UNIT 4

DISTRIBUTED FILE SYSTEM

COURSE CONTAINS

4.1 Introduction to Distributed File System

4.2 File Service Architecture

4.3 Introduction to Name Service

4.4 Name Services and Domain Name System

4.5 Google File System

4.6 Comparison of Different Distributed File System

4.1 INTRODUCTION TO DISTRIBUTED FILE SYSTEM

- Different types of information can be stored in files.
- File system performs organization , storage , retrieval , sharing and protection of files in the system . In distributed system , the file system works as resource management components.
- In computing, a **distributed file system (DFS)** or network file system is any file system that allows access to files from multiple hosts sharing via a computer network. This makes it possible for multiple users on multiple machines to share files and storage resources.
- A **Distributed File System (DFS)** is a file system that is distributed on multiple file servers or multiple locations.
- A distributed file system(DFS) is a method of storing and accessing files based in a client/ server architecture, which give permission to client to access the files or data from server.

- 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.
- The DFS makes it convenient to share information and files among users on a network in a controlled and authorized way.
- A distributed file system (DFS) is a file system with data stored on a server.

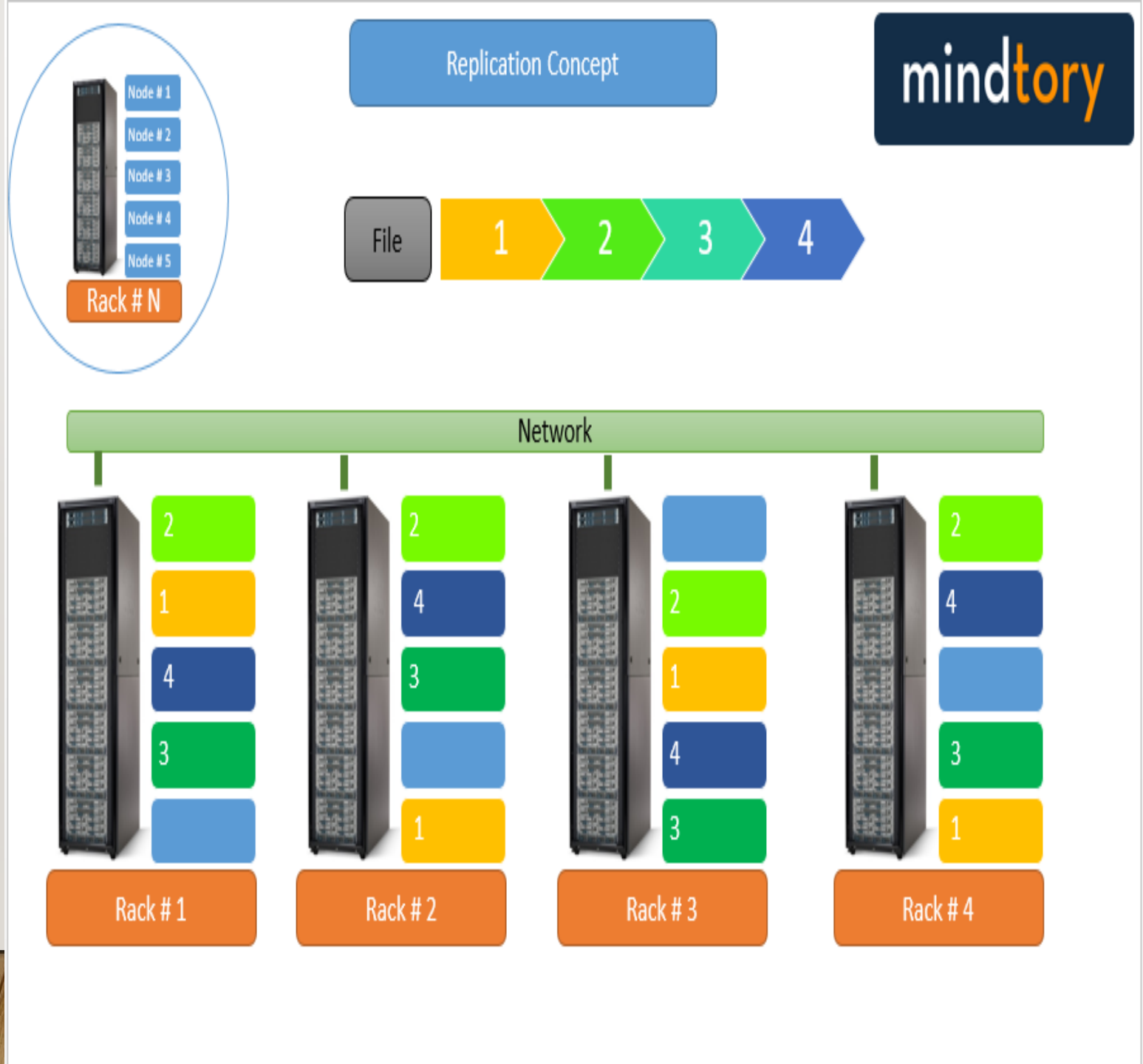
How Distributed file system (DFS) works?

Distributed file system works as follows:

1. Distribution: Distribute blocks of data sets across multiple nodes. Each node has its own computing power; which gives the ability of DFS to parallel processing data blocks.

2. Replication: Distributed file system will also replicate data blocks on different clusters by copy the same pieces of information into multiple clusters on different racks. This will help to achieve the following:

- ***Fault Tolerance***: recover data block in case of cluster failure or Rack failure.
- ***High Concurrency***: available same piece of data to be processed by multiple clients at the same time.



FEATURES OR REQUIREMENT OF DISTRIBUTED FILE SYSTEM (DFS)

1.Transparency :

- **Structure transparency –**
  There is no need for the client to know about the number or locations of file servers and the storage devices.
- **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.

2.Performance :
Performance is measured as the average amount of time needed to satisfy client requests. This time includes CPU time + time for accessing secondary storage + network access time. It is desirable that the performance of a distributed file system be comparable to that of a centralized file system

3. 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.

4. 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.

5. Scalability: Growth of nodes and users should not seriously disrupt service.

6. Security:
Users should be confident of the privacy of their data.

7. Heterogeneity:
There should be easy access to shared data on diverse platforms (e.g. Unix workstation, Wintel platform etc)

ADVANTAGES OF DISTRIBUTED FILE SYSTEM

Distributed file system provides the following main advantages:

1. DFS allows multiple user to access or store the data.
2. **Scalability**: You can scale up your infrastructure by adding more racks or clusters to your system.
3. **Availability:** Data replication will help to achieve fault tolerance. It improved the availability of file, access time and network efficiency.
4. **High Concurrency**: utilize the compute power of each node to handle multiple client requests (in a parallel way) at the same time.
5. Improved the capacity to change the size of the data and also improves the ability to exchange the data.
6. Distributed File System provides transparency of data even if server or disk fails

DISADVANTAGES OF DISTRIBUTED FILE SYSTEM

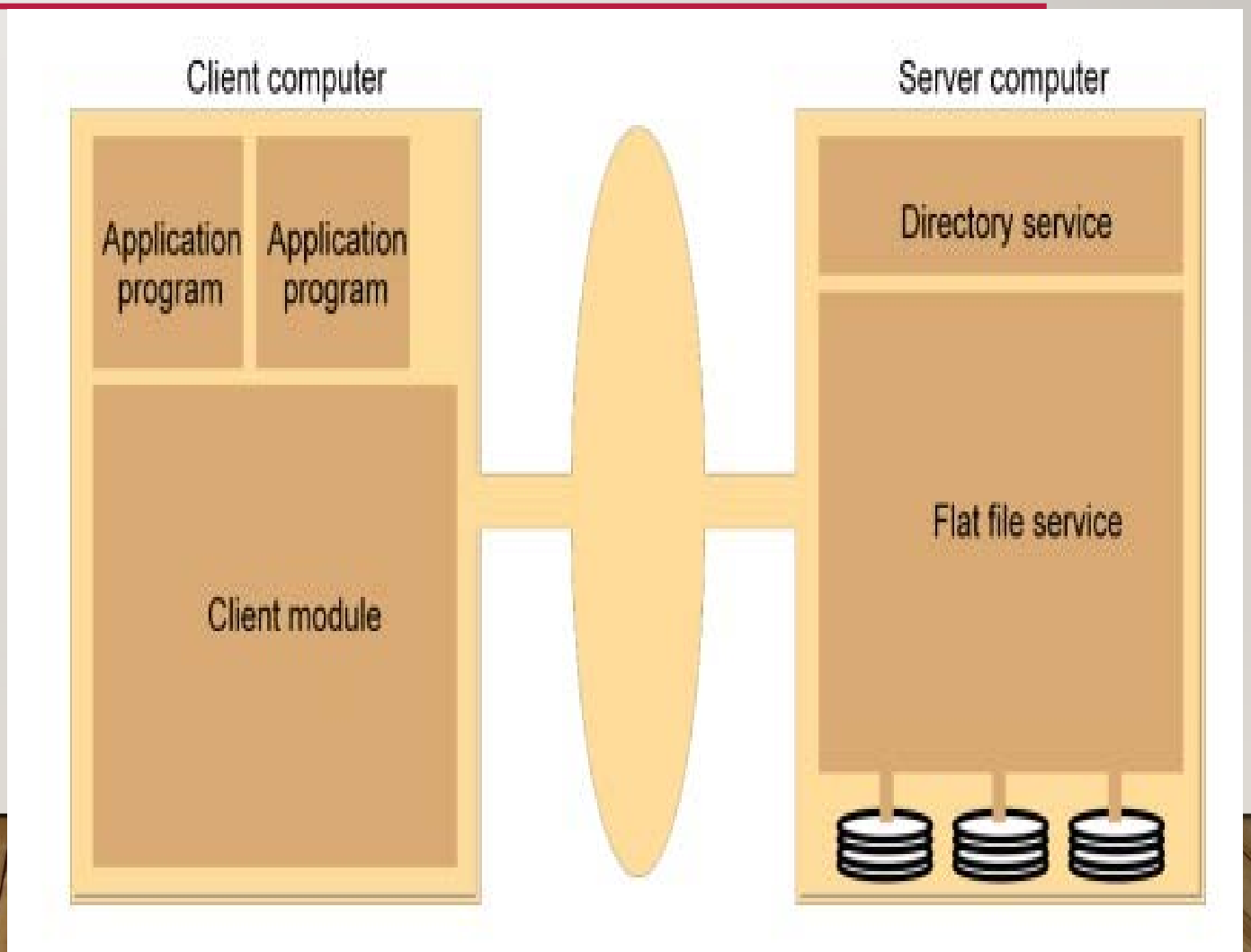
1. In Distributed File System nodes and connections needs to be secured therefore we can say that security is at stake.
2. There is a possibility of lose of messages and data in the network while movement from one node to another.
3. Database connection in case of Distributed File System is complicated.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

4.2 FILE SERVICE ARCHITECTURE

- An architecture that offers a clear separation of the main concern in providing access to files is obtained by structuring the file service as three component.

1. Flat file service
2. Directory service
3. Client module

figure : File Service Architecture



- Responsibilities of various modules can be defined as follows :

1. Flat file service

- A flat file service is concerned with implementing operations on the contents of file.
- **Unique File Identifiers (UFIDs**) are used to refer to files in all requests for flat file service operations . When the flat file service receives a request to create a file , it generates a new UFID for it and returns the UFID to the requester.

FLAT FILE SERVICE OPERATIONS

Create ()	Creates a new file of length() and delivers a UFID for it.
Read (FileId , i , n)	Reads a sequence of up to n items from a file starting at item i.
Write (FileId , i , Data)	Write a sequence of Data to a file , starting at item i.
Delete (FileId)	Removes the file from the file store.
GetAttributes(FileId)	Returns the file attributes for the file.
SetAttributes(FileId , Attr)	Sets the file attributes

2. DIRECTORY SERVICE

- A directory service provides a mapping between text names for files and their UFIDs(**Unique File Identifiers**).
- Clients may obtain the UFID of a file by quoting its text name to the directory service.
- Directory service supports functions to add new files to directories.

DIRECTORY SERVICE OPERATIONS

Lookup (Dir , Name)	Locates the text name in the directory and returns the relevant UFID. If Name is not in the directory , throws an exception.
AddName (Dir, Name , File)	If Name is not in the directory , adds (Name , File) to the directory and updates the file ‘s attribute record. If Name is already in the directory : throws an exception.
UnName (Dir , Name)	If Name is in the directory , the entry containing Name is removed from the directory. If Name is not in the directory , throws an exception.
GetNames (Dir , Pattern)	Returns all the text names in the directory that match the regular expression pattern.

3. CLIENT MODULE

- A client module runs on each client computer , integrating and extending the operations of a flat file service and directory service under a single application programming interface(API) that is available to user level programs in the client computer.
- It holds information about the network locations of flat-file and directory server processes.

4.3 INTRODUCTION TO NAME SERVICE

- In a distributed system names are used to refer to a wide variety of resources such as computers, services, remote objects, and files as well as users.
- Users cannot communicate with one another through a distributed system unless they cannot name one another i.e name facilitate communication and resource sharing
- Names are used for identification as well as for describing attributes.
- For many purposes, names are preferable to identifiers because they are more meaningful to users
- Names (e.g. URLs) are bound to objects (e.g. web pages). Names must be resolved before the corresponding objects can be invoked. The use of addresses as names is deprecated because it prevents the relocation of the named objects.
- File names , URLs ,domain names

NAME, ADDRESS AND OTHER ATTRIBUTES

- Any process that requires access to a specific resource must posses a name or identifier for it. Example : URL http::://www.cdk3.net
- A name has to be looked up before it can be used.
- A name is said to be resolved when it is translated into data about the named resource or object ,

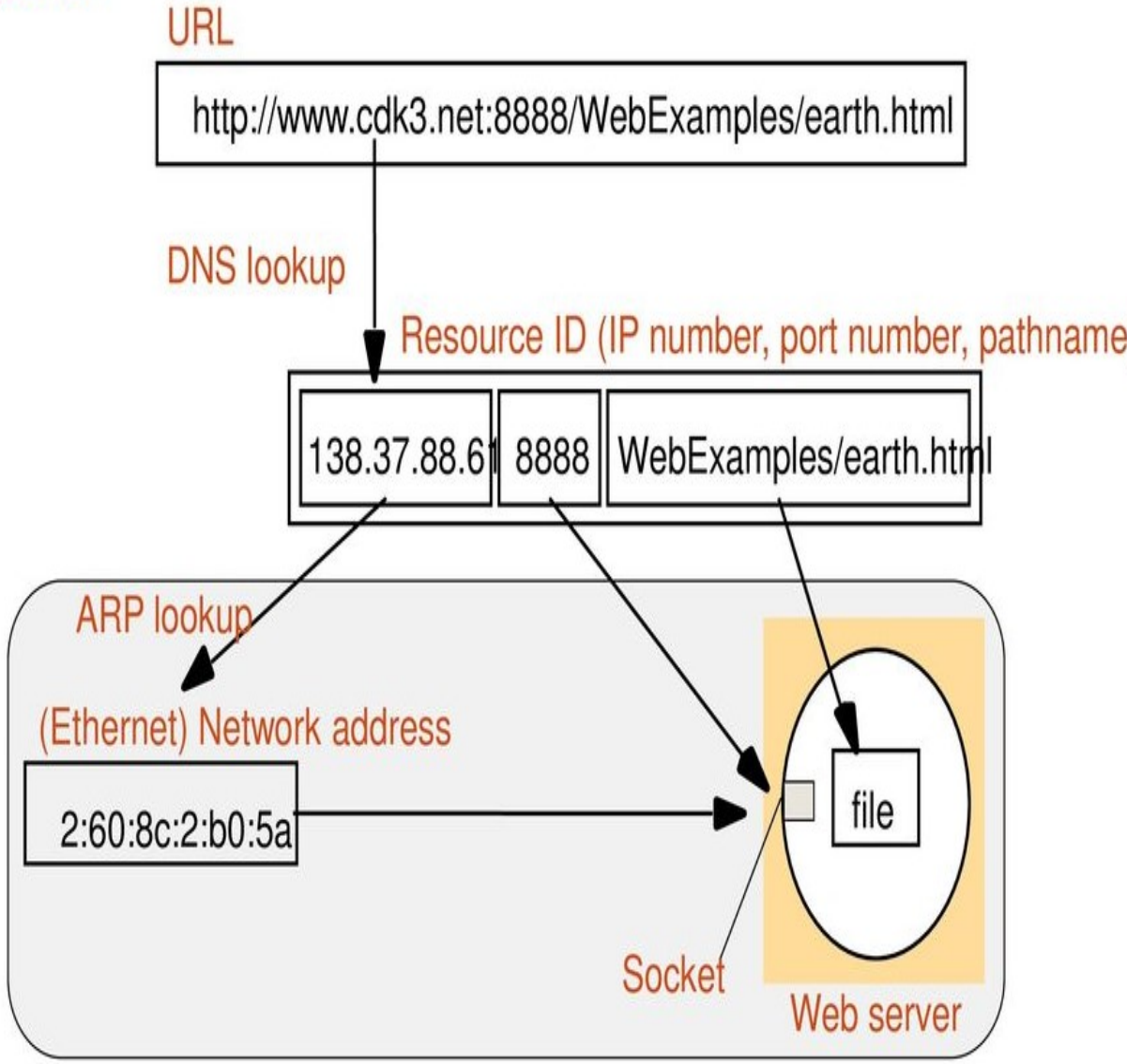
NAMES AND BINDING

- Names are bound to the attributes of named objects(and not to any specific implementations)
- The association between a name and an object is called a binding.
- Services are written to map between names and the attributes of object they refer to . The Domain Name System maps domain names to the attributes of a host computer i.e. its IP address .
- Example: domain name → Domain Name Service(DNS) maps → attributes of the host computer

www.imm.dtu.dk (name) is bound to the IP address "192.38..82.230" of the IMM webserver (resource

Composed naming domains used to access a resource from a URL

Figure 9.1



NAME SERVICE

- A name service stores a collection of one or more naming contexts.
- Naming context is the set of bindings between textual names and attributes for objects.
- Provides a general naming scheme for entities (such as users and services) that are beyond the scope of a single service.
- Major operation: resolve a name - to look up attributes from a given name
- Other operations required: creating new binding, deleting bindings, listing bound names and adding and deleting contexts.

GENERAL NAME SERVICE REQUIREMENTS

- Handle arbitrary number of names and to serve arbitrary number of administrative organizations.
- A long lifetime
- High availability
- Fault isolation
- Tolerance of mistrust

4.4 NAME SERVICES AND DOMAIN NAME SYSTEM

- The domain Name System is a name service design whose main naming database is used across internet.
- Performs name-to-IP mapping.
- The internet DNS is partitioned both organizationally and geographically.
- Com – commercial organization
- Edu- universities and other educational institutions

- The Domain Name System (DNS) is an important part of the internet, providing a way to map names (a website you're seeking) to numbers (the address for the website).
- The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.
- Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1 (in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1::c629:d7a2 (in IPv6).

4.5 GOOGLE FILE SYSTEM

- Google file system is a scalable distributed file system developed by Google to provide efficient and reliable access to data using large clusters of commodity hardware .
- It is designed to meet the rapidly growing demand of Google's data processing need.
- It provides performance, scalability, reliability and availability of data across distributed system for handling and processing big data.

CHARACTERISTICS OF GFS

1. Files are organized hierarchically in directories and identified by path name.
2. It supports all the general operations on files like read, write , open ,delete and so on.
3. It provides atomic append operation known as record append.
4. It performs two operations : snapshot and record append.

WHY BUILT GFS?

1. Node failures happen frequently.
2. Files are huge
3. Most files are modified by appending at the end
4. High sustained bandwidth is important than low latency

Common goals of GFS

1. Performance,
2. Scalability,
3. Reliability
4. Availability
5. Fault Tolerance

Limitations of GFS

1. Not optimal for small sized files.
2. It has high storage overheads.
3. It needs specialization for record append operation.

4.6 COMPARISON OF DIFFERENT DISTRIBUTED FILE SYSTEM

1. **NFS** stands for **Network File System**. It is a client-server architecture that allows a computer user to view, store, and update files remotely. NFS has stateless servers.

2 . **AFS** stands for **Andrew File System** and it has stateful servers . Stateful servers in AFS allow the server to inform all clients with open files about any updates made to that file by another client, through what is known as a callback. AFS is a distributed file system which uses a set of trusted servers to present a homogeneous, location-transparent file name space to all the client workstations.

3. **Sprite** use data caching model but employs the remote service model under certain circumstances.

4. **Locus** and NFS use the remote service model but add caching for better performance.

Accessing Remote Files

a. Remote Service Model

- Client's request processed at server's node.
- In this case packing and communication overhead can be significant.

b. Data caching model

- Client's request processed on the client's node itself by using the cached data.
- This model greatly reduces network traffic.
- Cache consistency problem may occur.

5. The **Apollo** DOMAIN system is a fully operational distributed computing environment for a network of personal workstations and network servers.

Its distributed system focus was on a file system that provided users of autonomous workstations with the same ease of file sharing.

6. **Remote file sharing (RFS)** is a type of distributed file system technology that enables file and/or data access to multiple remote users over the Internet or a network connection.

RFS is also known as a general process of providing remote user access to locally stored files and/or data.

•

File system	Cache size and location	Writing policy	Consistency guarantees	Cache validation
NFS	Fixed , memory	On close or 30 sec ,delay	Sequential	On open , with server consent
AFS	Fixed , disk	On close	Sequential	When modified server asks client
Sprite	Variable , memory	30 sec , delay	Sequential , Concurrent	On open , with server consent
Locus	Fixed , memory	On close	Sequential, Concurrent	On open , with server consent
Apollo	Variable , memory	Delayed or on unlock	Sequential	On open , with server consent
RFS	Fixed , memory	Write-through	Sequential, Concurrent	On open , with server consent