# 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.

# Introduction

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

# Introduction

Clients and Servers:

- Clients access files and directories that are provided by one or more file servers.
- File Servers provide a client with a file service interface and a view of the file system
- Servers allow clients to perform operations from the file service interface on the files and directories
- Operations: add/remove, read/write
- Servers may provide different views to different clients

# Introduction

- A distributed file system is a resource management component of a distributed operating system
- It implements a common file system that can be shared by all the autonomous computer in the system

# Introduction

A good distributed file system should have the following features.

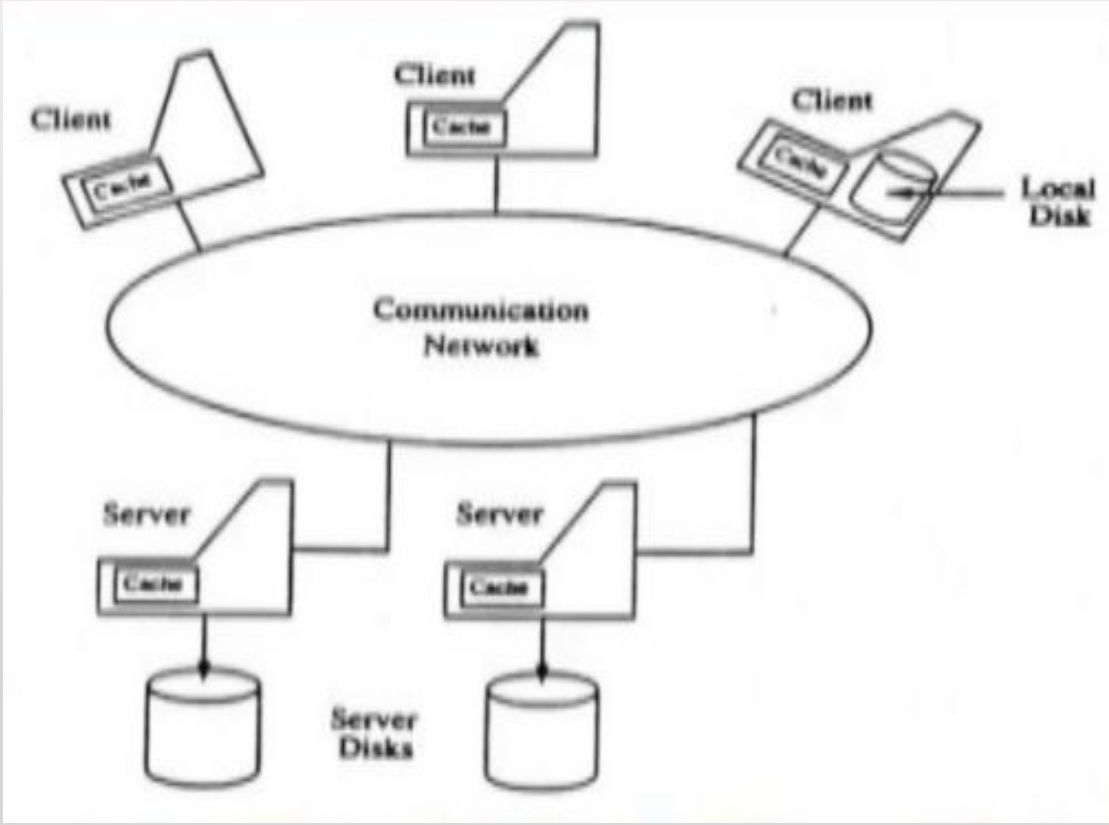
1. Transparency – Structure, access, naming
2. Simplicity and ease of use
3. Reliability
4. Performance
5. Scalability
6. Data integrity
7. Security

# Introduction

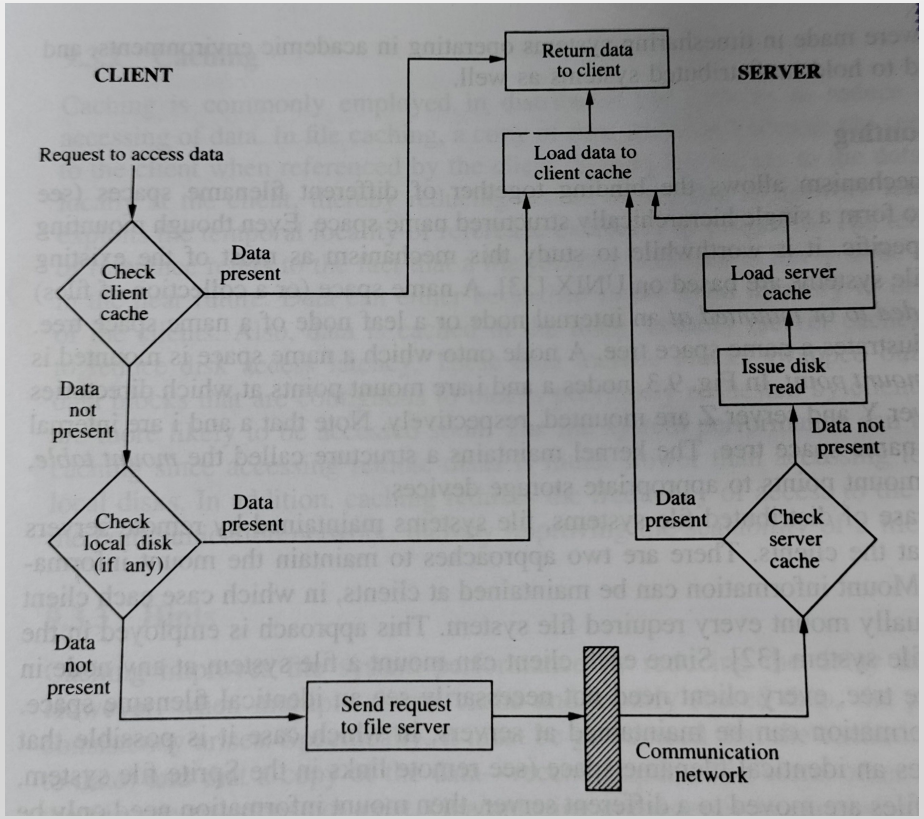
Two important goals of a distributed file systems are :

1. Network transparency
2. High Availability

# Architecture



# Architecture



# Mechanisms For Building Distributed File Systems

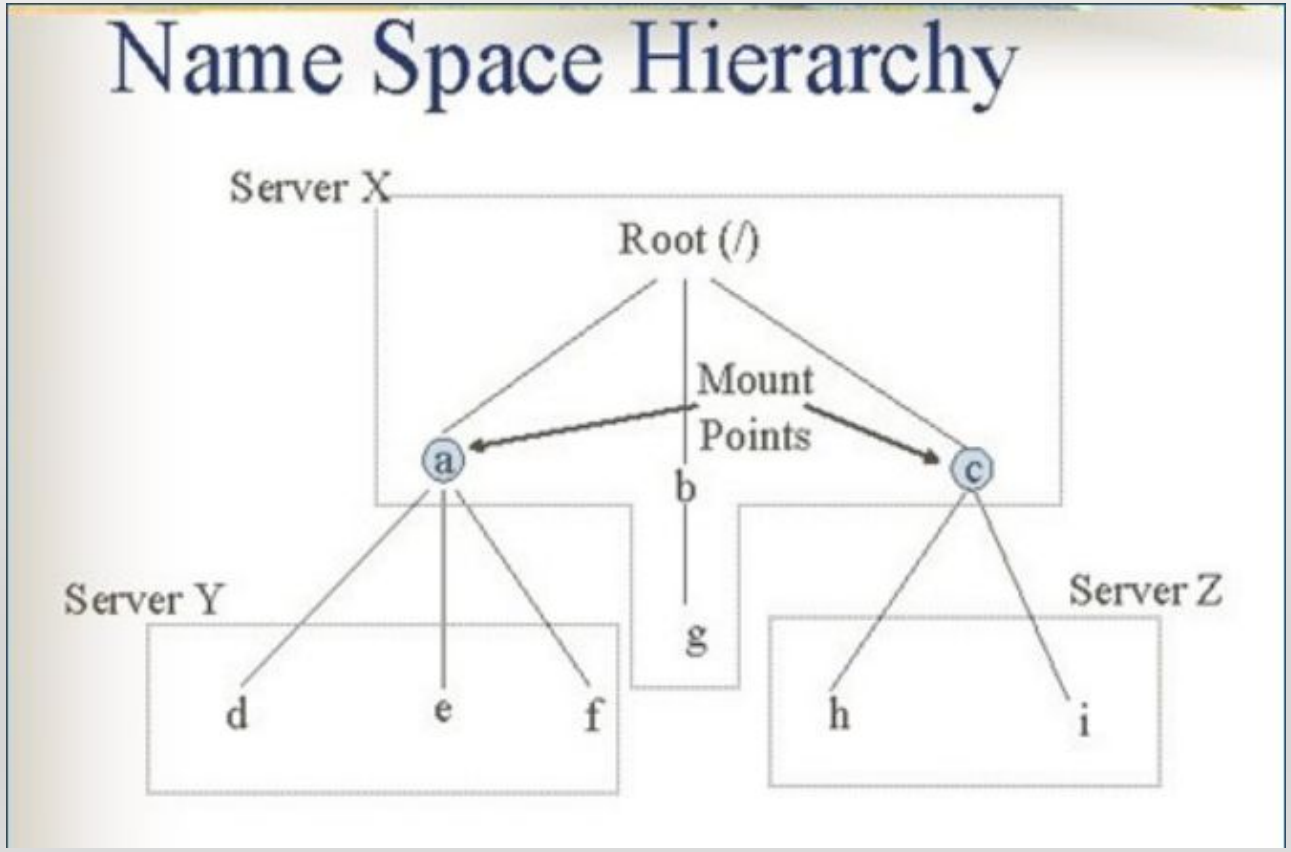
The basic mechanisms underlying the majority of the distributed file systems operating today are:

- 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

- 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.
- Only those class of applications which can recover after discovering that cached data are invalid can use this approach.tim

# 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.

# Design Issues

The various issues that must be addressed in the design and implementation of distributed file systems are:

1. Naming and Name Resolution
2. Caches on Disk or Main Memory
3. Writing Policy
4. Cache Consistency
5. Availability
6. Scalability
7. Semantics

# Naming and Name Resolution

- Name refers to an object such as file or a directory.
- Name Resolution refers to the process of mapping a name to an object that is physical storage.
- Name space is collection of names.
- Names can be assigned to files in distributed file system in three ways:

a) Concatenate the host name to the names of files that are stored on that host.

b) Mount remote directories onto local directories.

c) Maintain a single global directory where all the files in the system belong to single namespace.

# Caches on Disk or Main Memory

- Caching refers to storage of data either into the main memory or onto disk space after its first reference by client machine.

**Advantages of having cache in main memory:**

- Diskless workstations can also take advantage of caching.
- Accessing a cache in main memory is much faster than accessing a cache on local disk.
- The server cache is in the main memory at the server, a single design for a caching mechanism is used for clients and servers.

# Caches on Disk or Main Memory

**Limitations:**

- Large files cannot be cached completely so caching done block oriented which is more complex.
- It competes with virtual memory system for physical memory space, so a scheme to deal with memory contention cache and virtual memory system is necessary.
- Thus, more complex cache manager and memory management is required.

**Advantages of having cache on a local disk**:

- Large files can be cached without affecting performance.
- Virtual memory management is simple.
- Portable workstation can be incorporated in distributed system.

# Writing Policy

This policy decides when a modified cache block at client should be transferred to the server. Following policies are used:

- **Write Through**
- **Delayed Writing Policy**
- Another scheme delays the updating of files at the server until the file is closed at the client.

# Cache Consistency

- When multiple clients want to modify or access the same data, then cache consistency problem arises.
- Two schemes are used to guarantee that data returned to the client is valid.

  a) Server initiated approach

  b ) Client initiated approach

# Availability

- It is one of the important issue is design of Distributed file system.
- Server failure or communication network can affect the availability of files.
- Replication: The primary mechanism used for enhancing availability of files is replication. In this mechanism, many copies or replicas of files are maintained at different server

# Availability

**Limitations**

- Extra storage space is required to store replicas.
- Extra overhead is required in maintained all replicas up to date

**Following situations cause inconsistency among replicas**

- Replica is not updated due to failure of server storing the replica
- All the file servers storing the replicas of file are not reachable from all clients due to network partition and replicas of file in different partition are updated differently.

# Scalability

- The design of DFS should be such that new systems can be easily introduced without affecting it.
- Generally, client-server organization is used to define DFS structure.
- Caching is used in this organization to improve performance.
- Server initiated cache invalidation is used to maintain cache consistency.
- In this approach, server maintain a record based information regarding all the clients sharing file stored on it. This information represents server state.
- As the system grows both the size of server state and load due to invalidations increases on server.

# Scalability

Following schemes can be used to reduce server state and server load:

a) Exploit knowledge about usage of files that is it is found that most commonly used and shared files are accessed in read only mode. So, there is no need to check the validity of these files of maintain the list of clients at servers for validation purpose.

b) Generally, data required by a client is found in another client's cache so a client can obtain required data from another client rather than server.
   - Structure of server process play an important role.
   - If server is designed with single process, then many clients have to wait for a long time whenever a disk input/ output is initiated. This can be avoided if separate process is assigned to each client.

# Semantics

- The semantic of a file system represent the affects of accesses on file.
- The basic semantic is that a read operation will return the data (stored ) due to latest write operation.
- The semantic can be guaranteed in two ways:
  - All read and writes from various clients will have to go through the server.
  - Sharing will have to be disallowed either by server or by the use of locks by application.
  - In first way, the server become bottleneck and in second way, the file is not available for certain clients.

# Log structured file system

- Two design aspects of existing file systems make it hard to improve file system performance
- Log structured file system have been proposed to deal with the technological and workload changes

# Disk space management

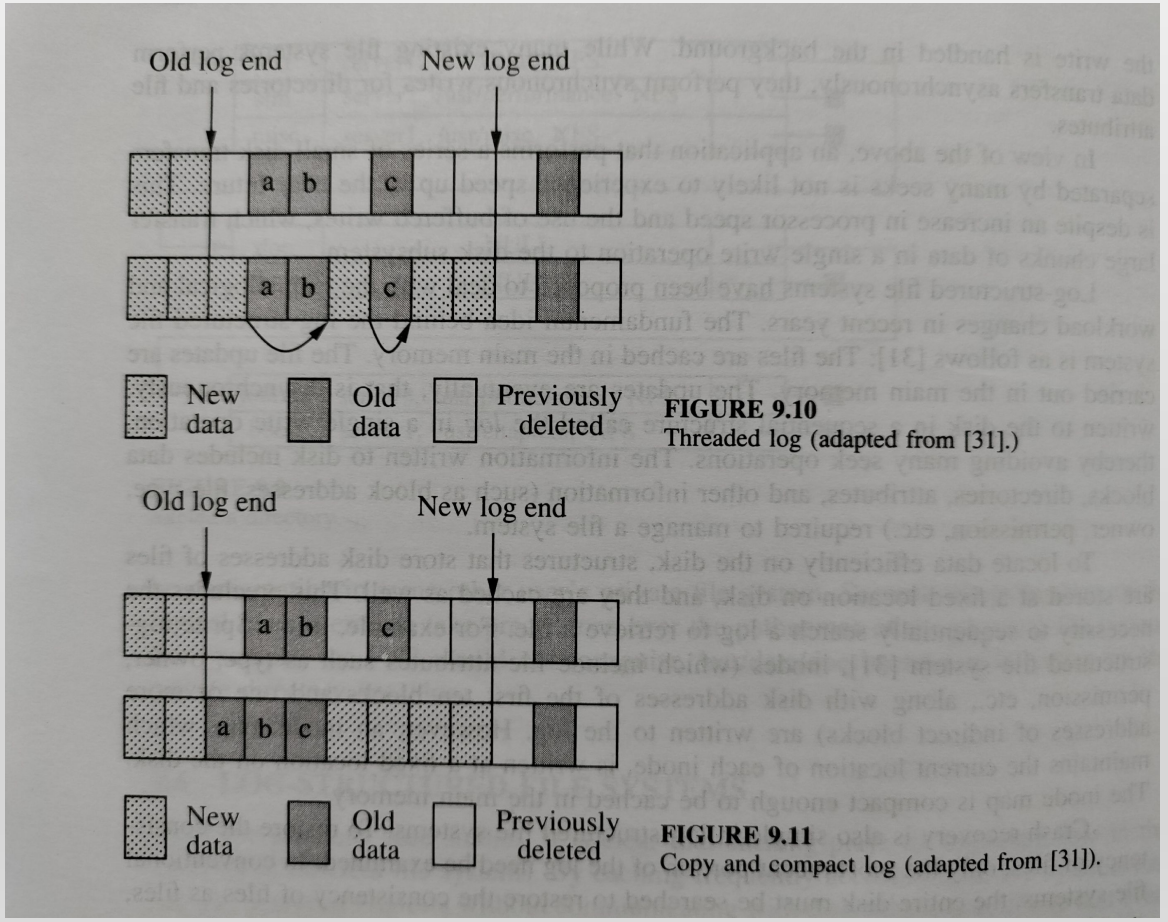



**FIGURE 9.10**
Threaded log (adapted from [31].)




**FIGURE 9.11**
Copy and compact log (adapted from [31]).