Distributed File System

Introduction

- The main purposes of using file in operating systems are:
  - *Permanent storage of information*
  - *Sharing the information*: A file can be created by one application and the shared with different applications.

- A file system is a subsystem of an operating system that perform file management activities such as organization, storing, retrieval, naming, sharing and protection of files.

Distributed File System

- 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.

What distributed file system provide ?

Access to data stored at servers using file system interfaces

What are the file system interfaces?

- Open a file, check status of a file, close a file
- Read data from a file
- Write data to a file
- Lock a file or part of a file
- List files in a directory, create/delete a directory
- Delete a file, rename a file, add a symlink (is a special type of fileis a special type of file that contains a reference to another file or directory in the form of an absolute or relative path) to a file etc

- 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.

- A distributed file system works as follows:
- **Distribution**: First, a DFS distributes datasets across multiple clusters or nodes.

  Each node provides its own computing power, which enables a DFS to process the datasets in parallel.
- **Replication**: A DFS will also replicate datasets onto different clusters by copying the same pieces of information into multiple clusters.
  - This helps the distributed file system to achieve fault tolerance

  —to recover the data in case of a node or cluster failure

  —as well as high concurrency, which enables the same piece of data to be processed at the same time.

File system replication:

- Early iterations of DFS made use of Microsoft’s File Replication Service (FRS), which allowed for straightforward file replication between servers.
- The most recent iterations of the whole file are distributed to all servers by FRS, which recognizes new or updated files.
- “DFS Replication” was developed by Windows Server 2003 R2 (DFSR). By only copying the portions of files that have changed and minimising network traffic with data compression, it helps to improve FRS.

Features of DFS :

- **Transparency :**
  - **Structure transparency –**
    There is no need for the client to know about the number or locations of file servers and the storage devices.
    - Multiple file servers should be provided for performance, adaptability, and dependability.
  - **Access transparency –**
    Both local and remote files should be accessible in the same manner. The file system should be automatically located on the accessed file and send it to the client's side.
  - **Naming transparency –**
    There should not be any hint in the name of the file to the location of the file. Once a name is given to the file, it should not be changed during transferring from one node to another.
  - **Replication transparency –**
    If a file is copied on multiple nodes, both the copies of the file and their locations should be hidden from one node to another.

- **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.

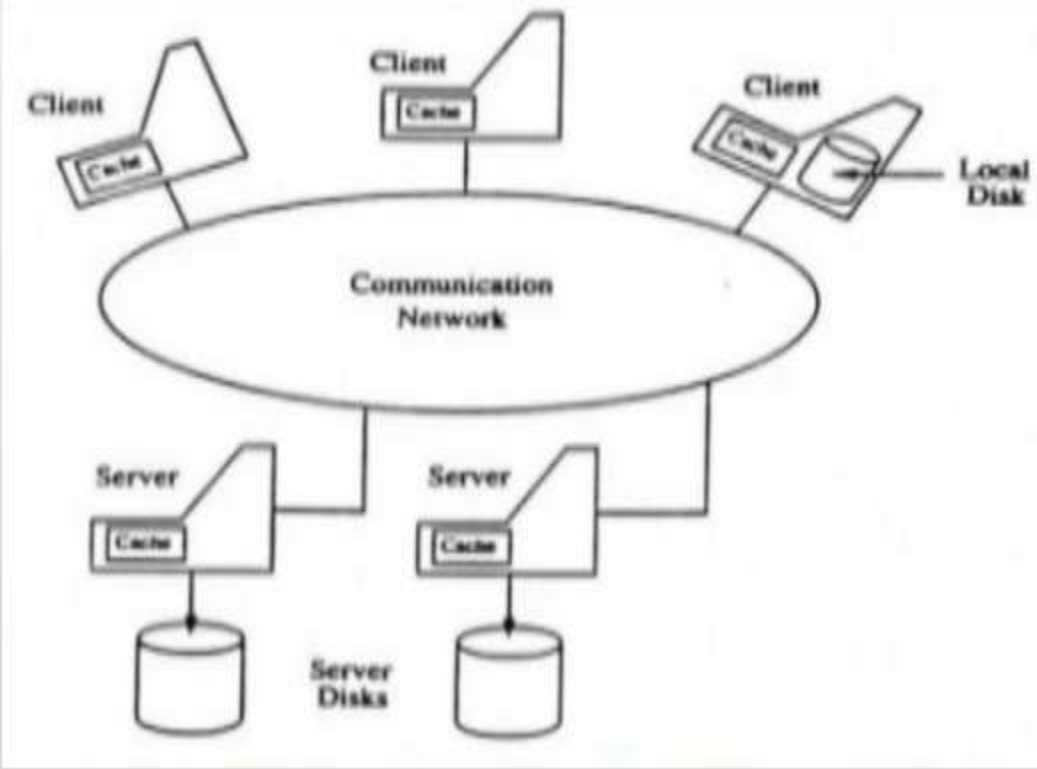
- **High reliability :**
  The likelihood of data loss should be minimized as much as feasible in a suitable distributed file system.
  - That is, because of the system’s unreliability, users should not feel forced to make backup copies of their files.
  - Rather, a file system should create backup copies 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.

- **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.
- **Heterogeneity :**
  Heterogeneity in distributed systems is unavoidable as a result of huge scale. Users of heterogeneous distributed systems have the option of using multiple computer platforms for different purposes.

Applications :

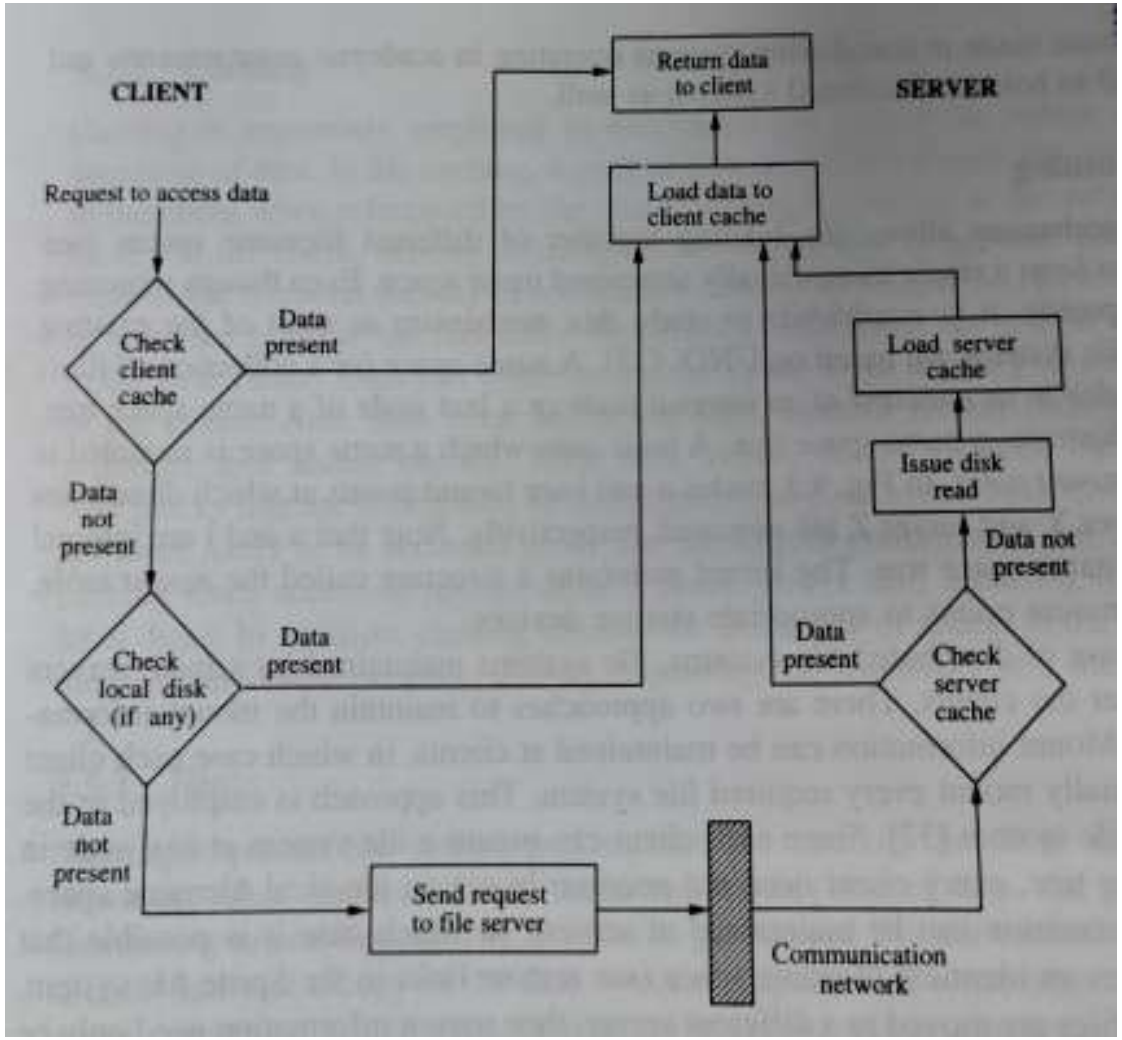
- **NFS –**
  NFS stands for Network File System. It is a client-server architecture that allows a computer user to view, store, and update files remotely. The protocol of NFS is one of the several distributed file system standards for Network-Attached Storage (NAS).
- **CIFS –**
  CIFS stands for Common Internet File System. CIFS is an accent of SMB(Server Message Block). That is, CIFS is an application of SMB protocol, designed by Microsoft.
- **SMB –**
  SMB stands for Server Message Block. It is a protocol for sharing a file and was invented by IMB(Integrated Module Board). The SMB protocol was created to allow computers to perform read and write operations on files to a remote host over a Local Area Network (LAN). The directories present in the remote host can be accessed via SMB and are called as “shares”.
- **Hadoop –**
  Hadoop is a group of open-source software services. It gives a software framework for distributed storage and operating of big data using the MapReduce programming model. The core of Hadoop contains a storage part, known as Hadoop Distributed File System (HDFS), and an operating part which is a MapReduce programming model.
- **NetWare –**
  NetWare is an abandon computer network operating system developed by Novell, Inc. It primarily used combined multitasking to run different services on a personal computer, using the IPX network protocol.

Architecture



- A cache manager is a program that stores data in the form of a cache.
- Intermediary between a computer's main memory and the permanent storage system.
- The cache manager has two main functions: to improve performance by storing copies of recently accessed data in memory. In addition, it also protects against data loss by keeping backup copies of data on permanent storage.
- Use cache managers for many different purposes, such as caching web pages for faster access, caching files for faster loading times, and optimizing disk usage.
- It is the client side of DFS that checks the local cache at first on receiving a user's request.
- If no similar file is found in the local cache, the cache manager forwards the request to the file server machine and caches data on disk or in memory.

Architecture



Mechanisms For Building Distributed File Systems

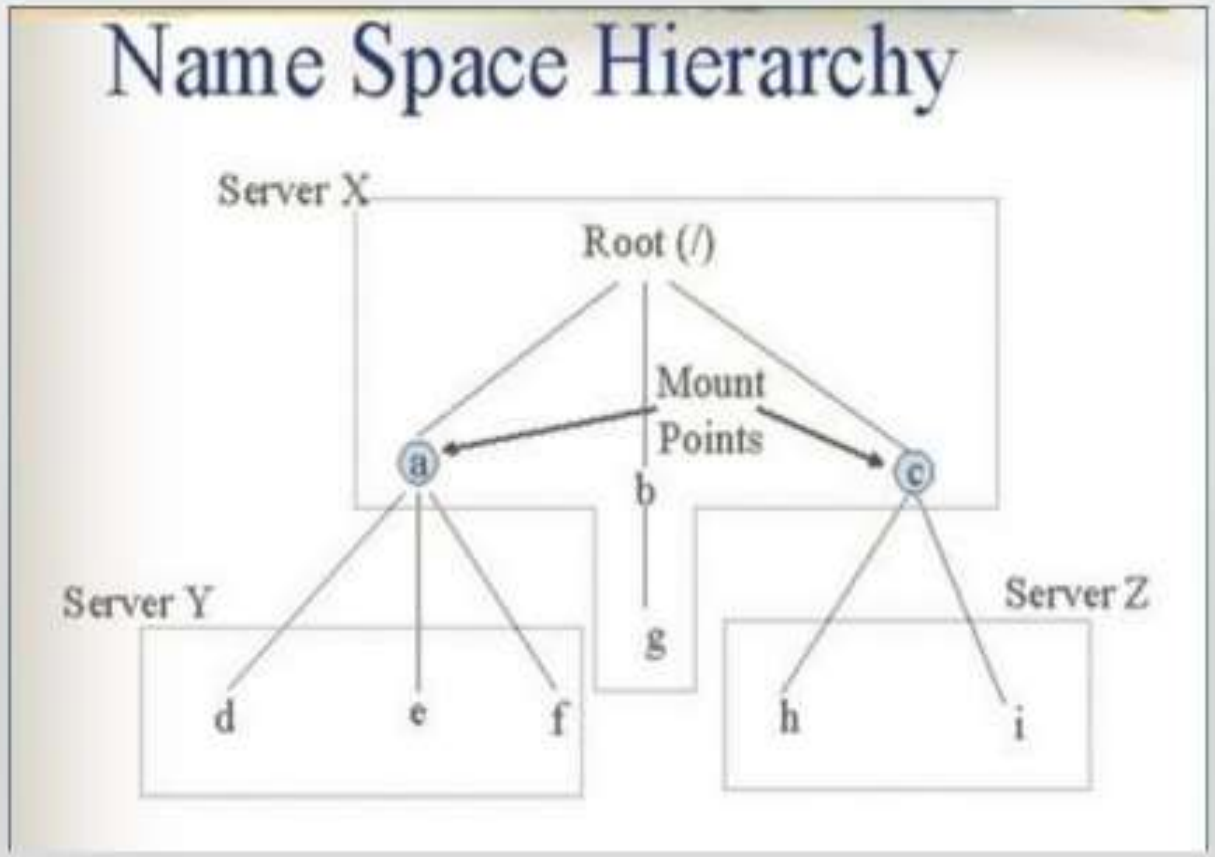
The basic mechanisms underlying the majority of the distributed file systems operating today are:

- Mounting
- Caching
- Hints
- Bulk Data Transfer
- Encryption

Mounting

- This mechanism provides the binding together of different filename spaces to form a single hierarchically structured name space.
- It is UNIX specific and most of existing DFS are based on UNIX.
- A filename space can be bounded to or mounted at an internal node or a leaf node of a namespace tree.
- A node onto which a name space is mounted is called mount point.
- The kernel maintains a mount table, which maps mount points to appropriate storage devices.

Mounting


Name Space Hierarchy
Server X
Root (/)
Mount
Points
a
b
c
Server Y
Server Z
g
d
e
f
h
i

Mounting

Uses of Mounting in DFS

- File systems maintained by remote servers are mounted at clients so that each client have information regarding file servers.
- Two approaches are used to maintain mount information.

Approach 1: Mount information is maintained at clients that is each client has to individually mount every required file system. When files are moved to a different server then mount information must be updated in mount table of every client.

Approach 2: Mount information is maintained at servers. If files are moved to a different servers, then mount information need only be updated at servers.

Caching

- This mechanism is used in DFS to reduce delays in accessing of data.
- In file caching, a copy of data stored at remote file server is brought to client when referenced by client
- Subsequent access of data is performed locally at client, thus reducing access delays due to network latency.
- Data can be cached in main memory or on the local disk of the clients.
- Data is cached in main memory at servers to reduce disk access latency.

Need of Caching in DFS

- File system performance gets improved accessing remote disks is much slower than accessing local memory or local disks. It also reduces the frequency of access to file servers and the communication network, so scalability gets increased.

Hints

A timestamp based method is used to validate cached block before they are used

- Caching results in the cache consistency problem when multiple clients cache and modify shared data.
- This problem can be avoided by great level of co-operation between file servers and clients which is very expensive.
- Alternative method is that is cached data are not expected to be completely accurate.

Bulk Data Transfer

- In this mechanism, multiple consecutive data blocks are transferred from server to client.
- This reduces file access overhead by obtaining multiple number of blocks with a single seek, by formatting and transmitting multiple number of large packets in single context switch and by reducing the number of acknowledgement that need to be sent.
- This mechanism is used as many files are accessed in their entirely

Encryption

- This mechanism is used for security in Distributed systems.
- The method was developed by Needham Schrodkar is used in DFS security.
- In this scheme, two entities which want to communicate establish a key for conversation with help of authentication server.
- The conversation key is determined by the authentication server, but is never sent in plain text to either of the entities.