Distributed File Systems

Overview

- What is Distributed File System (DFS)?
- DFS Architecture
- DFS Requirements
- The Google File System (GFS)
- References

What is Distributed File System(DFS) ?

- Distributed File System: A file system that joins together the file systems of individual machines in network. Files are stored (distributed) on different machines in a computer network but are accessible from all machines.

- • Distributed file systems are also called network file systems

Clients should view a DFS the same way they would a centralized FS; the distribution is hidden at a lower level.

Performance is concerned with throughput and response time.

cont...

- In DFS more than one client may access the same data simultaneously, the server must have a mechanism in place to organize updates so that the client always receives the most current version of data and that data conflicts do not arise.

- DFS typically use file or database replication (distributing copies of data on multiple servers) to protect against data access failures.

- Files which are distributed across multiple servers appear to users as if they reside in one place on the network. Users no longer need to know and specify the actual physical location of files in order to access them.

DFS Architecture

Different DFS architecture exists

1. Client- Server Architecture : Sun Microsystem's Network System) which provides standardized view of its local file system.

2. Cluster-Based Distributed File : System such as (Google file system)GFS. It consists of a Single master along with multiple chunk servers and divided into multiple chunks.

3. Symmetric Architecture : Based on peer-to-peer technology. In this file system, the clients also host the metadata manager code,resulting in all nodes understanding the disk structures.

***Meta data** is a set of data that describes and gives information about other data*

cont...

4. Asymmetric Architecture : There are one or more dedicated metadata managers that maintain the file system and its associated disk structures. Example is traditional NFS file systems.

5. Parallel Architecture : Here, data blocks are striped, in parallel, across multiple storage devices on multiple storage servers. Support for concurrent read and write capabilities.

DFS: Communication

DFS's use Remote Procedure Call method to communicate as they make the system independent from underlying OS, networks and transport protocols

- In RPC approach, there are two communication protocols to consider, TCP and UDP.
- TCP is mostly used by all DFS's.
- UDP is considered for improving performance .

Distributed File system requirements

- Related requirements in distributed file systems are:
  - Transparency (Access, location, mobility, performance and scaling)
  - Concurrency
  - Replication(for load balancing and fault tolerance)
  - Heterogeneity (different hardware and operating systems)
  - Fault tolerance
  - Consistency
  - Security
  - Efficiency

Transparency (1)

- **Naming :** is the mapping between logical and physical objects. Example: A user filename maps to <cylinder, sector>.
- In a conventional file system, it's understood where the file actually resides; the system and disk are known.
- In a **transparent** DFS, the location of a file, somewhere in the network, is hidden.

Transparency (2)

- The currently common approach employs -
- 1. Central metadata server to manage file name space. Therefore decoupling metadata and data improve the file namespace and relief the synchronization problem.
- 2. Metadata distributed in all nodes resulting in all nodes understanding the disk structure.

cont...

- Location transparency –
-
- The name of a file does not reveal any hint of the file's physical storage location.
- File name still denotes a specific, although hidden, set of physical disk blocks.
- This is a convenient way to share data.

cont...

- **Location independence** –
- The name of a file doesn't need to be changed when the file's physical storage location changes. Dynamic, one-to-many mapping.
- Better file abstraction.
- Promotes sharing the storage space itself.
- Separates the naming hierarchy from the storage devices hierarchy.

Most DFSs today:

Support location transparent systems.

- Do NOT support **migration**; (automatic movement of a file from machine to machine.)
- Files are permanently associated with specific disk blocks.

Consistency

Most of DFS employ checksum to validate the data after sending through communication network.

- Caching and Replication play an important role in DFS when they are designed to operate over wide area network.
- It can be done in many ways such as Client-side caching and Server-Side replication.
- There are two types of data need to be considered for replication: metadata replication and data object replication.

Cont...

- Accessing Remote Files

Two choices:

- Remote File Access – operations on remote files and directories are always sent to remote machine
- Caching – remote files and directories are cached on client

Cont...

- Reduce network traffic by retaining recently accessed disk blocks in a cache, so that repeated accesses to the same information can be handled locally.
- If needed data not already cached, a copy of data is brought from the server to the user.
- Accesses are performed on the cached copy.
- Files identified with one master copy residing at the server machine, but copies of the file (or parts of the file) are scattered in different caches.
- **Cache-consistency problem** – keeping the cached copies consistent with the master file

Cont...

Comparing Caching and Remote Service

- In caching, many remote accesses handled efficiently by the local cache; most remote accesses will be served as fast as local ones.
- Servers are contracted only occasionally in caching (rather than for each access).

-Reduces server load and network traffic.

-Enhances potential for scalability.

- Remote server method handles every remote access across the network; penalty in network traffic, server load, and performance.
- Total network overhead in transmitting big chunks of data (caching) is lower than a series of responses to specific requests (remote-service)

Cont...

Issues in File Caching

- Cache Location – on disk or main memory
- (Complete) File caching vs block caching
- Cache consistency
- Stateful vs Stateless servers

Cont...

Cache Location Disk vs. Main Memory

Advantages of disk caches

- More reliable.
- Cached data kept on disk are still there during recovery and don’t need to be fetched again.

Advantages of main-memory caches:

- Permit workstations to be diskless.
- Data can be accessed more quickly
- Server caches (used to speed up disk I/O) are in main memory regardless of where user caches are located; using main-memory caches on the user
- machine permits a single caching mechanism for servers and users.

Cont...

Cache Update Policy

- Write-through – write data through to disk as soon as they are placed on any cache. Reliable, but poor performance.
- Delayed-write – modifications written to the cache and then written through to the server later. Write accesses complete quickly; some data may be overwritten before they are written back, and so need never be written at all. Poor reliability; unwritten data will be lost whenever a user machine crashes.

Cont...

- A stateful file server remembers client data (state) from one request to the next.
- A stateless server keeps no state information
- Using a stateless file server, the client must specify complete file names in each request specify location for reading or writing re-authenticate for each request
- Using a stateful file server, the client can send less data with each request
- A stateful server is simpler

Cont...

Stateless File Server

- Each client request provides complete information needed by the server (i.e., filename etc).
- Avoids state information by making each request self-contained.
- Each request identifies the file and position in the file.
- No need to establish and terminate a connection by open and close operations.

Cont...

Distinctions Between Stateful & Stateless Service

Performance is better for stateful.

- Don't need to parse the filename each time, or "open/close" file on every request.
- Stateful can have a read-ahead cache.

Fault Tolerance: A stateful server loses everything when it crashes.

- Server must ask clients in order to renew its state.
- Client crashes force the server to clean up its encached information.
- Stateless remembers nothing so it can start easily after a crash.

Replication

- Replicas of the same file reside on failure-independent machines.
- Improves availability and can shorten service time.
- Naming scheme maps a replicated file name to a particular replica.
- Existence of replicas should be invisible to higher levels.
- Replicas must be distinguished from one another by different lower-level names.
- Updates – replicas of a file denote the same logical entity, and thus an update to any replica must be reflected on all other replicas.

Security

Authentication Issues and access control are some of the important security issues in DFS's that need to be analyzed.

- Most DFS employ security with authentication, authorization and privacy.

- Some DFS's for specific purposes such as GFS base on the trust between all nodes and clients.

Fault Tolerance

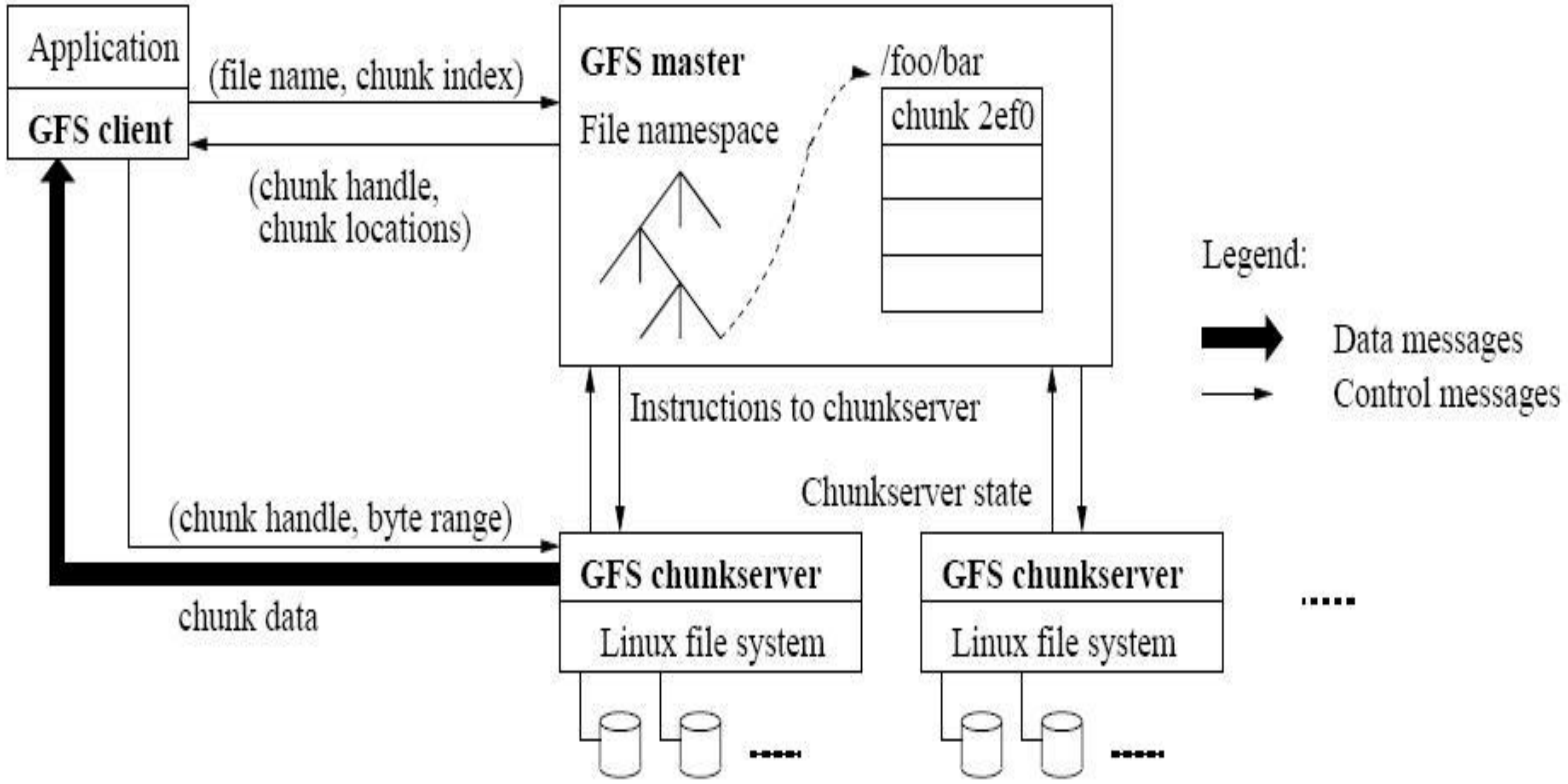
It is very much related to the replication feature because replication is created to provide availability and support transparency of failures to users.

- There are two approaches for fault tolerance : failure as exception and failure as norm.
- Failure as exception systems will isolate the failure node or recover the system from last normal running state.
- Failure as norm systems employ replication of all kind of data.

The Google File System

- A scalable distributed file system for large data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.
- It is widely deployed within Google as the storage platform for the generation and processing of data as well as research and development efforts that require large data sets.
- GFS is optimized for Google's core data storage and usage needs which can generate enormous amounts of data that need to be retained.
- The architecture is cluster based distributed file system.

GFS Architecture



The End.