s6. List out the transparencies in file system.

A distributed file system is to present certain degrees of transparency to the user and the system:

Access transparency: Clients are unaware that files are distributed and can access them in the same way as local files are accessed.

Location transparency: A consistent name space exists encompassing local as well as remote files. The name of a file does not give it location.

Concurrency transparency: All clients have the same view of the state of the file system. This means that if one process is modifying a file, any other processes on the same system or remote systems that are accessing the files will see the modifications in a coherent manner.

Failure transparency: The client and client programs should operate correctly after a server failure.

Heterogeneity: File service should be provided across different hardware and operating system platforms.

Scalability: The file system should work well in small environments (1 machine, a dozen machines) and also scale gracefully to huge ones (hundreds through tens of thousands of systems).

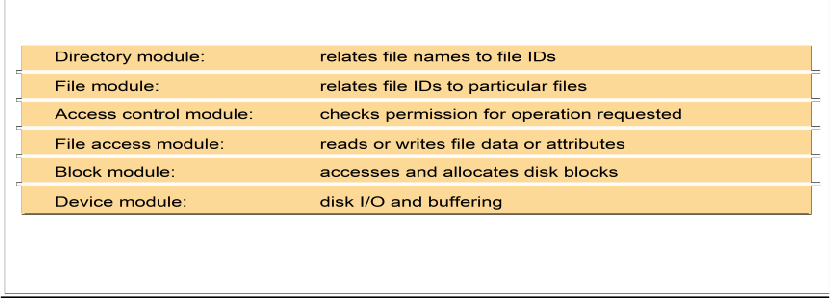
Replication transparency: To support scalability, we may wish to replicate files across multiple servers. Clients should be unaware of this.

Migration transparency: Files should be able to move around without the client's knowledge.

Support fine-grained distribution of data: To optimize performance, we may wish to locate individual objects near the processes that use them.

Tolerance for network partitioning: The entire network or certain segments of it may be unavailable to a client during certain periods (e.g. disconnected operation of a laptop). The file system should be tolerant of this.

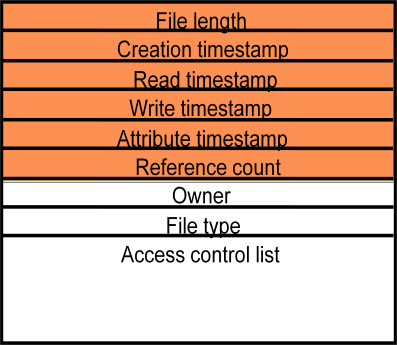
4. List out file system modules.



5. Sketch the file attributes and record structure.

File systems are responsible for the organization, storage, retrieval, naming, sharing and protection of files.

Files contain both data and attributes



10. Discuss the distributed file system design requirements.

Distributed File system requirements

 Related requirements in distributed file systems are:

 Transparency

 Concurrency

 Replication

 Heterogeneity

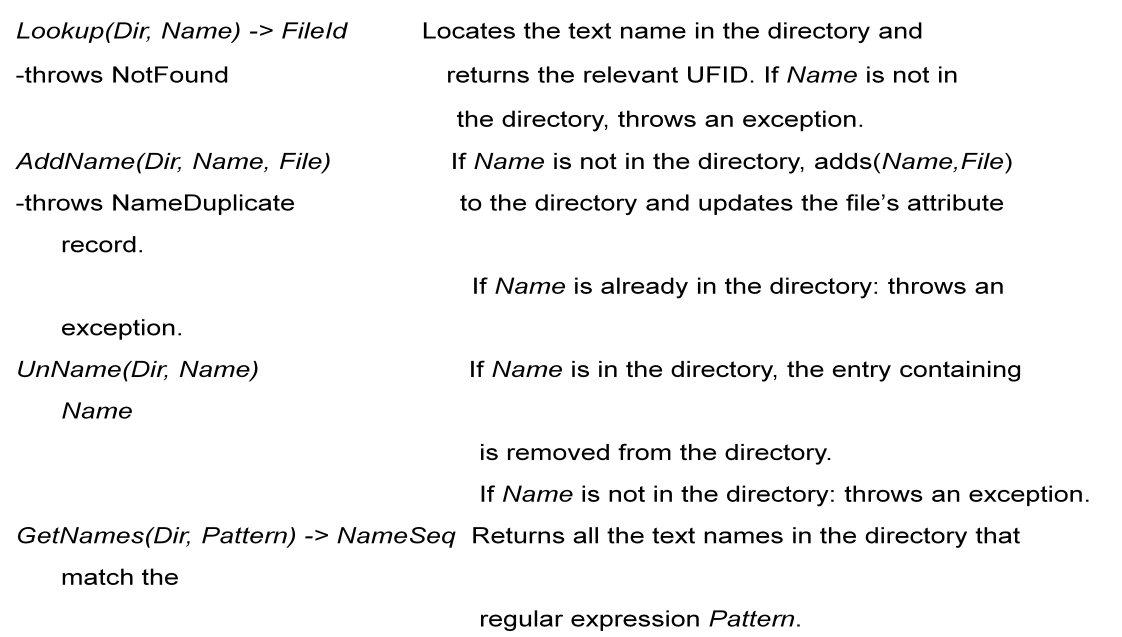
 Fault tolerance

 Consistency

 Security

 Efficiency

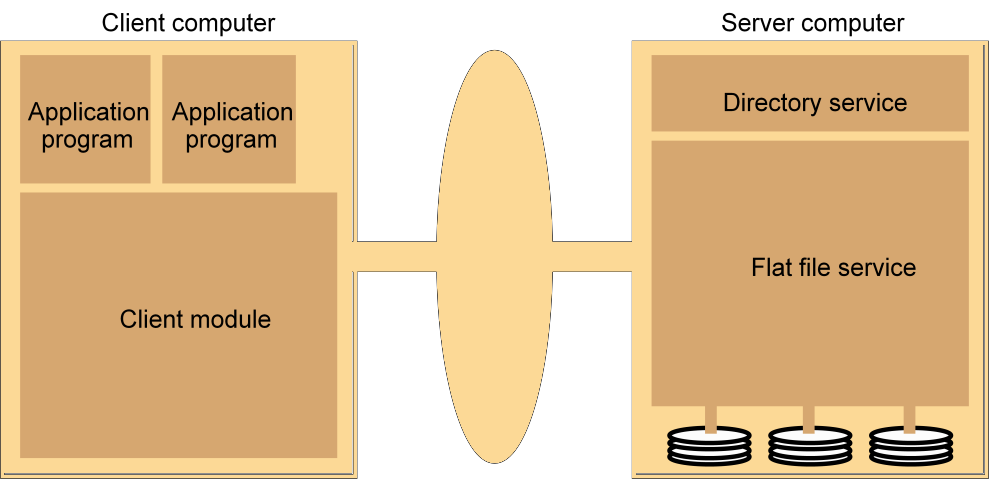
7. List the directory service operation.



With a neat diagram explain the components of file service architecture in

brief w. r .t. following; i) Flat File Service ii) Directory Service

Iii) Client Module



Flat file service:

 Concerned with the implementation of operations on the contents of file.

Unique File Identifiers (UFIDs) are used to refer to files in all requests

for

flat file service operations. UFIDs are long sequences of bits chosen

so that each file has a unique among all of the files in a distributed

system.

 Directory service:

 Provides mapping between text names for the files and their UFIDs.

Clients may obtain the UFID of a file by quoting its text name to

directory service. Directory service supports functions needed generate

directories, to add new files to directories.

 Client module:

 It runs on each computer and provides integrated service (flat file and

directory) as a single API to application programs. For example, in

UNIX hosts, a client module emulates the full set of Unix file

operations.

It holds information about the network locations of flat-file and

directory server processes; and achieve better performance through

implementation of a cache of recently used file blocks at the client.



2. Discuss model architecture of distributed file system and its components.\

same as 3

9. Describe the characteristics of file system

**Transparency**

Transparency refers to hiding details from a user. The following types of transparency are desirable.

i. Structure transparency: Multiple file servers are used to provide better performance, scalability, and reliability. The multiplicity of file servers should be transparent to the client of a distributed file system. Clients should not know the number or locations of file servers or the storage devices instead it should look like a conventional file system offered by a centralized, time sharing operating system.

ii. Access transparency: Local and remote files should be accessible in the same way. The file system should automatically locate an accessed file and transport it to the client’s site.

iii. Naming transparency: The name of the file should not reveal the location of the file. The name of the file must not be changed while moving from one node to another.

iv. Replication transparency: The existence of multiple copies and their locations should be hidden from the clients where files are replicated on multiple nodes.

**2. User mobility**

The user should not be forced to work on a specific node but should have the flexibility to work on different nodes at different times. This can be achieved by automatically bringing the users environment to the node where the user logs in.

**3. Performance**

Performance is measured as the average amount of time needed to satisfy client requests, which includes CPU time plus the time for accessing secondary storage along with network access time. Explicit file placement decisions should not be needed to increase the performance of a distributed file system.

**4. Simplicity and ease of use**

**5. User interface to the file system be simple and number of commands should be as small as possible. A DFS should be able to support the whole range of applications.**

**6. Scalability**

A good DFS should cope with an increase of nodes and not cause any disruption of service. Scalability also includes the system to withstand high service load, accommodate growth of users and integration of resources.

**7. High availability**

A distributed file system should continue to function even in partial failures such as a link failure, a node failure, or a storage device crash. Replicating files at multiple servers can help achieve availability.

**8. High reliability**

Probability of loss of stored data should be minimized. System should automatically generate backup copies of critical files in event of loss.

**9. Data integrity**

Concurrent access requests from multiple users who are competing to access the file must be properly synchronized by the use of some form of concurrency control mechanism. Atomic transactions can also be provided to users by a file system for data integrity.

**10. Security**

A distributed file system must secure data so that its users are confident of their privacy. File system should implement mechanisms to protect data that is stored within.

**11. Heterogeneity**

Distributed file system should allow various types of workstations to participate in sharing files via distributed file system. Integration of a new type of workstation or storage media should be designed by a DFS.

Explain the name service in detail.

In a Distributed System, a Naming Service is a specific service whose aim is to provide a

consistent and uniform naming of resources, thus allowing other programs or services to localize

them and obtain the required metadata for interacting with them.

Key benefits

– Resource localization

– Uniform naming

– Device independent address (e.g., you can move domain name/web site from one

server to another server seamlessly).

The role of names and name services

 Resources are accessed using identifier or reference–

–





An identifier can be stored in variables and retrieved from tables quickly

Identifier includes or can be transformed to an address for an object

 E.g. NFS file handle, Corba remote object reference

– A name is human-readable value (usually a string) that can be resolved to an

identifier or address

 Internet domain name, file pathname, process number

 E.g ./etc/passwd, http://www.cdk3.net/

For many purposes, names are preferable to identifiers

– because the binding of the named resource to a physical location is deferred and

can be changed

– because they are more meaningful to users

Resource names are resolved by name services

– to give identifiers and other useful attributes