RESEARCH ARTICLE OPEN ACCESS

# A Review on Google File System

Richa Pandey [1], S.P Sah [2]
Department of Computer Science
Graphic Era Hill University
Uttarakhand – India

## ABSTRACT

Google is an American multinational technology company specializing in Internet-related services and products. A Google file system help in managing the large amount of data which is spread in various databases.

A good Google file system is that which have the capability to handle the fault, the replication of data, make the data efficient, memory storage. The large data which is big data must be in the form that it can be managed easily. Many applications like Gmail, Facebook etc. have the file systems which organize the data in relevant format.

In conclusion the paper introduces new approaches in distributed file system like spreading file's data across storage, single master and appends writes etc.

***Keywords:-*** GFS, NFS, AFS, HDFS

## I. INTRODUCTION

The Google file system is designed in such a way that the data which is spread in database must be saved in a arrange manner.

The arrangement of data is in the way that it can reduce the overhead load on the server, the availability must be increased, throughput should be highly aggregated and much more services to make the available data more accurate and reliable. Many methods are introduced in that process.

## II. GFS EVOLUTION

The need of GFS arises because of the original design of GFS. Mainly the single master design selection was not that much efficient and contains a lot of risk. So Google people decide to research so as to make the master distributed file system to solve existing challenges it faces.

**Some of the problems that Google faced:**

1) Size of storage memory increased in the range of petabytes. The single master started becoming a problem when thousand client requests came simultaneously.

2) 64 MB standard chunk size design choice which was fixed created problems. The system had to deal with applications generating large number of small files e.g.Gmail.

## III. KEY IDEAS

**3.1. Design and Architecture:** GFS cluster consist of single master and multiple chunk servers used by multiple clients. Since files to be stored in GFS are large, processing and transferring such huge files can consume a lot of bandwidth. To efficiently utilize bandwidth files are divided into large 64 MB size chunks which are identified by unique 64-bit chunk handle assigned by master.

**3.2. No caching**: File data is not cached by the client or chunk server.Large streaming reads offer little caching benefits since most of the cache data will always be overwritten.

**3.3. Single Master**: Simplifies design and allows a simple centralized management. Master stores metadata and co-ordinates access. All metadata is stored in master's memory that makes operations fast. It maintains 64 bytes/chunk. Hence, master memory is not a problem. To reduce master involvement lease mechanism is used. Lease is used to maintain a consistent mutation (append or write) order across replicas.

**3.4. Garbage collection**:The system has a special approach for this. Once a file is deleted its data are not regain immediately.Such files are removed if they exist for 3 days during the regular scan. The

advantages offered by it are: 1) Simple in operation 2) Deleting of files can take place during master's idle periods and 3) Safety against accidental deletion.

**3.5.Relaxed consistency model**

1) File namespace transformation are always atomic.
2) File region is consistent if all clients read same values from replicas.
3) File region is defined if clients see mutation writes in entirety.

## IV. GFS FEATURES INCLUDE

- Fault tolerance
- Critical data replication
- Automatic and efficient data recovery.
- High aggregate throughput.
- Reduced client and master interaction because of large chunk server size.
- Namespace management and locking.
- High accessibility.

The largest GFS clusters have more than 1,000 nodes with 300 TB disk storage capacity.
Google file system is a distributed file system built for large distributed data intensive applications like gmail etc.Originally it was built to store data generated by its large crawling and indexing system. The files generated by this system were usually huge. Maintaining and managing such huge files and data processing demands was a challenge with the existing file systems. The main objective of the designers was building a highly fault tolerant system while running inexpensive hardware.

**4.1.GFS design assumptions**:

1) System fail a lot and GFS should be able to recover from it.
2) Files stored are of high GB.
3) Reads of two types: large streaming reads and small random reads.
4) Once files are written they are mostly read.Most of the write operations are of append type.
5) Support concurrent appends by multiple clients to the same file.
6) High supply bandwidth and throughput are more important than low latency.

## V. GENERAL ARCHITECTURE OF GOOGLE FILE SYSTEM

GFS is clusters of computers. A cluster is simply a network of computers. Each cluster might contain hundreds or even thousands of machines. In each GFS clusters there are three main entities:

1. Clients
2. Master servers
3.Chunkservers.

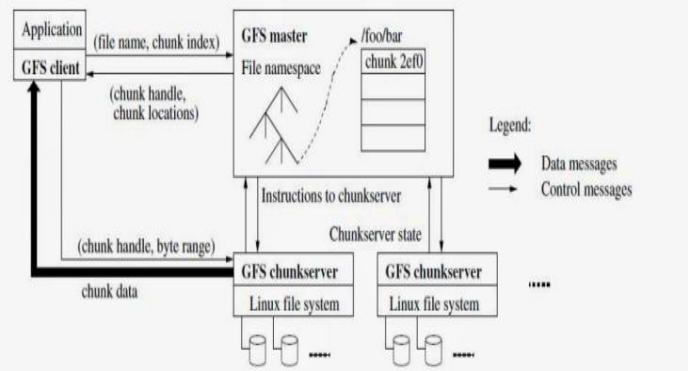


Figure 1 GFS Architecture

1.Client are other computers or computer application which make a file request. Requests can range from retrieving and manipulating existing files to create new files on the system. Clients can be thought as customers of the GFS.

2.Master Server is the manager for the cluster. Its task include:-

(a).Maintaining an operation log, that keeps track of the activities of the cluster. The operation log helps keep service interruptions to a minimum if the master server crashes.

(b) The master server also keeps track of metadata, which is the information that describes chunks. The metadata tells the master server to which files the chunks are related and where they fit in the overall file.

3.Chunk Servers are the powerstation of the GFS. They store 64-MB file chunks. The chunk servers send requested chunks directly to the client. The GFS

copies every chunk multiple times and stores it on different chunk servers. Each copy is called a replica. By default,GFS makes three replicas per chunk, but users can change the setting and make more or fewer replicas as desired.

## VI. COMPARISON

Comparing GFS with other distributed file system like Sun Network file system (NFS) and Andrew File system (AFS) and Hadoop File System(HDFS):

| GFS | NFS | AFS | HDFS |
|---|---|---|---|
| Cluster based architecture | Client-Server based architecture | Cluster based architecture | Cluster based architecture |
| No caching | Client and server caching | Client caching | No caching |
| Not similar to UNIX | Similar to UNIX | Similar to UNIX | Not similar to UNIX |
| End users do not interact. | End users interact | End users interact | End user interact |
| Server replication | No replication | Server replication | Server replication |

## VII. PROS AND CONS

**Pros**:

1) Very high availability and fault tolerance through replication: a) Chunk and master replication and b) Chunk and master recovery.

2) Simple and efficient centralized design with a single master. Delivers good performance for what it was designed for i.e. large sequential reads.

3) Concurrent writes to the same file region are not serializable. Thus replicas might have duplicates but there is no interleaving of records. To ensure data integrity each chunkserver verifies integrity of its own copy using checksums.

4) Read operations takes at least a few 64KB blocks therefore the checksum costs reduces.

5) Batch operations like writing to operation log, garbage collection help increase the bandwidth.

6) Atomic append operations ensures no synchronization is needed at client end.

7) No caching eliminates cache coherence issues.

8) Decoupling of flow of data from flow of control allows to use network efficiently.

9) Orphaned chunks are automatically collected using garbage collection.

10) GFS master constantly monitors each chunkserver through continous messages.

**Cons:**

1) Special purpose design is a limitation when applying to general purpose design.

2) Inefficient for small files.:

i) Small files will have small number of chunks. This can lead to chunk servers storing these files to become special in case of many client requests.

ii) Also if there are many such small files the master involvement will increase and can lead to a problem. Thus,single master node can become an issue.

3) Slow garbage collection can become a problem when the files are not static. If there many deletions then not recycling can become trouble.

4) Since a relaxed consistency model is used clients have to perform consistency checks on their own.

5) Performance can degrade if the numbers of writers and random writes are more.

6) Master memory is a limitation.

7) The whole system is tailored according to workloads present in Google. GFS as well as applications are adjusted and tuned as necessary since both are controlled by Google.

8) No reason is given for the choice of standard chunk size (64MB).

**Future relevance**: GFS is good at for the application it was designed for:i.e. sequential reads for large files by data-parallel workloads. Since HDFS has become sort of an industry standard for storing large amounts of data, it's increasingly being used for other types of workloads. H Base is one example of this (a more database-like column store), which definitely does a lot more random I/Os.

The GFS node cluster is a single master with multiple chunk servers that are continuously accessed by different client systems. Chunk servers store data as Linux files on local disks. Stored data is divided into large chunks (64 MB), which are replicated in the network a minimum of three times. The large chunk size reduces network overhead.

GFS is designed to accommodate Google's large cluster requirements without burdening applications. Files are stored in hierarchical directories which are identified by path names. Metadata - such as namespace, access control data, and mapping information - is controlled by the master, which interacts with and monitors the status updates of each chunk server through timed heartbeat messages.Thus, a more efficient file system must be design which overcomes all the shortcoming of the curent gfs.

## REFERENCES


[1] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung,Google

[2] GFS:Evolution on fast-forward : http://queue.acm.org/detail.cfm?id=1594206

[3] Garth A. Gibson, David F. Nagle, Khalil Amiri, Jeff Butler, Fay W. Chang, Howard Gobioff, Charles Hardin, ErikR iedel, David Rochberg, and Jim Zelenka. A cost-effective, high-bandwidth storage.

[4] Thomas Anderson, Michael Dahlin, Jeanna Neefe, David Patterson, Drew Roselli, and Randolph Wang. Serverless networkfil e systems. In *Proceedings of the 15th ACM Symposium on Operating System Principles*, pages 109–126, Copper Mountain Resort, Colorado, December 1995.

*[5]* Remzi H. Arpaci-Dusseau, Eric Anderson, Noah Treuhaft, David E. Culler, Joseph M. Hellerstein, David Patterson, and Kathy Yelick. Cluster I/O with River: Making the fast case common. In *Proceedings of the Sixth Workshop on Input/Output in Parallel and Distributed Systems (IOPADS '99)*, pages 10–22, Atlanta, Georgia, May 1999.

[6] Luis-Felipe Cabrera and Darrell D. E. Long. Swift: Using distributed disks triping to provide high I/O data rates. *Computer Systems*, 4(4):405–436, 1991.

[7] http://www.users.cselabs.umn.edu/classes/Fall-2012/csci8980-2/papers/gfs.pdf

[8] http://en.wikipedia.org/wiki/Andrew_File_System.