IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-6, Issue-6, June 2018 E-ISSN: 2347-2693

Comparative study of GFS, HDFS and Gluster FS

Anupom Chakrabarty[1*], Chandan Kalita[2]

[1*]Department of Information Technology, Gauhati University, Guwahati, Assam, India
[2]Department of Information Technology, Gauhati University, Guwahati, Assam, India

**Corresponding Author: anupom.chakrabarty@gmail.com, Tel.: +91-9613518605*

Available online at: www.ijcseonline.org

Accepted: 06/Jun/2018, Published: 30/Jun/2018

Abstract— Distributed file system is an essential part of data intensive works as it is used as a primary storage solution. Distributed file system also provides distributed environment of processing for fast and effective processing of large set of data. Over the years, there are a number of DFS has been developed. All the DFS are designed to handle a large set of data efficiently, so that users can access the data quickly regardless of how the data are stored. For a user it becomes hard to choose one against a number of available DFSs. A thorough study about the DFSs will definitely guide users to choose their favorable DFS in their applications. In this paper, we give a brief description about GFS, HDFS and GlusterFS and then compare them on some fundamental issues of DFS such as scalability, transparency and fault tolerant.

Keywords— DFS, GFS, Hadoop, HDFS, GlusterFS, EHA

I. Introduction

Distributed file system is a network attached file system where a number of machines are interconnected through networks. It gives the solution of easy scalability, efficient performance and reliability. It has been used as a solution to variety of applications such as weather forecasting, aerodynamic research, space operations, big data, GIS data analysis etc that have a large amount of data and require distributed environment to process that data. Again with the increasing use of internet, there is a sharp rise in digital data across various internet platforms. The digital data are in the form of search keywords, images, document files, videos, GPS locations etc that do not have any definite structure. So to store, manage and process the structured and unstructured data the concept of big data come into the picture. Big data requires scaling up the storage as the amount of data rapidly increases as well as need to provide low latency for handling that data for any analytical work [1]. Some leading big data practitioners are Google, Facebook, Apple, Microsoft etc. which use hyper scale storage environment.

As the number of distributed file system increases, it becomes important to do a thorough and comparative study of the DFSs to guide the users to choose the best DFS. A distributed file system should be scalable, transparent and fault-tolerant. In this paper, we do a comparative analysis of GFS, HDFS and GlusterFS, to know how they address scalability, transparency, fault tolerant issues.

Google was the first one to face the problem of big data. To handle the problem of big data Google come up with GFS (Google file system) [2]. It is an easily scalable distributed file system to handle large amount of data [2]. GFS is designed with most of the common goal of distributed file systems with some assumptions, such as use of low cost commodity hardware, frequent failure of hardware etc [2].

A well known open source Apache Hadoop project[3] also include similar kind of module named Hadoop distributed File System(HDFS) to store large amount of data in a scalable commodity hardware. The Apache Hadoop's key components- MapReduce[8] and HDFS[4] are originally derived from Google's GFS[2].

Similarly Gluster File System is an open source network attached file system with the easy provision of scale up the storage as well as the processing power [7][10]. Gluster file system aggregate the storage servers connecting through ethernet or infiniband RDMA in a one larger parallel network storage file system [7].

The rest of the paper is organized as following: section II discusses about the design, background details of GFS, HDFS and GlusterFS distributed file system. Section III gives a brief analysis of the DFSs and presents a comparison among the DFSs on some fundamental issues of DFS. In section IV we draw a conclusion.

II. BACKGROUND WORK

It is very difficult to make a thorough study given the number of DFSs available. Here we do the choice of DFSs according to their popularity and used in production purposes: GFS, HDFS and GlusterFS are some of them. The basic details about GFS, HDFS and GlusterFS are as shown below-

1. GFS

The Google File System is a scalable and highly fault tolerant distributed file system for large distributed data-intensive applications. It was designed based on the need from application workloads and technological environment at Google [2].

1.1. Architecture

GFS is a centralized distributed file system. It is composed of a master node and a number of chunk nodes that are run on commodity hardware. The data are split into a numbers of fixed size chunks and are stored across the chunk servers [2]. The master node maintains a Meta data that include file system namespace and also the mapping of chunk servers and their states. The master node is the central part of the file system. Any failure in the master node makes the whole cluster unavailable. To avoid this single point of failure a shadow master node is maintain and has the same data that of the master node.

1.2. Naming

In GFS the namespace is handled by the master node. Here the files are divided into number of chunks and are stored in the chunk servers. Thus the file to chunk mapping is needed and are stored in the namespace [2]. The namespace is consisting of Meta data and hierarchy of files and directories.

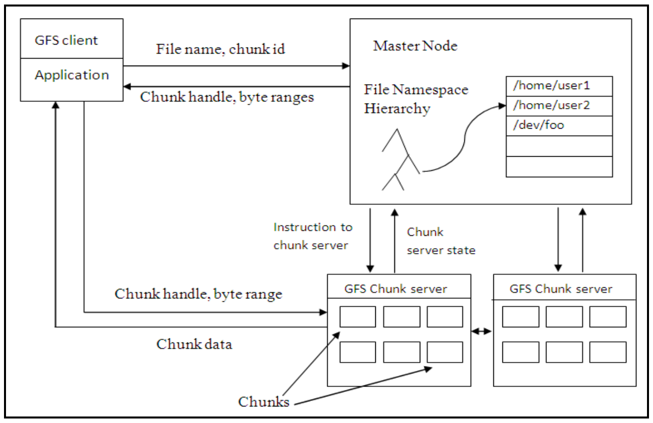

Figure *1. GFS Architecture*

1.3. Replication and Synchronization

In GFS, files are divided into chunks and are stored in the chunk servers. To increase the reliability, the chunks are replicated across the whole system according to replication policy. The default replication number of a chunk is 3. While doing the replication the load balancing of the whole rack [2] is taken into consideration. A rack is consisting of a number of chunk nodes. Normally, a chunk node can hold only one copy of a chunk, and a rack can hold two copy of the chunk. Since the chance of node failure is higher than that of the complete rack.

1.4. Fault detection

All servers in GFS are fully connected and are communicate with each other to detect any problems such as network or any server failure and to keep the system secure and available. At startup each chunk node communicates with the master node to check whether it belongs to the chunk node list. If they do not then the chunk nodes are shutdown and thus maintain the integrity of the system. Each chunk nodes sends heart beat [2] to the master node to conform its availability. The master node considers a chunk node as unavailable if it does not receive any heart beat for a definite time period. Heartbeats also provide statistical information (such as storage capacity, number of data transfers in progress etc.) to the master node so that it can make decisions for load balancing.

2. HDFS

HDFS is the Hadoop distributed File System under Apache license 2.0 developed by the Apache Software foundation [3]. HDFS is designed to be highly fault tolerant since it is implemented in commodity hardware.

2.1. Architecture

HDFS has a number of similarities with GFS when we take architectural part into consideration. It is a centralized distributed file system where the namespace is managed by the Name node. The files are divided into fixed size blocks and are distributed and replicated across the data nodes. A secondary name node is provided and is a persistent copy of the name node. In case of name node failure, the namespace can be retrieve successfully from the secondary name node.

2.2. Naming

HDFS supports traditional hierarchy of file organization namespace. HDFS handles its namespace using inode concept which contains metadata such as permissions, space disk quota, access time etc. The namespace and the metadata are maintained by the name node. Any changes in the file system or its properties are recorded by the name node.

2.3. Replication and Synchronization

HDFS is designed to be reliable to store large set of data across a cluster of data nodes. To avoid loss of data because of data node failure, blocks are replicated across the clusters. The number of replication and size of blocks are configurable. HDFS uses rack aware replica placement policy to improve the reliability, availability and network bandwidth utilization. A rack is a cluster of data nodes. Normally a HDFS cluster is configured with replication number 3, and in a rack no data node can have more than one copy of a block and a rack as a whole can hold only two copies of the block. Since the chances of whole rack failure is far less than that of a data node failure. The whole replication is maintained and monitored by the name node.

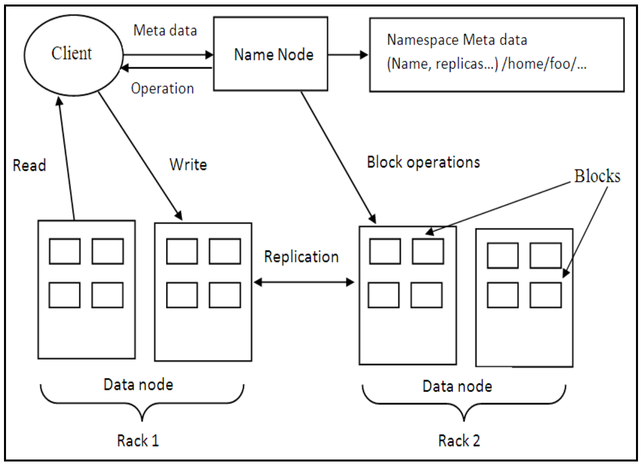

Figure *2. HDFS architecture*

A heartbeat is a signal send to the name node by the data nodes after a fixed interval of time to inform about their availability. A block report contains the list of all blocks in a data node. If name node detects any block is under or over replicated, then it instruct the data node to take the appropriate operation.

In HDFS data are replicated asynchronously. We can access the data while replication is underway. This improves the accessibility of data, but inconsistency occurs if modification is done before the synchronization is completed.

2.4. Fault detection

At startup each data node compares its registered namespace id with that of the name node. If no match found then the data node is shutdown, thus preserve the integrity of the system. Data nodes send heartbeats to the name node periodically to inform their availability. A network partition can cause the data node loss connectivity with the name node. Name node detects this by the absence of heart beat and thus does not send any I/O operations to the data nodes.

3. GlusterFS

GlusterFS[10] is an open source distributed file system developed by the Gluster core team. It cluster together the storage building block over Infiniband RDMA or TCP/IP interconnect [7], aggregate the disk storage and memory resources and manage the data in a single global namespace.

3.1. Architecture

GlusterFS is a network attached file system with the easy provision to scale up the system. It has a client server design with no central Meta data server. It stores data and Meta data on different nodes connected across the servers. Each node exports a directory which is called a brick. A set of bricks from different connected nodes creates a logical volume. The system can be configuring to stripe the data into blocks and then store or replicate the blocks across the different volumes. The Meta data for the files are stored across the system. Thus the system does not have any central part.

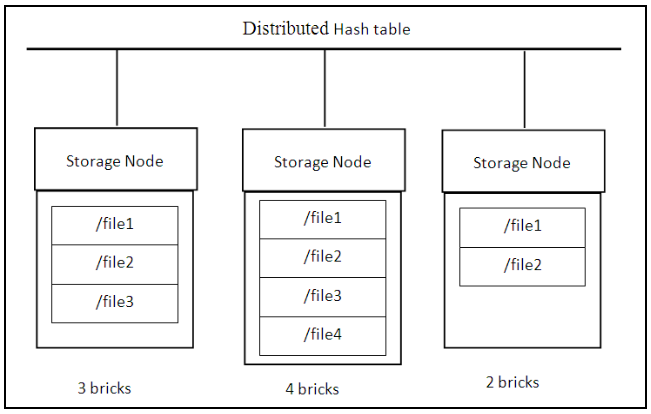

Figure *3. Architecture of GlusterFS*

3.2. Naming

GlusterFS does not maintain Meta data in a dedicated centralized server, thus does not have any single point of failure. Instead GlusterFS locate any file using Elastic hashing algorithm (EHS) [7]. It systematically locates a file solely based on their name using EHA. EHA uses a hash function that converts the file path name to a fixed sized, uniform and unique value. Thus we can access a file using that unique value only.

3.3 Replication and Synchronization

GlusterFS does not replicated data one by one as compare to other file systems. It relies upon RAID. It makes several copies of the whole storage and then replicated them other storages inside a same volume using synchronous writes.

Since replication in GlusterFS is done synchronously, files cannot be access until all the files are synchronized successfully.

3.4. Fault detection

In GlusterFS when a server is not available, it is removed from the system and no I/O operation is done to that server. The Meta data in GlusterFS are replicated across the whole servers. Thus the system is free from single point of failure.

III. ANALYSIS AND COMPARISON

The aim of analyzing GFS, HDFS and GlusterFS is to know how they address the scalability, transparency, fault tolerant issues of a distributed file system.

All distributed file system must face with the increasing number of client request, I/O operations and storage of different sized files. So DFS must be design with the easy provision of scale up the system in term of processing power, storage capacity. Scalability is the ability of any DFS to efficiently and dynamically increase the performance by addition of servers to the system.

In distributed file system, users do not need to know about the underlying complexity of the system (such as system design, data locations, fault detection etc.). The user should able to access the system regardless of their location and to do the same set of operations on the DFS as they do on any local file system.

Failures are consider as a norm rather than exception in case of distributed file system. So a DFS should not stop functioning in case of any transient or partial failure. The failure can be network or any server. There should always be a provision to maintain data integrity, consistency of the system in case of any failure.

In the next section, we discuss about GFS, HDFS and GlusterFS on the basis of scalability, transparency and fault tolerance and give a brief comparison among the DFSs.

1. Scalability

In GFS and HDFS we can scale up the storage capacity by adding new servers to the system as many as we want. The metadata (namespace, mapping from files to blocks/chunks) of both the DFS is handled by a single dedicated server. So there is a natural limitation on amount of client request the system can process at a time. HDFS loads the metadata on the memory of the namenode to increase the performance of the system. But this limits the number of files can be store in distributed file system. This is the reason why GFS and HDFS are suitable for storing small number of large sized files.

On the other hand GlusterFS is distributed the metadata on each and every machines of the system. Thus GlusterFS can implement distributed request management [7]. The storage and the client request handling capacity can be scale up by simply adding new servers to the system. Unlike GFS and HDFS, GlusterFS can handle both small and large sized files smoothly.

2. Transparency

All the DFS provide various mode such as web based, command line based interface to access the system. The users do not have to know about the underlying complexity of the system to use it. Again DFS should detect any failure in the system before that affect on the users performance. For the detection of failure different type of methods such as periodical message sharing [3] within the nodes are used.
In GFS and HDFS, the files are divided into blocks and are stored in the data/chunk nodes. The mapping of a file to their blocks, called indexing, is then maintained in the master node/namenode [2][5]. User can access the file with the correct index values. HDFS uses heart-beat signals to known about the availability of data nodes. All data nodes periodically send heart beat signals to namenode to inform about their availability. If the namenode does not receive any heart beat from a particular data node for a definite duration of time then the namenode consider the data node as unavailable and thus removes the node from the list.

In GlusterFS there is no definite index is maintained for any file. A user needs to calculate the file location using Elastic hashing algorithm (EHA). The metadata servers only provide the require information for the algorithm to calculate the location of a file.

4. Fault Tolerance

Due to the distributed nature failures in distributed file system are treated as usual rather than as exception. The failure may be network or any server, but a DFS should be capable of handling the failure and should ensure the availability, integrity and consistency of data.

GFS and HDFS are two examples of centralized file system. Here the master node/namenode is responsible for maintaining metdata of the file system. Thus failure of metadata server makes the whole cluster unavailable. Hence for GFS and HDFS, the metadata server is the single point of failure. To overcome this problem, GFS and HDFS maintain another metdadata server called secondary master node/namenode [2][5], which is the replica of master node/namenode. Again to avoid any accidental loss of data due to data/chunk node failure, data blocks/ chunks are replicated across the data nodes and racks. The default replication factor for GFS is 3 which can be change and for HDFS we can configure it according to our need. HDFS uses asynchronous replication due to which inconsistency arises in data. To solve this problem HDFS uses WORM mechanism. That is write once and read many times. GlusterFS is a highly available distributed file system. Since the metadata and data are distributed across the nodes, there is no single point of failure. And also if any node becomes unavailable, it does not affect the availability of data. GlusterFS replicated data synchronously.

Table 1. Comparison of DFSs

Sl no		GFS	HDFS	GlusterFS
1	Architecture	Centralized	Centralized	Decentralized
2	Naming	Handled by the master node. Data are divided into blocks and their indexing is stored in master node	Handled by the namenode. Data are divided into blocks and their indexing is stored in namenode	Meta data are distributed across the clusters. Files are located using EHA algorithm
3	Replication	Asynchronized and automatic replication	Asynchronized and automatic replication	Synchronized replication
4	Failure	Single point of failure	Single point of failure	Highly available due to distributed nature

IV. Conclusion

Distributed file system is used as a solution to store and process large set of data. All the DFSs are designed to adopt the increasing client request, storage capacity and also maintain the system transparency, fault tolerance. All the DFSs handle the transparency and fault tolerant issue almost the similar ways. The main difference lies in their design. In GFS and HDFS, we can easily increase the storage capacity of the system by adding new node to the cluster. But the whole metadata (namespace of the file system) is handled by a single namenode/master node. This results in performance limitation and single point of failure. Again in GlusterFS, due to decentralized architecture the metadata is distributed across the system. So, there is no single point of failure. In GFS and HDFS the files are divided into blocks and the indexing (mapping of blocks to file) are maintained by the namenode/master node. In GlusterFS there is no definite indexing is maintained. The locations of the files are calculated by an algorithm. In GFS and HDFS the replication of data are done asynchronously. This enhances the overall performance of cluster. Again in GlusterFS replication is done synchronously. Files cannot be accessed until all the files synchronized successfully. From all these information we can conclude that distributed file system should be chosen according to the user’s requirement.

References

[1] Snijders, C., Matzat, U., & Reips, U. D. (2012). " Big Data": big gaps of knowledge in the field of internet science. *International Journal of Internet Science*, *7*(1),1-5

[2] Ghemawat, S., Gobioff, H., & Leung, S. T. (2003). *The Google file system* (Vol. 37, No. 5, pp. 29-43). ACM

[3] Shvachko, K.; Hairong Kuang ; Radia, S. ; Chansler, R. " The Hadoop Distributed File System". Published in: IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, May 2010. Page(s): 1 - 10. E-ISBN: 978-1-4244-7153-9. Print ISBN: 978-1-4244-7152-2. INSPEC Accession Number: 11536653. D.O.I: 10.1109/MSST.2010.5496972. Link: ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6308241

[4] Borthakur, D. (2008). HDFS architecture guide. *Hadoop Apache Project*, *53*

[5] K. Shvachko, H. Kuang, S. Radia and R.Chansler, “The Hadoop Distributed File System”, IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), (2010) May 3-7: Incline Village, USA.

[6] Claudia Hauff, “Big Data Processing”, 2013/14 Lecture 5 (Web Information Systems).

[7] An introduction to Red Hat Gluster Storage architecture. (2015). Retrieved April 20, 2018, from www.redhat.com: https://www.redhat.com/en/files/resources/en-rhst-gluster-storage-tech-detail-11395347.pdf

[8] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

[9] Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010, May). The hadoop distributed file system. In Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on (pp. 1-10).

[10] E. B. Boyer, M. C. Broomfield, and T. A. Perrotti. GlusterFS One Storage Server to Rule Them All. echnical report, Los Alamos National Laboratory (LANL), 2012.

Authors Profile

Anupom Chakrabarty received his B.E. degree from Gauhati University and currently pursuing M.Tech degree at Department of Information Technology, Gauhati University, Guwahati.

Chandan Kalita currently works at the Department of Information Technology, Gauhati University. His research work is in File System/ Non Volatile Memory. His recent research interests are Distributed File System, NOSQL, Big data.