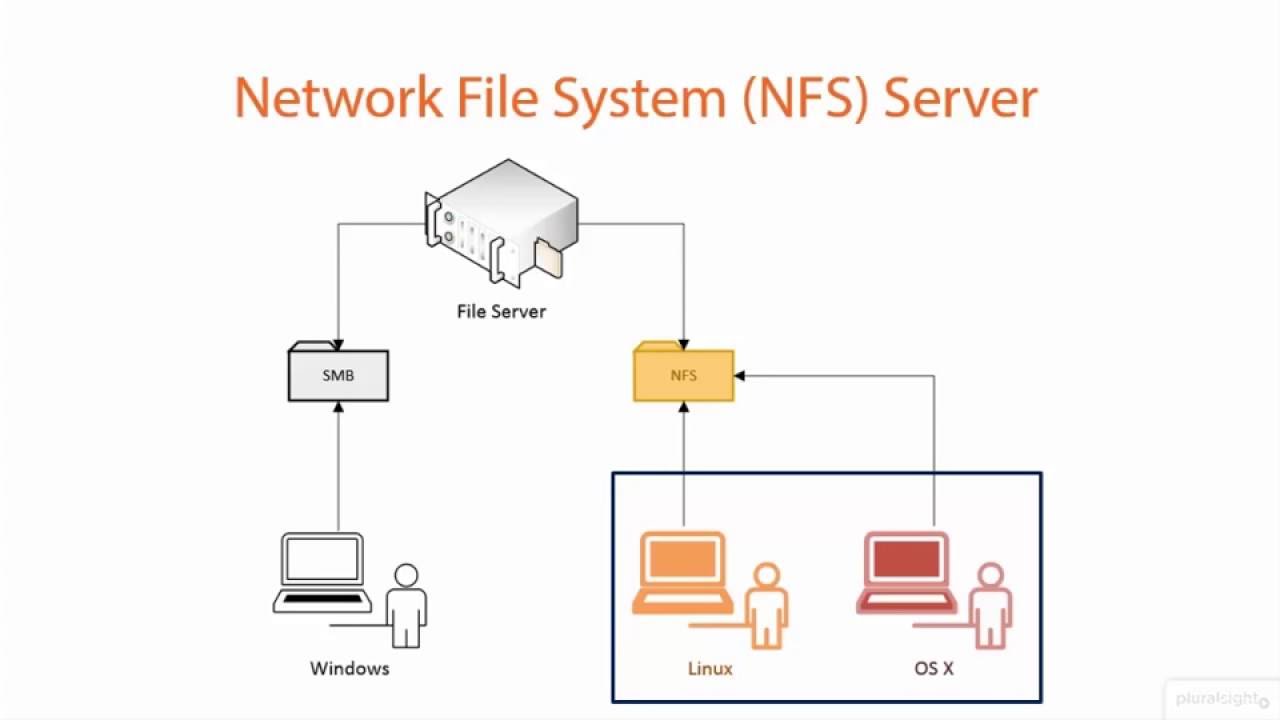
***Aim :*** Study Of Network File SystemAnd Distributed File System

***Prequisite :*** Computer Network

***NFS Diagram***



***Theory :***

NFS stands for the network file system. It is a type of file system mechanism that enables data storage and retrieval from multiple disks and directories across a shared network.

A Network file system is implemented in a client/server computing model. An NFS server manages the authentication, authorisation and management of clients and all the data shared within a specific file system. Once authorised, users can view and access the data through their local systems, much like they should access it from an internal disk drive.

Network File System is a protocol that operates on all networks IP-based. It is implemented in that client/server application in which the server of NFS manages the authorization, authentication, and clients. This protocol is used with Apple Mac OS, UNIX, and Unix-like operating systems such as Solaris, Linux, FreeBSD, and AIX.

It is equivalent to NFS. Using the SMB protocol allows a customer to access files or other resources at a small server. While there are many equalities between SMB and NFS like the opportunity for shared create to documents, printers, and serial ports between nodes on a web there are some notable differences, too.

NFS can arise to send documents in a local directory, enabling users’ systems to approach remote data as a local folder. The users can pass through subdirectories, look up file permissions, and read, write, and generate files. NFS interprets the document paths and file commands to work with the equivalent file system.

The NFS protocol supports a set of RPCs for remote services such as lookup, create, rename, getattr, setattr, read, write, delete, mkdir, etc. The processes can be conjured just after a file manager for the remotely mounted directory has been created. NFS servers are stateless servers.

A stateless file server prevents keeping state data by creating each request self-included. That is, each request recognizes the file and the location of the file in complete. Therefore, the server requires not to save a file pointer.

It requires not to create or remove a connection by opening a file or closing a file, respectively. For reading a directory, NFS does not employ any file pointer, it operates a magic cookie.

***Advantages of NFS :***

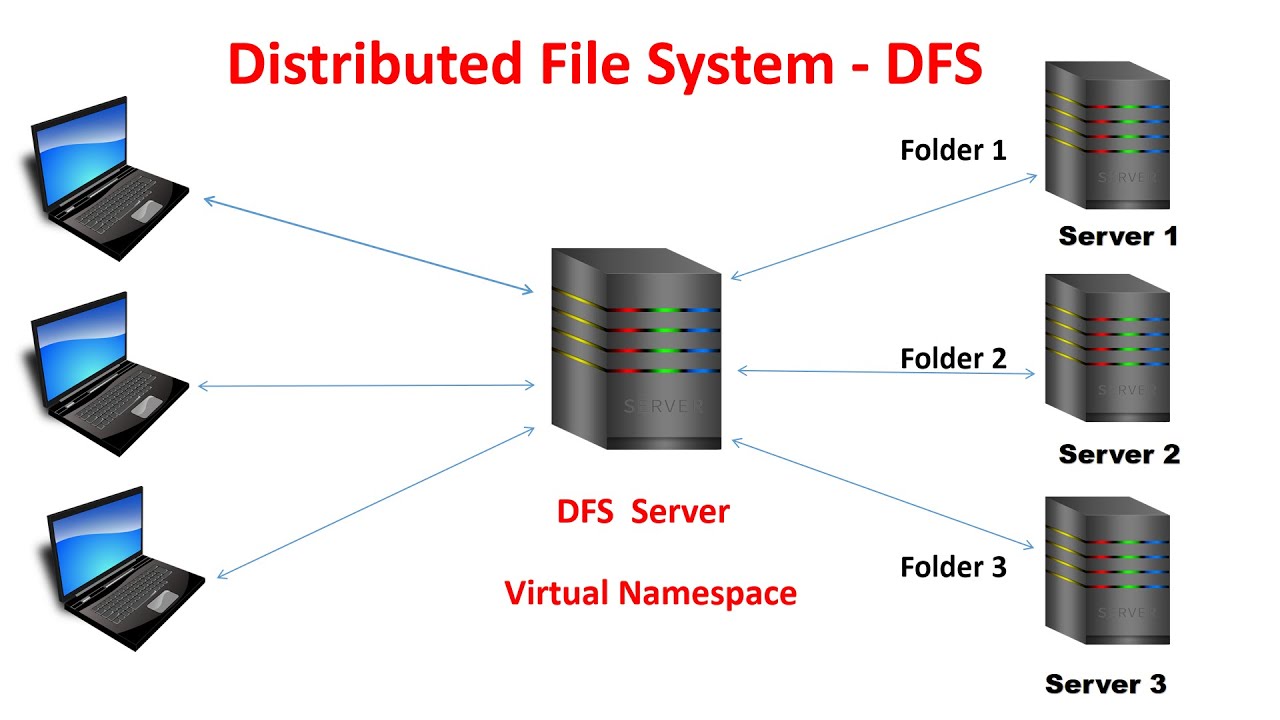
The benefits of NFS are as follows −

* NFS supports central management.
* NFS allows for a customer to log into any server and have access to their files transparently.
* There is no manual refresh needed for new files.
* It can secure it with firewalls and Kerberos.
* Multiple clients can use the same files, which allows everyone on the network to use the same data, accessing it on remote hosts as if it were acceding local files.
* Computers share applications, which eliminates the needs for local disk space and reduces storage costs.
* All users can read the same files, so data can remain up-to-date, and it’s consistent and reliable.
* Mounting the file system is transparent to all users.
* Support for heterogeneous environments allows you to run mixed technology from multiple vendors and use interoperable components.
* System admin overhead is reduced due to centralization of data.
* Fewer removable disks and drives laying around provides a reduction of security concerns—which is always good!

***Disadvantages Of NFS :***

* **Security**—First and foremost is a security concern, given that NFS is based on RPCs which are inherently insecure and should only be used on a trusted network behind a firewall. Otherwise, NFS will be vulnerable to internet threats.
* **Protocol chattiness**—The NFS client-server protocol requires a lot of request activity to be set up to transfer data. The NFS protocol requires many small interactions or steps to read and write data, which equates to a ton of overhead for someone actively interacting with today’s AI/ML/DL workloads that consume a tremendous number of small files.
* **File sharing is highly complex**—Configuring and setting up proper shared file access via file locking and caching is a daunting task at best. On the one hand, it adds a lot of the protocol overhead, leading to the chattiness mentioned above. On the other hand, it still leaves a lot to be desired, inasmuch any each host’s mount command for the same file system can easily go away.
* **Parallel file access**—NFS was designed as a way to sequentially access a shared network file, but these days applications are dealing with larger files and non-sequential or parallel file access is required. This was added to NFSv4, but not a lot of clients support it yet.
* **Block size limitations**—The current NFS protocol standard allows for a maximum of 1MB of data to be transferred during one read or write request. In 1984, 1MB was a lot of data, but that’s no longer the case. There are classes of applications that should be transferring GBs not MBs of data.

**DFS Diagram**



***Theory :***

A **Distributed File System (DFS)**as the name suggests, is a file system that is distributed on multiple file servers or multiple locations. It allows programs to access or store isolated files as they do with the local ones, allowing programmers to access files from any network or computer.

The main purpose of the Distributed File System (DFS) is to allows users of physically distributed systems to share their data and resources by using a Common File System. A collection of workstations and mainframes connected by a Local Area Network (LAN) is a configuration on Distributed File System. A DFS is executed as a part of the operating system. In DFS, a namespace is created and this process is transparent for the clients.

 DFS has two components:

* **Location Transparency –**  
  Location Transparency achieves through the namespace component.
* **Redundancy –**  
  Redundancy is done through a file replication component.

In the case of failure and heavy load, these components together improve data availability by allowing the sharing of data in different locations to be logically grouped under one folder, which is known as the “DFS root”.

It is not necessary to use both the two components of DFS together, it is possible to use the namespace component without using the file replication component and it is perfectly possible to use the file replication component without using the namespace component between servers.

#### **Features of DFS :**

* **User mobility :**   
  It will automatically bring the user’s home directory to the node where the user logs in.
* **Performance :**   
  Performance is based on the average amount of time needed to convince the client requests. This time covers the CPU time + time taken to access secondary storage + network access time. It is advisable that the performance of the Distributed File System be similar to that of a centralized file system.
* **Simplicity and ease of use :**   
  The user interface of a file system should be simple and the number of commands in the file should be small.
* **High availability :**   
  A Distributed File System should be able to continue in case of any partial failures like a link failure, a node failure, or a storage drive crash.   
  A high authentic and adaptable distributed file system should have different and independent file servers for controlling different and independent storage devices.
* **Scalability :**   
  Since growing the network by adding new machines or joining two networks together is routine, the distributed system will inevitably grow over time. As a result, a good distributed file system should be built to scale quickly as the number of nodes and users in the system grows. Service should not be substantially disrupted as the number of nodes and users grows.
* **High reliability :**  
  The likelihood of data loss should be minimized as much as feasible in a suitable distributed file system. That is, because of the system’s unreliability, users should not feel forced to make backup copies of their files. Rather, a file system should create backup copies of key files that can be used if the originals are lost. Many file systems employ stable storage as a high-reliability strategy.
* **Data integrity :**  
  Multiple users frequently share a file system. The integrity of data saved in a shared file must be guaranteed by the file system. That is, concurrent access requests from many users who are competing for access to the same file must be correctly synchronized using a concurrency control method. Atomic transactions are a high-level concurrency management mechanism for data integrity that is frequently offered to users by a file system.
* **Security :**   
  A distributed file system should be secure so that its users may trust that their data will be kept private. To safeguard the information contained in the file system from unwanted & unauthorized access, security mechanisms must be implemented.

#### **Advantages :**

* DFS allows multiple user to access or store the data.
* It allows the data to be share remotely.
* It improved the availability of file, access time, and network efficiency.
* Improved the capacity to change the size of the data and also improves the ability to exchange the data.
* Distributed File System provides transparency of data even if server or disk fails.

#### **Disadvantages :**

* In Distributed File System nodes and connections needs to be secured therefore we can say that security is at stake.
* There is a possibility of lose of messages and data in the network while movement from one node to another.
* Database connection in case of Distributed File System is complicated.
* Also handling of the database is not easy in Distributed File System as compared to a single user system.
* There are chances that overloading will take place if all nodes tries to send data at once.