A Comparative Study on Distributed File Systems

Chapter · April 2021

DOI: 10.1007/978-3-030-68291-0_5

2 authors:

Suman De
Birla Institute of Technology and Science Pilani
20 PUBLICATIONS 30 CITATIONS
SEE PROFILE

Megha Panjwani
SAP Labs India
3 PUBLICATIONS 6 CITATIONS
SEE PROFILE

A Comparative Study on Distributed File Systems

Suman De[1*], Megha Panjwani[2]

[1]SAP Labs, Bangalore, Karnataka,
India
suman.de@sap.com

[2]SAP Labs, Bangalore, Karnataka,
India
megha.panjwani@sap.com

Abstract. Distributed File Systems are mainstays for the way massive amounts of data is stored. The emergence of Hadoop File Systems, Google File Systems and Network File Systems have changed the course of how data is managed in servers and has its own implications on Cloud Computing and Big Data management. Each file system offers its own advantages and challenges in terms of performance, fault-tolerance, consistency, scalability and availability. This opens an open debate on how these can be taken up for implementation. The choice of a feature available with each one of them has their own metrices that differentiates them from other file systems.

This paper looks at a comparative study on the file systems and proposes the criterion behind the choice of selection of a specific file system. The study also explores the advantages of using a file system and the benefits and disadvantages associated with them.

Keywords: Distributed File System, Hadoop File System, Google File System, Network File System, OpenAFS

1 Introduction

The need for distributed environments is increasing with the immense need to store, process and analyze data in the fields of aerodynamic research, weather forecasting, scientific applications, banking, etc. A Distributed File System is a client/server architecture-based application that facilitates the access and process of data stored on multiple servers and responds to the client like it does for data stored in a local system. The process is facilitated with the client receiving a copy of the file and the same being cached in the local system. This type of file system organizes files from individual servers into a global directory and hence it appears that the remote access to the file is not location specific but is still identical from the client. The Distributed Files Systems comes with a mechanism to avoid conflicts and try and share the most current version of the data when requested by a client.

[*] No academic titles or descriptions of academic positions should be included in the addresses. The affiliations should consist of the author's institution, town/city, and country

The files are replicated over various sources by means of file or database replication to handle situations of data failure and provide the possibility of data recovery. The requirements to be considered while designing such a DFS ranges from [8]:

- Fault tolerance: Measures how fast the data can be recovered during any failure.
- Hugh file size: A significant number of files can size in GBs. Certain systems process files in chunks that provides a benefit of limiting the amount of data processed by a single function from significant Gigabytes to a few Megabytes.
- Write-once-read-many pattern: Many Distributed File Systems provide optimized functionalities for file write and read operations.
- Role of Metadata: Distributed File Systems allocate a specific vertex as the primary node, that manages the meta information the associated stored files.

The different areas that are to be considered for designing such systems are Transparency, flexibility, reliability, performance, scalability and security, as well as factors of Architecture, Processes, Communication, Naming, Synchronization, Caching & Replication and approaches to handle Fault Tolerance.

2 History of File Systems

A prefilled copyright form is usually Network File System was one of the most popular Distributed File Systems developed by Sun Microsystems in 1985 and was an open, and widely used distributed file system for a long time. It allows mounting all or portion of a file system and can be accessed as per the privileges assigned. Network File System are classified in two types: NFSv3, that has seen its usage for a large period and NFSv4, which was presented in 2003 and consists of multiple upgrades over NFSv3.

NFSv4.1(RFC-5661) was ratified for improving scalability and give better performance. NFv2 and NFSv3 supported the use of UDP protocol and facilitated stateless connections whereas NFSv4 only supported TCP protocol. The stored files are classified as a list of bytes and the system follows a tree-like modelling as a representation that assists the use of hard and symbolic links.

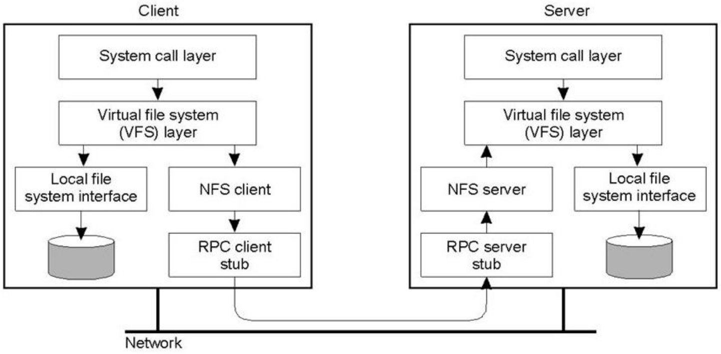

Client
Server
System call layer
Virtual file system (VFS) layer
Local file system interface
NFS client
RPC client stub
System call layer
Virtual file system (VFS) layer
NFS server
Local file system interface
RPC server stub
Network

Fig 1. Network File System Architecture

Andrew File System started as part of a larger project called Andrew. It was developed by the Carnegie Mellon University and was initially called “Vice”. This was primarily designed for systems which run operating systems like BSD, UNIX and Mach and uses a set of trusted servers to provide a homogeneous, geography-independent namespace to all clients. The development of Andrew File System is currently continued through a project known as, OpenAFS. This development works on multiple offerings like Linux, Apple Mac OSX, Sun Solaris and MS Windows NT. [8]

3 Literature Survey

Distributed File Systems, as of now found in the past area, has been the field for various progressions and here, we investigate certain pertinent works done in the theme as of late. In a 2017 paper named, "An Efficient Cache Management Scheme for Accessing Small Files in Distributed File Systems" by Kyuongsoo Bok, Hyunkyo Oh, Jongtae Lim and Jaesoo Yoo, they proposed a disseminated store the executives plot that considers reserve metadata for effective gets of little documents in Hadoop Distributed File Systems (HDFS). The conveyed reserve the executives plot that applied store metadata synchronization with adaptable guideline of correspondence cycle to improve little record get to speed and limit organize load with NameNodes in HDFS. The proposed plot essentially diminished the recurrence of DataNode circle access by keeping up little records that are much of the time utilized by clients in each DataNode reserve and overseeing store metadata on them in the NameNode. Moreover, the proposed plot limited the correspondences with the NameNode by keeping up square metadata and reserve metadata in the customer store and diminished superfluous document gets to by applying a store metadata update strategy. [2]

In a paper identified based on Metadata the in 2019 named, "An Efficient Ring-Based Metadata Management Policy for Large-Scale Distributed File Systems" by Yuanning Gao, Xiaochun Yang, Jiaxi Liu and Guihai Chen, they proposed a novel hashing plan called AngleCut to segment metadata namespace tree and serve huge scope conveyed capacity frameworks. AngleCut first uses a territory protecting hashing (LPH) capacity to extend the namespace tree into straight keyspace, i.e., various Chord-like rings. AngleCut, a proficient and adaptable metadata the executives to parcel metadata namespace tree and serve EB-scale document frameworks. AngleCut first tasks the namespace tree into various Chord-like rings utilizing a novel ring-based region saving capacity. At that point we structure a very much planned history-based distribution procedure to dispense the metadata consistently to MDS's. The two-layer metadata store instrument is proposed to improve the question productivity and decrease the system overhead. To wrap things up, we structure a productive dispersed preparing convention called 2PC-MQ to ensure the consistency of conveyed metadata exchanges. AngleCut jam the metadata territory basically just as keeping up a high burden adjusting degree between MDS's. The hypothetical confirmation and broad examinations on Amazon EC2 displayed the prevalence of AngleCut over the past writing. [1]

4 Google File System

The Google File System was revealed in 2003 to handle the ever-expanding data processing requirements of Google and its applications. It is comprised of groups that contains a considerable lot of storage machines which have been created using cheaper tools and technologies and uses cluster-based model. It has groups made up of a lone master vertex, many block servers and gets connected by different clients. Files are segregated into fixed-size chunks which are grouped using a unique 64-bit chunk handle. Every block is replicated on different block servers to confirm reliability.

It is developed to manage Google’s huge group needs by keeping the existing application functionalities intact. Files are dumped in tree-like structures distinguished by their path names. Meta Information like entry grant requirement, namespace and related data is maintained by the primary responsible, that communicates and observes the progress of every block server via periodic heartbeat notifications.

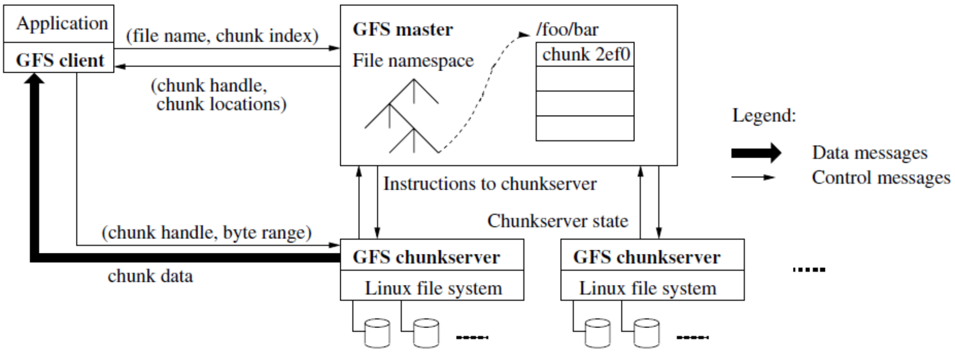


Fig 2. Google File System Architecture

Google File System consists of characteristics like:

- Error tolerance
- Copying mechanism of important data
- Self-reliant data backup and recovery
- Larger productivity
- Less communication of primary and sub-category servers because of block server
- Identification mechanism and authorization scenarios
- Significant presence and lesser downtime

The Google File System considers block copy placement principles that serves two purposes: increase data reliability & availability and enhanced network bandwidth usage. This confirms few copies of a block that is present although a complete rack is hampered or is disconnected. The biggest GFS group has more than 1K vertices with 300 Terabytes of disk storage capacity.

5 Hadoop File System

The Hadoop Distributed File System is an opensource variant of Google File System from Yahoo and was intended to run over hardware equipment to encourage promotion serving and Search Engine Optimization (SEO) prerequisites. The critical distinction of HDFS with different frameworks is that, it is profoundly more fault tolerant when compared to other counterparts. It gives larger productive rights to information and is principally intended for web offerings that comprises of huge information sets.[3] Facebook, eBay, LinkedIn and Twitter are among the web organizations that utilize Hadoop File System to support large information chunks and prerequisites for data analytics' projects.

Hadoop File System group comprises of a singular name vertex which goes about as the ace hub that handles the record scaffolding group convention and refers to the entry point to files provided by users. The client information is put away as records, which are divided in about a single square, and the mentioned squares are put away in a lot of Information Vertices. These perform creation of blocks, cancellation, and replication at whatever point the Name Node commands so.

A Hadoop File System is for the most part conveyed to enormous scope usage and thus, the help for minimal effort product equipment is an especially valuable component. The mentioned systems execute regularly on a GNU/Linux based framework (Operating System). Hadoop File System is made utilizing Java and other systems which underpins Java as a runtime for the Information Vertex or the Data Node offering. Electronic applications utilizing such information concentrated activities can go into several petabytes and many hubs/vertices. Accordingly, must be strong as server disappointments are basic at such scale.

It was used by The New York Times for enormous scope picture transformations, Media6Degrees and Fox Audience Network for log handling, information mining and AI, LiveBet for log stock piling and chances investigation and Joost for meeting. It is the center of many open source information distribution center other options, which are called information/data lakes.

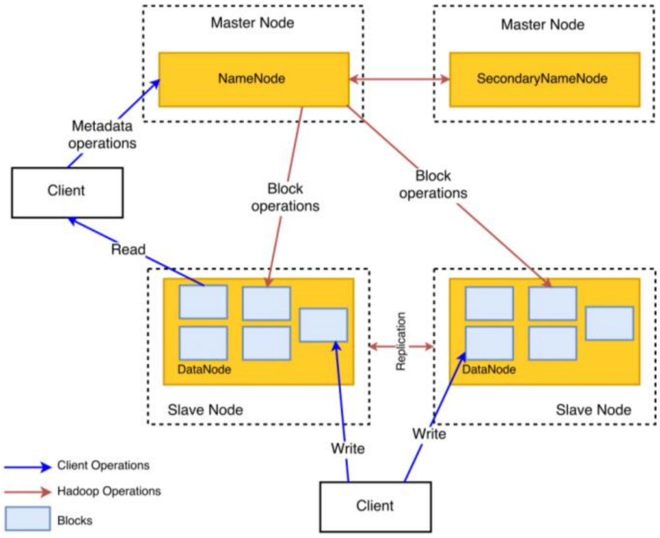


Fig 3. Hadoop File System’s Master Slave Architecture

6 Other File Systems and Services

File systems, like OpenAFS, OpenIO, GlusterFS, MapR FS, Aluxio, have been widely used and we discuss about a few open sourced and proprietary File Systems below.

6.1. OpenAFS

OpenAFS offers a Distributed File System presentation that offers a client-server model with integrated file and copied read-only content distribution, location independence, scalability, security, and migration capabilities. AFS works for a range of diversified systems like UNIX, Linux, MacOS X, and MS Windows. [4]

6.2. OpenIO

OpenIO presents an item stockpiling answer for building hyper-adaptable IT frameworks with a wide scope of applications. OpenIO has local item APIs alongwith SDKs for Python, C, and Java, and coordinates an HTTP REST/API that reserves solid similarity with Amazon S3 and OpenStack’s Swift Application Programming Interfaces. It presents a restrictive File System connector to get to information put away in an OpenIO object-store through file access techniques and depends on Fuse and gives a POSIX File System that can be distributed and managed over nearby systems with the help of NFS, SMB, and FTP. [5]

6.3. Google Cloud Storage

Google Cloud Storage is a RESTful online document web administration for getting to information on Google Cloud Platform framework. The administration consolidates the presentation and adaptability of Google's cloud. It is an Infrastructure as a Service (IaaS), tantamount to Amazon S3 online capacity administration. [7]

6.4. MapR File System (MapR FS)

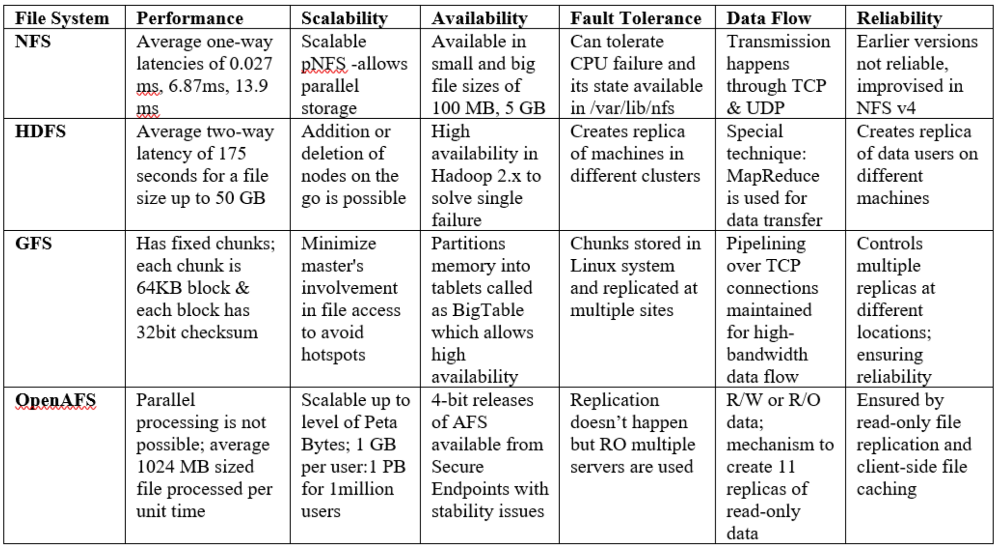
The MapR FS refers to a grouped record framework that enhances an assortment of interfaces including traditional read/compose by means of Network File System and a FUSE interface, for platforms like, Apache Hadoop and Spark. [6] The main characteristics associated with MapR FS are as follows:

- Repeated indexes, no single hub repeats entirety of the meta-information

- Dispersed bunch metadata which courses its action into replication chains
- Productive utilization of B-trees even with extremely huge indexes
- Group apportioning without loss of consistency
- Refresh simultaneously without requiring worldwide locking structures
- Moving overhauls and online filesystem upkeep

7 Comparison

Table 1. Comparison of various File Systems on various parameters and design considerations



File System	Performance	Scalability	Availability	Fault Tolerance	Data Flow	Reliability
NFS	Average one-way latencies of 0.027 ms, 6.87ms, 13.9 ms	Scalable pNFS -allows parallel storage	Available in small and big file sizes of 100 MB, 5 GB	Can tolerate CPU failure and its state available in /var/lib/nfs	Transmission happens through TCP & UDP	Earlier versions not reliable, improvised in NFS v4
HDFS	Average two-way latency of 175 seconds for a file size up to 50 GB	Addition or deletion of nodes on the go is possible	High availability in Hadoop 2.x to solve single failure	Creates replica of machines in different clusters	Special technique: MapReduce is used for data transfer	Creates replica of data users on different machines
GFS	Has fixed chunks; each chunk is 64KB block & each block has 32bit checksum	Minimize master's involvement in file access to avoid hotspots	Partitions memory into tablets called as BigTable which allows high availability	Chunks stored in Linux system and replicated at multiple sites	Pipelining over TCP connections maintained for high-bandwidth data flow	Controls multiple replicas at different locations; ensuring reliability
OpenAFS	Parallel processing is not possible; average 1024 MB sized file processed per unit time	Scalable up to level of Peta Bytes; 1 GB per user:1 PB for 1million users	4-bit releases of AFS available from Secure Endpoints with stability issues	Replication doesn't happen but RO multiple servers are used	R/W or R/O data; mechanism to create 11 replicas of read-only data	Ensured by read-only file replication and client-side file caching

Conclusion

Although there are multiple different file systems being used around the world, we have tried to study and compare the most popular ones. Based on above comparative values, we can conclude that HDFS has most preferred attributes with high performance, availability and a strong file replication strategy against fault tolerance as well. GFS comes next in line as regards scalability and using chunks of data for pipelining transmission over TCP channels. Talking about NFS and OpenAFS, NFS is bit preferred at is an older file system and is thus considered more stable by users but OpenAFS also provides some user-centric features such as scalability up to the levels of Peta Bytes giving 1 GB to each user. Also, OpenAFS is an open source version of the traditional AFS and thus, is being preferred by users in today’s date.

References

1. Y. Gao, X. Gao, X. Yang, J. Liu and G. Chen, "An Efficient Ring-Based Metadata Management Policy for Large-Scale Distributed File Systems," in IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 9, pp. 1962-1974, 1 Sept. 2019, doi: 10.1109/TPDS.2019.2901883
2. Kyoungsoo Bok, Jongtae Lim, Hyunkyo Oh and Jaesoo Yoo, "An efficient cache management scheme for accessing small files in Distributed File Systems," *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jeju, 2017, pp. 151-155, doi: 10.1109/BIGCOMP.2017.7881731
3. M. Nithya and N. U. Maheshwari, "Load rebalancing for Hadoop Distributed File System using distributed hash table," *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, Palladam, 2017, pp. 939-943, doi: 10.1109/ISS1.2017.8389317
4. Available Website: http://www.openafs.org/. Last Accessed: 25th May 2020
5. Available Website: https://www.openio.io/. Last Accessed: 25th May 2020
6. Available Website: http://www.mapr.com/. Last Accessed: 25th May 2020
7. Available Website: cloud.google.com/storage/. Last Accessed: 25th May 2020
8. L.Sudha Rani, K.Sudhakar, S.Vinay Kumar, “Distributed File Systems: A Survey”, *International Journal of Computer Science and Information Technologies,* Vol. 5(3), 2014, 3716-3721