Distributed File Systems

G.K.PARINITHA

File Model

- Different File systems use different conceptual models.
- Unstructured and structured files.
- Mutable and Immutable files.

Unstructured and Structured Files

- In the unstructured model, file is an unstructured sequence of data. There is no substructure known to the file server. File content is also an uninterpreted sequence of bytes

- The interpretation of the meaning and structure of the data stored in the files is up to the application.
- Kernel does not interpret the content or structure of the file.

Unstructured and Structured Files

- In structured files, the file appears to the file server as an ordered sequence of records. Records of different files of the same file system can be of different sizes.
- Record is the smallest unit in structured file. Read and write operation is performed on set of records.

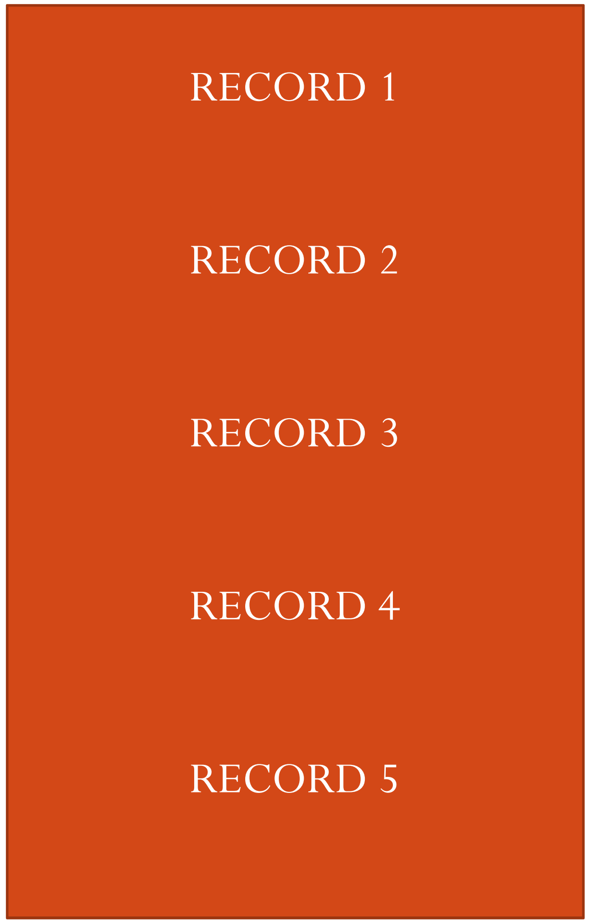
Unstructured and Structured Files

- Structured files are of two types: index record and non-index record file.
- Index Record file: Records have one or more key fields. Key is used to access the records. File is maintained by B-tree or any other suitable data structures like hash tables and indices.

- Non-indexed record file: File record is accessed by specifying its position within the file

Unstructured and Structured Files

Structured file format



Mutable and Immutable Files

- Mutable and Immutable files are based on the modifiability criteria.

- A mutable file can be updated or extended. This means you can change, add or remove elements of a collection.
- Most existing operating systems use the mutable file model. An update performed on a file overwrites its old contents to produce the new contents.

Mutable and Immutable Files

- An immutable file is one that, once created, cannot be changed.
- Immutable files are easy to cache and to replicate across servers since their contents are guaranteed to remain unchanged.

- In the immutable model, rather than updating the same file, a new version of file is created each time a change is made to the file contents and the old version is retained unchanged.

Mutable and Immutable Files

- Immutable file model eliminates all problems associated with mutable versions. It suffers from two problems:
- Increased use of disk space
- Increased disk allocation activity

File Accessing Model

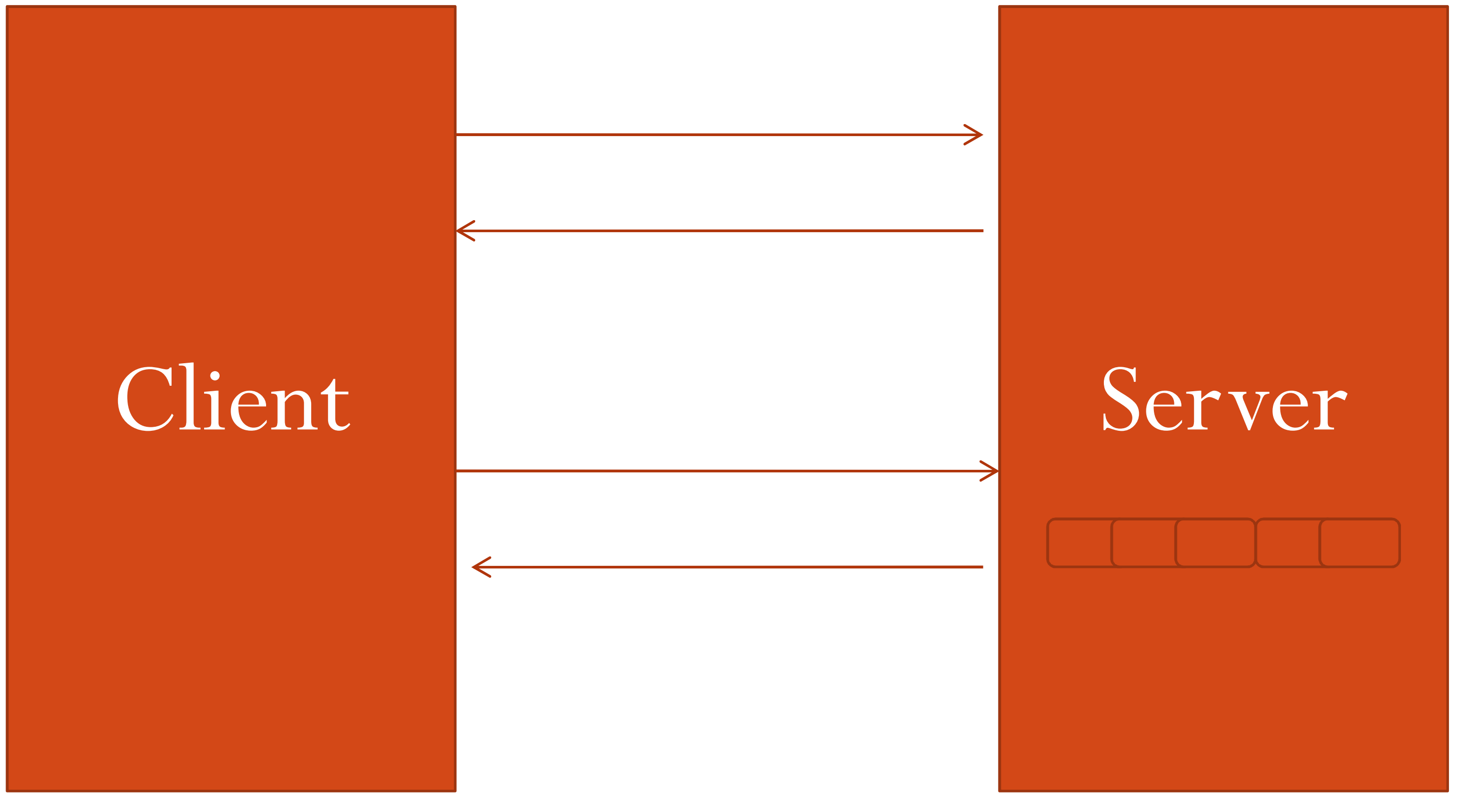
- ACCESSING REMOTE FILES:
- Accessing remote files are of two types
    - Remote service model
    - Data caching model

- In Remote service model, the client submits requests to server; all processing is done on server. File never moves from server. Problem is server bottleneck

File Accessing Model

- In Data Caching Model, it uses locality feature to reduce network traffic.
- For open file operation, transfer entire file to client and for close operation, it transfers entire file to server.
- Client works on the file locally.

File Accessing Model



Request from client to access remote file

File stays on the server

File Accessing Model

- This takes advantage of the locality feature of the found in file accesses.
- A replacement policy such as LRU is used to keep the cache size bounded.
- It is simple and efficient if working on entire file.
- Problem is it needs local disk space.

Unit of Data Transfer

- Transfer levels are file, block, byte and record.
- The unit of data transfer when a request is satisfied by a server can be:
    - A whole file
    - A number of block of a file
    - A specific number of bytes
    - A number of records.

Transfer Level : File

- Whole File is moved to the client side or server side before performing operation.
- It is simple and involves less communication overhead
- Immune to server.
- A client required to have large storage space.
- If small fraction of a file is required, moving whole file is wasteful.

Transfer Level :Block

- File transfer between client and server takes place in blocks.
- Also called page level transfer model.
- A client not required to have large storage.
- Used in diskless workstation
- No need for copying entire file.
- Network traffic overhead.

Transfer Level :Byte

- File transfer between client and server takes place in bytes
- Flexibility maximized
- Difficult cache management to handle the variable length data

Transfer Level : Record

- File transfer between client and server takes place in records.
- Handling structured and indexed files.
- More network traffic.
- More overhead to re-construct a file

File Sharing Semantics

- Files are shared between number of users.
- File protection, naming and sharing is issue for file sharing.
- Directory structure may allow files to be shared by users.
- Sharing must be done through a protection scheme.
- Consistency semantics is related with file sharing on the network. If there is a difference in the content of file, then it creates a problem.

File Sharing Semantics

- Unix Semantics:
- It enforces an absolute time ordering on all operations and ensures that every read operation on a file sees the effects of all previous write operations
- It implements:
  - Writes to an open file visible to other users of the same open file
  - Sharing file pointer to allow multiple users of the same open file

File Sharing Semantics

- A file is coupled with a single physical image that is associated as special resource.
- If there is a conflict for single then it causes delays in user processes.
- Centralized systems use UNIX semantics.

File Sharing Semantics

- Session Semantics:
- Andrew File systems implemented complex remote file sharing semantics:
  - Writes to a file by an user is not visible to other users.
  - Once the file is closed, the changes are visible only to new sessions.

File Sharing Semantics

- In this semantics, a file can be associated with multiple views.
- Almost no constraints are imposed on scheduling accesses.
- No user is delayed in reading or writing the personal copy of the file
- AFS file systems may be accessible by systems around the world.
- Access control is maintained through complicated access control lists, which may grant access to the entire world or to specifically named remote environments.

File Sharing Semantics

- Immutable shared file semantics:
- Under this system, when a file is declared as shared by its creator, it becomes immutable and the name cannot be re-used for any other resource.
- Hence it becomes read-only, and shared access is simple
- Once a file is declared as shared by its creator, it cannot be modified.
- An immutable file has two key properties. Its name may not be reused and its contents may not be altered.

File Sharing Semantics

- Transaction-like semantics:
- All changes occur atomically. Begin transaction, perform operations and end transaction.
- Partial modifications made to the shared data by a transaction will not be visible to other concurrently executing transactions until the transaction ends.
- The final file content is the same as if all the transactions were run in some sequential order.

THANK YOU