UNIT- V Distributed File System

What is DFS (Distributed File System)?

A Distributed File System (DFS) 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. In this article, we will discuss everything about Distributed File System.

What is DFS (Distributed File System)?

A distributed file system (DFS) is a networked architecture that allows multiple users and applications to access and manage files across various machines as if they were on a local storage device. Instead of storing data on a single server, a DFS spreads files across multiple locations, enhancing redundancy and reliability.

- This setup not only improves performance by enabling parallel access but also simplifies data sharing and collaboration among users.
- By abstracting the complexities of the underlying hardware, a distributed file system provides a seamless experience for file operations, making it easier to manage large volumes of data in a scalable manner.

Components of DFS

- **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.

Distributed 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 recognises 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. Additionally, it provides users with flexible configuration options to manage network traffic on a configurable schedule.

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.
- **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.

Applications of DFS

- **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. That is, CIFS is an application of SIMB protocol, designed by Microsoft.
- **SMB:** SMB stands for Server Message Block. It is a protocol for sharing a file and was invented by IBM. 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.

Advantages of Distributed File System(DFS)

- 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.

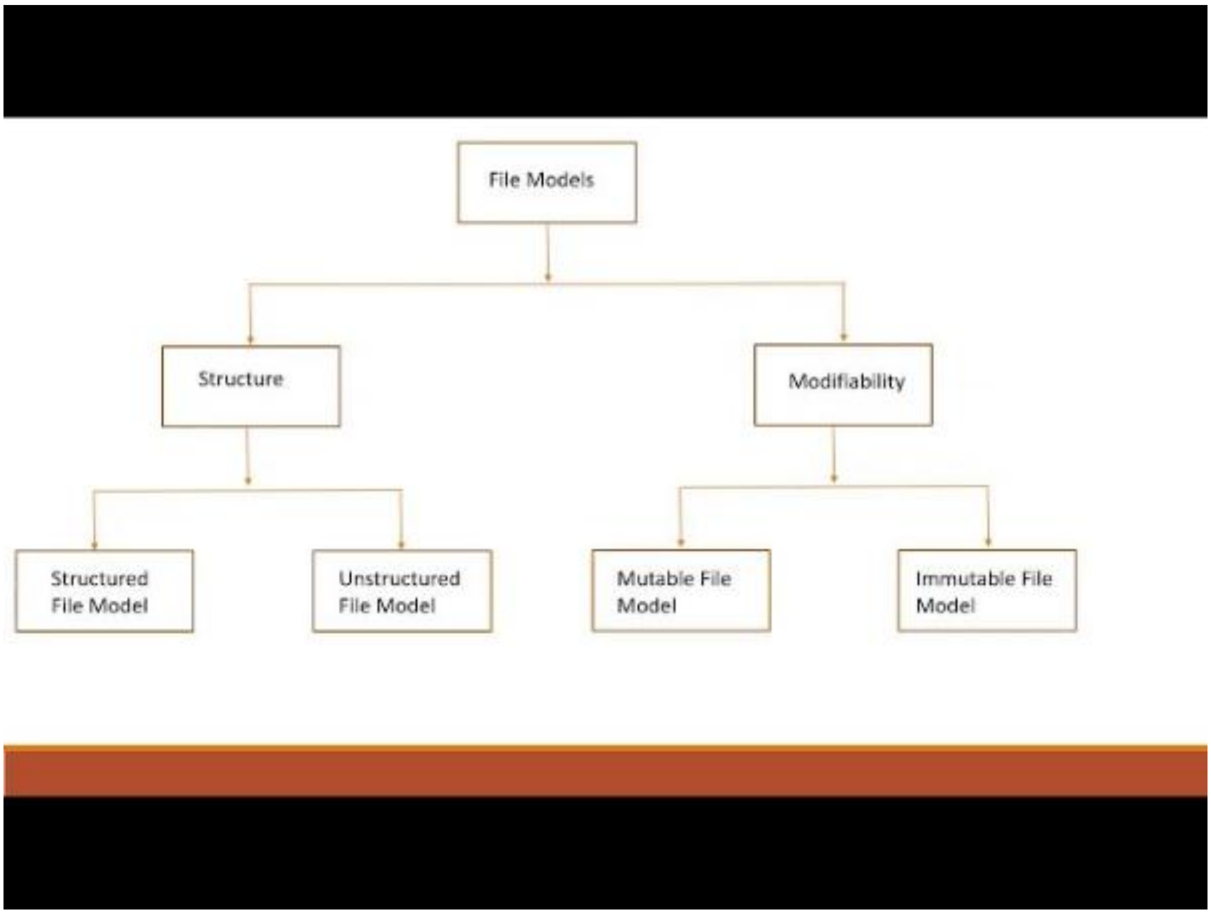
Disadvantages of Distributed File System(DFS)

- 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.

File Accessing Models in Distributed System

Last Updated : 16 Feb, 2023

In Distributed File Systems (DFS), multiple machines are used to provide the file system's facility. Different file system utilize different conceptual models of a file. The two most usually involved standards for file modeling are structure and modifiability. File models in view of these standards are described below.



File Accessing Models:

The file accessing model basically to depends on

- **The unit of data access/Transfer**
- **The method utilized for accessing to remote files**

Based on the unit of data access, following file access models may be utilized to get to the particular file.

1. File-level transfer model: In file level transfer model, the all out document is moved while a particular action requires the document information to be sent the whole way through the circulated registering network among client and server. This model has better versatility and is proficient.

2. Block-level transfer model: In the block-level transfer model, record information travels through the association among client and a server is accomplished in units of document blocks. Thus, the unit of information move in block-level transfer model is document blocks. The block-level transfer model might be used in dispersed figuring climate containing a few diskless workstations.

3. Byte-level transfer model: In the byte-level transfer model, record information moves the association among client and a server is accomplished in units of bytes. In this way, the unit of information move in byte-level exchange model is bytes. The byte-level exchange model offers more noteworthy versatility in contrast with the other record move models since, it licenses recuperation and limit of a conflicting progressive sub range of a document. The significant hindrance to the byte-level exchange model is the trouble in store organization because of the variable-length information for different access requests.

4. Record-level transfer model: The record-level file transfer model might be used in the document models where the document contents are organized as records. In record-level exchange model, document information travels through the organization among client and a server is accomplished in units of records. The unit of information move in record-level transfer model is record.

The Method Utilizes for Accessing Remote Files:

A distributed file system might utilize one of the following models to service a client's file access request when the accessed to file is remote:

1. Remote service model: Handling of a client's request is performed at the server's hub. Thusly, the client's solicitation for record access is passed across the organization as a message on to the server, the server machine plays out the entrance demand, and the result is shipped off the client. Need to restrict the amount of messages sent and the vertical per message.

- Remote access is taken care of across the organization so it is all the slower.
- Increase server weight and organization traffic. Execution undermined.
- Transmission of series of responses to explicit solicitation prompts higher organization overhead.
- For staying aware of consistency correspondence among client and server is there to have a specialist copy predictable with clients put away data.
- Remote assistance better when essential memory is close to nothing.

- It is only an augmentation of neighborhood record system interface across the network.

2. Data-caching model: This model attempts to decrease the organization traffic of the past model by getting the data got from the server center. This exploits the region part of the found in record gets to. A replacement methodology, for instance, LRU is used to keep the store size restricted.

- Remote access can be served locally so that access can be quicker.
- Network traffic, server load is reduced. Further develops versatility.
- Network over head is less when transmission of huge of information in comparison to remote service.
- For keeping up with consistency, if less writes then better performance in maintaining consistency ,if more frequent writes then poor performance.
- Caching is better for machines with disk or large main memory.
- Lower level machine interface is different from upper level UI(user interface).

Benefit of Data-caching model over the Remote service model:

The data -catching model offers the opportunity for expanded execution and greater system versatility since it diminishes network traffic, conflict for the network, and conflict for the document servers. Hence almost all distributed file systems implement some form of caching.

Introduction to Distributed Computing Environment (DCE)

The Benefits of Distributed Systems have been widely recognized. They are due to their ability to Scale, Reliability, Performance, Flexibility, Transparency, Resource-sharing, Geo-distribution, etc. In order to use the advantages of Distributed Systems, appropriate support and environment are needed that supports execution and development of Distributed Applications.

A distributed application is a program that runs on more than one machine and communicates through a network. It consists of separate parts that execute on different nodes of the network and cooperate in order to achieve a common goal. It uses Client-Server Model.

Distributed Computing Environment(DCE) is an integrated set of services and tools which are used for building and running Distributed Applications. It is a collection of integrated software components/frameworks that can be installed as a coherent environment on top of the existing Operating System and serve as a platform for building and running Distributed Applications.

Using DCE applications, users can use applications and data at remote servers. Application programmers or clients need not be aware of where their programs will run or where the data that they want to have access, will be located.

DCE was developed by the Open Software Foundation(OSF) using software technologies contributed by some of its member companies which are now popularly known as The Open Group.

DCE framework/Services include:

- **Remote Procedure Call(RPC):** It is a call made when a Computer program wants to execute a subroutine in a different computer(another computer on a shared network).

- **Distributed File System(DFS):** It provides a transparent way of accessing a file in the system in the same way as if it were at the same location.|

- **Directory Service:** It is used to keep track location of Virtual Resources in the Distributed System. These Resources include Files, Printers, Servers, Scanner, and other machines. This service prompts the user to ask for resources(through the process) and provide them with convenience. Processes are unaware of the actual location of resources.

- **Security Service:** It allows the process to check for User Authenticity. Only an authorized person can have access to protected and secured resources. It allows only an authorized computer on a network of Distributed Systems to have access to secured resources.

- **Distributed Time Service:** Inter-Process Communication between different system components requires synchronization so that communication takes place in a designated order only. This service is responsible for maintaining a global clock and hence synchronizing the local clocks with the notion of time.

- **Thread Service:** The Thread Service provides the implementation of lightweight processes (threads). Helps in the synchronization of multiple threads within a shared address space.

DCE Architecture

DCE supports the structuring of distributed computing systems into so-called **cells** which consist of 3 types of machines, User, administrator, and Server. This is done to keep the size of the administration domain manageable. A cell is basically a set of nodes that are managed together by one authority.
Cell boundaries of a cell represent security firewalls; access to resources in a foreign cell requires special authentication and authorization procedures that are different from secure intra-cell interactions.
The highest privileges within a cell are assigned to a role called **DCE cell administrator** which has control over all system services within the network, remotely. It has privileges over all resources within a **Distributed Computing Environment cell.**
Major components of cell:

- **Security Server** which is responsible for User Authenticity
- **Cell Directory Server(CDS) –** the repository of resources
- **Distributed Time Server** – provides the clock for synchronization of the entire cell.

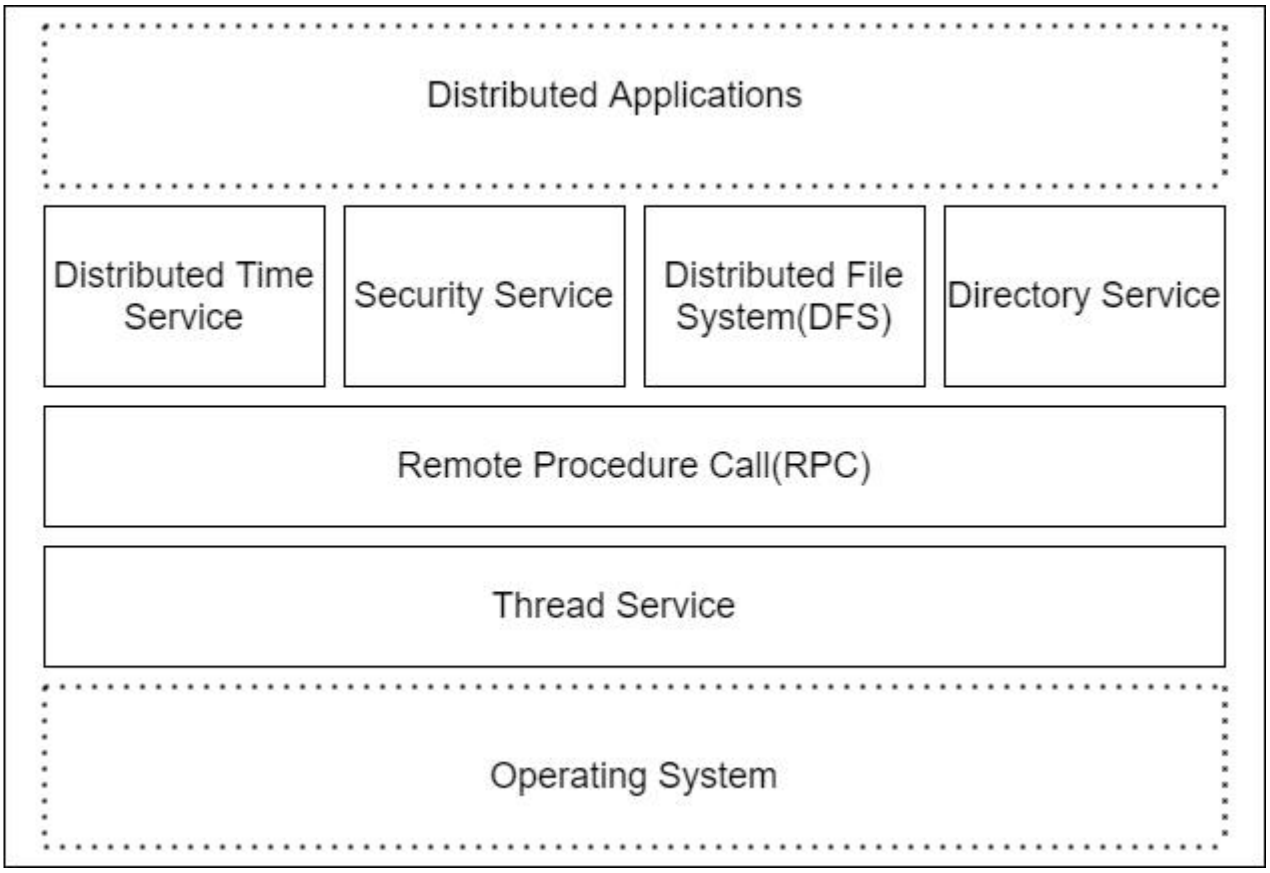


figure 1: DCE Architecture

Advantages of DCE:

- Security
- Lower Maintenance Cost
- Scalability and Availability
- Reduced Risks