FUDMA Journal of Sciences (FJS)
ISSN: 2616-1370
Vol. 2 No. 2, June, 2018, pp 223 – 233

A COMPARATIVE STUDY OF THE ARCHITECTURES AND APPLICATIONS OF SCALABLE HIGH-PERFORMANCE DISTRIBUTED FILE SYSTEMS

Dada, E. G. and Joseph, S. B.

Department of Computer Engineering, University of Maiduguri
Maiduguri - Borno State, Nigeria
Correspondence: gbengadada@unimaid.edu.ng

ABSTRACT

Distributed File Systems have enabled the efficient and scalable sharing of data across networks. These systems were designed to handle some technical problems associated with network data. For instance reliability and availability of data, scalability of infrastructure supporting storage, high cost of gaining access to data, maintenance cost and expansion. In this paper, we attempt to make a comparison of the key technical blocks that are referred to as the mainstay of Distributed File Systems such as Hadoop FS, Google FS, Luster FS, Ceph FS, Gluster FS, Oracle Cluster FS and TidyFS. This paper aims at elucidating the basic concepts and techniques employed in the above mentioned File Systems. We explained the different architectures and applications of these Distributed File Systems.

Keywords: *Distributed File Systems, Google File System, Hadoop File System, Oracle Cluster File System, Ceph File System.*

INTRODUCTION

Distributed File System (DFS) is an addition to the concept of file system which performs the function of managing files and data that are stored on various devices on the computer system. They are characterised with high performance, scalability and dependability through the use various state-of-the-art techniques. The people outside sees the DFS as one undivided storage medium (Vaidya and Deshpande, 2016). File systems are concept that allows users to read, maneuver and arrange data (Shyam and Sudarshan, 2015). Usually, the data is kept in storage locations as files in a hierarchical tree in which the nodes are referred to as directories or folders. The file system allows a similar view, unconnected to the primary medium of storage that can be floppy disk, hard disk, flash disk and CDs (Suralkar *et al.*, 2013).

A distributed file systems (DFS) is a system that permits many users to retrieve, via the network, a file structure stored on one or other machines in isolated or distant locations (File Servers) by means related structures to the one employed to retrieve the file stored on the local machine. It uses a client/server model in which data is distributed among many storage locations, usually known as nodes (Elomari *et al.*, 2017). Distributed file systems are occasionally perceived to be a single storage device but in actual fact they are interface to a larger extent creating the platform for the storage of data on several machines. However, the DFS offers location transparency and duplication to enhance data accessibility in situation where there is failure or heavy load. One of the drawbacks of DFS is bottlenecks which can lead to traffic jam and restricted access in some workstations. Such situation can be expanded to cover a sizable quantity of storage locations and offering moderate performance degradation in their operations while there are possibilities of hardware failure.

HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

The Hadoop is a distributed parallel, fault-tolerant distributed file system that drew its creative ideas from the Google file system (Suralkar *et al.*, 2013). The architecture was meant to dependably store files that are very huge in size around machines having large allocation unit and deployed on low-cost hardware (Elomari *et al.*, 2017). The Hadoop DFS is advantageous because it allows large volume of data to be processed within a short time and is appropriate for use in storing applications with huge data sets. Hadoop Distributed File System (HDFS) divides large data files into different clusters and each segment is managed by different machines in the group (Shvachko, *et al.*, 2010). Each segment is duplicated across many machines in a cluster, so that if any of the machines malfunctions it does not makes the data to be inaccessible (Pooja, 2014). In spite of the several benefits of HDFS, there is still a foremost

challenge which is data security. Whenever a virus, malware or worm infect a data, it rapidly affects all the data. Another challenge with HDFS is that it is susceptible in nature and unsuitable for storing small data. Each file is stored as a sequence of blocks; all blocks in a file excluding the last block are of the same size. Blocks belonging to a file are reproduced for fault tolerance. The block size and replication factor are can be configured for each file. The files are "write once" and only one writer can write on it at any time (Hurwitz *et al.*, 2013). Other drawbacks of Hadoop DFS include centralisation. The Hadoop system uses a centralized master server. Thus, the Hadoop cluster is unreachable whenever its NameNode is inoperative (Gemayel, 2016). A suitable recovery approach to solve this problem so far is to restart the NameNode, and adequate steps are being taken regarding the ability of the system automatically recover. Moreover, HDFS have scalability issues. Since the NameNode stores the entire namespace and block locations in memory, the size of the NameNode heap reduces the number of files and blocks that can be addressed. A possible solution to this problem is to permit the sharing of physical storage by several namespaces and NameNodes within a cluster (Suralkar *et al.*, 2013).

Architecture of HDFS

HDFS cluster contains one namenode, a master server and several datanodes known as slaves in the architecture. The HDFS stores filesystem metadata and application data independently. HDFS stores metadata on an autonomous dedicated server named Namenode and Application data are stored on separate servers termed Datanodes. Every servers are fully connected and communicated with the TCP based protocols. Figure 1 shows the complete architecture of the HDFS.

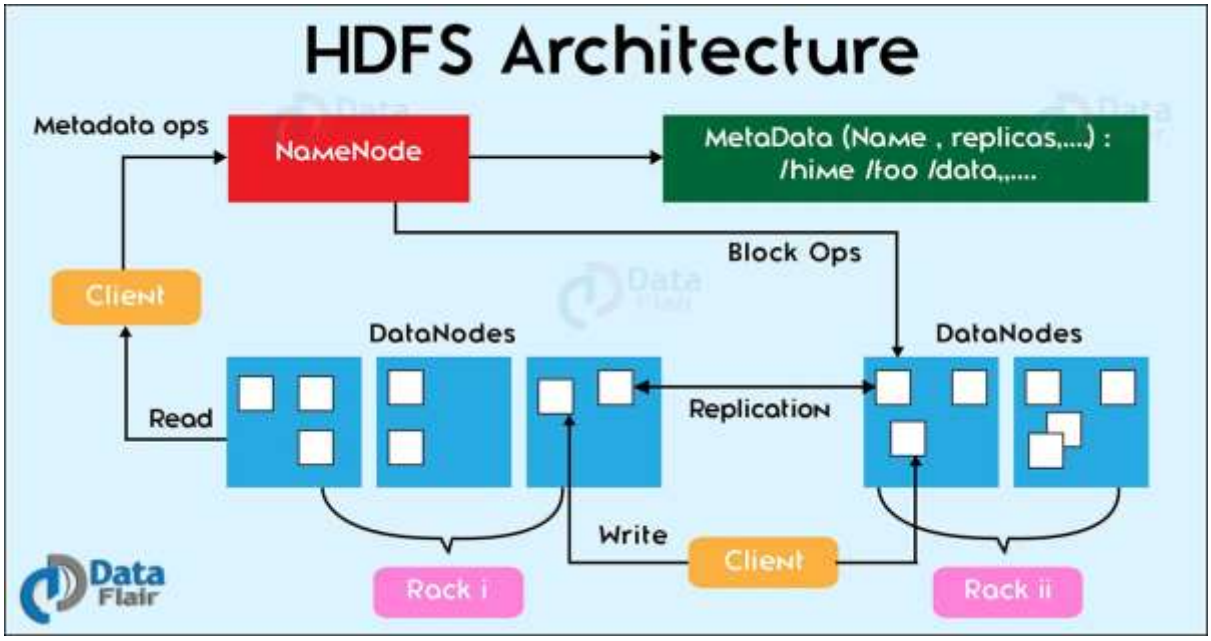

Fig.1: HDFS Architecture (Source: DataFlair Team, 2017).

Areas of Application of HDFS:

According to Sravanthi and Reddy (2015), the areas of application of HDFS among many others include the following;

1. **Agriculture:** HDFS have found application in the agricultural sector. For instance in genetic engineering, where sensors are used to monitor plant reactions to various changes in the environment, large amount of data is collected and the simulation will allow for the discovery of the most appropriate environmental conditions for various plants.
2. **Stock Exchange:** HDFS is broadly used through an analytical database to identify illegal trading patterns and also to uncover fraudulent activities.
3. **Big Data Applications (BDA):** HDSF can be used to effectively store big data. Big data application is a software application which analyses big data by processing them in a very large parallel framework. For example data from traffic routine, stock market updates, tweet massages and others.
4. **Clustering:** Data store in HDFS can be divided clusters. This is used mainly to identify and address group of data through a single click using a k-means algorithm.

5. **Enterprise:** HDFS is used for in-data analytics help business to make decisions faster than another traditional analytic tool.
6. **Credit cards:** HDFS can assist an organisation to detect possible fraudulent acts in credit cards transactions. Companies depend on the in-database analytics because of its speed and accuracy. It does a kind of verification before authorization where there is any suspicious activity.
7. **Consumer goods:** HDFS is used for storing collection of consumers' details and activities. For example type of goods or product, place of purchase, quantity purchased, online transactions etc. When analyzed, it can help a company to understand some information such as the reason why customers purchase some goods more than others and the areas where they have to intensify their efforts such as marketing the product.
8. **Banking:** Banks make use of HDFS for storing customer data and also assists in detecting any questionable customer activity. It provides easy access to customer's information wherever the customer requests for it. It helps the banks to also track their progress and enhance the efficiency of their service.
9. Hadoop is also being used by the giant ISP (Yahoo) and popular social media such as Facebook.

Google File System

Google File System (GFS) is a distributed file system introduced by Google to manage huge amount of data which is spread across different databases (Ghemawat *et al.,* 2003). GFS is mainly intended to provide efficient, dependable and fault tolerant access to data by employing huge clusters of commodity servers. Google system (GFS cluster) is made up of one master and many Chunk servers (nodes) and can be retrieved by several clients (Vijayakumari *et al.*, 2014). GFS files are partitioned into chunks of 64 megabytes, and are normally appended to or read. It is overwritten or compressed only in exceptionally seldom cases. Compared with other conventional file systems, GFS is planned and enhanced to operate in data centers to offer exceptionally great data processing capacity, minimal delay to rapidly changing data and continue in its operations despite individual server failures (Gemayel, 2016). The features that make GFS highly appealing include: continuous operation in the event of system breakdown, important data duplication, automatic and effective, salvaging corrupted or damaged data, superior sum of the data rates that are delivered to all terminals in a network, decreased client and master communication due of huge chunk server size, namespace organisation and securing, and better accessibility.

Some of the advantages of GFS include its very high availability and high faults tolerance through data duplication. Its single master design makes it very efficient and simple. GFS ensures data integrity by It is easy to run both a chunkserver and a client on the same machine, as long as verifying of each copy by chunk server using check sum. It has a reduced check sum cost. It has an increased bandwidth because of its batch operations such as cabbage collection and writing to operation log. At client end, no synchronization is needed because of the append operations. It takes care of caching issues. It automatically collects garbage chunks using garbage collection. Continuous monitoring of chunk server by GFS through continuous messaging (Ghemawat *et al.,* , 2003).

The GFS is limited to special purpose design and thus cannot accommodate a general-purpose design. It is very inefficient in the case of small files (Gemayel, 2016). Slow garbage collection is a setback i.e. when the files are not static. Consistency checks are done by the clients themselves. When the number of writers increase there can be degradation in the performance of the system. The master memory also poses a limitation (kaushiki, 2012).

Architecture of GFS

A GFS cluster is made up of a one *master* and many *chunkservers.* It can be retrieved by several *clients*, as shown in Figure 2. Each of the components is normally a service Linux machine running a user-level server process. The chunkserver and a client can be run on similar computer without difficulty provided that the resources on the computer system allows it and the inferior inconsistency brought about by running perhaps application code that is likely to act in an unusual manner is suitable (Ghemawat *et al.,* , 2003).

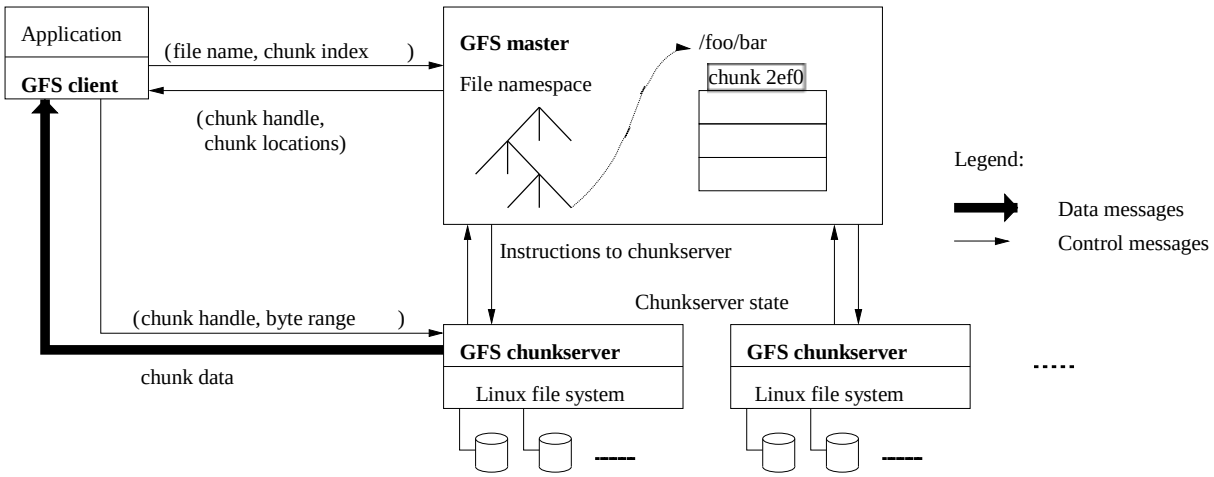

Fig. 2: Architecture of GFS (Source: Ray, 2015)

Areas of Application of GFSMapReduce:

This is the major application area of GFS. MapReduce is a programming model proposed by Google and used by both GFS and HDFS. The principal responsibility of MapReduce is to serve as a platform for the development and execution of large-scale data processing tasks. Hence, MapReduce takes advantage of the high processing power made available by computing clusters while at the same time presenting a programming model that makes easier the development of such distributed applications. MapReduce make the breakdown of jobs and fusion of results stress-free. It also offers the opportunity to easily track jobs and task.

LUSTER FILE SYSTEM

Luster is a file system designed for efficient storage (Paul *et al.,* 2013). Luster file systems have the capacity to change in size and can be integrated into several computer clusters that have thousands of client nodes, large storage capacity on several servers (Sage *et al.,* 2004). This makes Luster file systems a widely accepted file system for use in businesses where massive data centers are required. Luster file systems have high performance abilities and open licensing, it is commonly used in supercomputers (Paul *et al.,* 2013). Today, Luster is completely based on Linux and usually use kernel-based server modules to produce the required performance, but it can be re-exported with NFS or CIFS to allow use by Windows and OS X clients. Another notable features of Luster are incorporated network diagnosis, and performance tracking and fine-tuning mechanisms. Luster can support several kinds of clients and runs on almost any modern hardware. Scalability is one of the most critical characteristics of Luster. It can also be used to produce a single namespace of what seems to be practically immeasurable capacity (Paul *et al.,* 2013).

Architecture of Luster File System

The Luster architecture aims to connect a large number of clients with the data servers in an efficient and foolproof manner. According to (Knowledge Base, 2018), Luster file system is made up the following components:

- **Luster clients:** This DFS client software runs on machines like desktop nodes that interacts with file system's servers through the Luster Network (LNET) layer. In an establishment, Luster offers clients a combined all inclusive namespace for every files and data in the file system. When Luster is attached on a client, its users can manage file system data in a way that makes it look as if the data is stored locally. Nevertheless, the clients will under no circumstances be able retrieve data instantly from any part of a computer file, without having to read the file from the beginning in the main file storage.
- **Management Target (MGT):** The MGT keeps file system information settings for use by the clients and other Luster parts. Even though MGT storage prerequisites are reasonably low even when the system that controls how data is stored and retrieved file is very huge, the information kept in it is highly essential to log on to the system.
- **Management Server (MGS):** It manages the organization data stored on the MGT. Luster clients communicate with the MGS to access information from it.
- **Metadata Target (MDT):** It keeps filenames, directories, authorizations, and other data that gives information about the namespace.
- **Metadata Server (MDS):** The MDS is in charge of data about namespace kept on the MDT. Luster clients get in touch with the MDS to access information from the MDT.

The MDS does not participate in file read/write actions.

- **Object Storage Targets (OSTs):** The OSTs keeps user file data in one or many logical objects which have the ability to be streaked hrough various OSTs.
- **Object Storage Server (OSS):** The OSS supervises read/write actions for (normally) several OSTs.

Depicted in figure 3 below is the architecture of the Luster distributed file system.

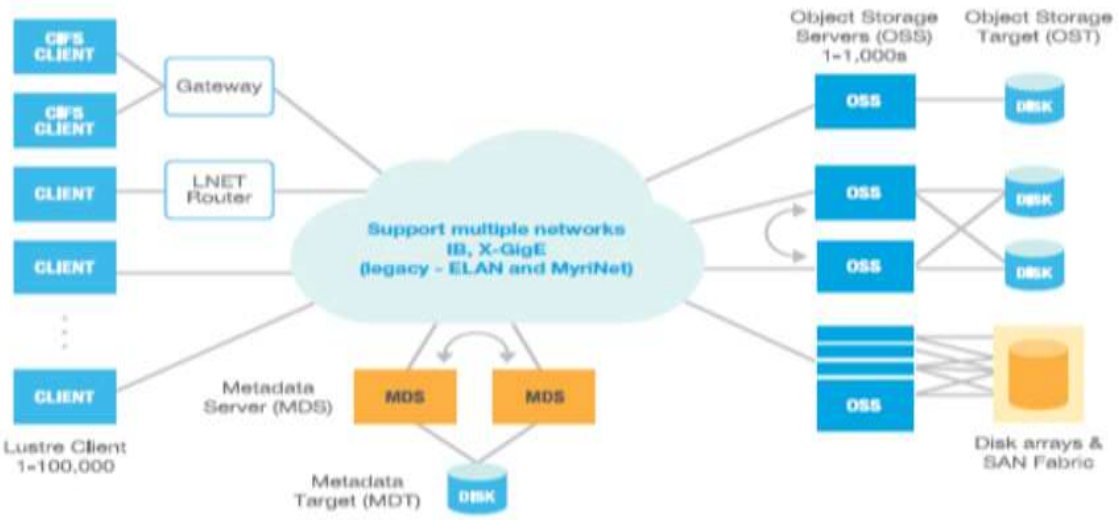

Fig. 3: Luster Architecture (Source: Torben, 2018)

Area of Application

This Luster file system are used in various areas of applications such as Aeronautic Engineering, Banking, Topography, Science of Weather and Climate. In in industries such as meteorology, simulation, oil and gas, life science, and finance. It is remarkable to know that a Luster file system is employed for a wide range of purpose, it is used for program interfaces and services relative to the initial user of these interfaces and service in many sites, from Internet service providers (ISPs) to large businesses that deals with financial and monetary transactions.

Ceph Distributed File System

Ceph a distributed file system that offers users outstanding performance, dependability, and scalability. Ceph exploits the split between data and metadata organization by substituting table that an operating system maintains on a hard disk that provides a map of the clusters with a pseudo-random data distribution function (CRUSH) designed for diverse and dynamic clusters of undependable object storage devices (OSDs). Ceph was designed to be a reliable, scalable fault-tolerant parallel file system. Incorporated into Ceph is an smart and robust data placement scheme, named CRUSH. The CRUSH algorithm allows a client to pre-calculate object placement and layout while taking into consideration of failure domains and hierarchical storage tiers. Ceph is built on top of a unified object management layer, RADOS. Both metadata and the file data can take advantage of this uniformity. Most of the Ceph processes reside in user-space. Generally speaking, this makes the system easier to debug and maintain. The client-side support has been integrated into Linux mainline kernel, which eases the deployment and out-of-box experience (Feiyi *et al.*, 2013).

The CephFS is very attractive because it provides a very robust data safety for mission critical applications. It makes practically boundless storage file systems possible. Applications that use file systems can use CephFS with POSIX semantics. Moreover, it does not need any integration or customization. CephFS automatically stabilise the file system to provide best performance. Also, CephFS enables enhanced scalability of the system in which clients execute huge read/write operations that linearly scale with the number of objects storage devices in the RADOS cluster. However, each Client is constrained by the bandwidth of its network link. Some of the weaknesses of CephFS is that it occasionally endangers file system by making unauthorised users to be aware of the existence such thereby compromising data privacy. Features like Snapshots are not offered by CephFS. Also, testing gets inadequate coverage in test suite.

Architecture of CephFS

CephFS is a file storage solution part of Ceph. It works as a high-level component within the system that provides files storage on top of RADOS, the

object is a store upon which all Ceph storage solution are built (Sage *et al.,* 2006). Two types of entities cooperate to provide a file system interface: Clients and metadata server (MDS). What is required of a client is just to open a file (i.e. you inform the MDS that you wants to use the file) and afterward the read/write is changed instantaneously, with only intermittent updates to the MDS (Giacinto *et al.,* 2014). The CephFS is depicted in figure 4 below.

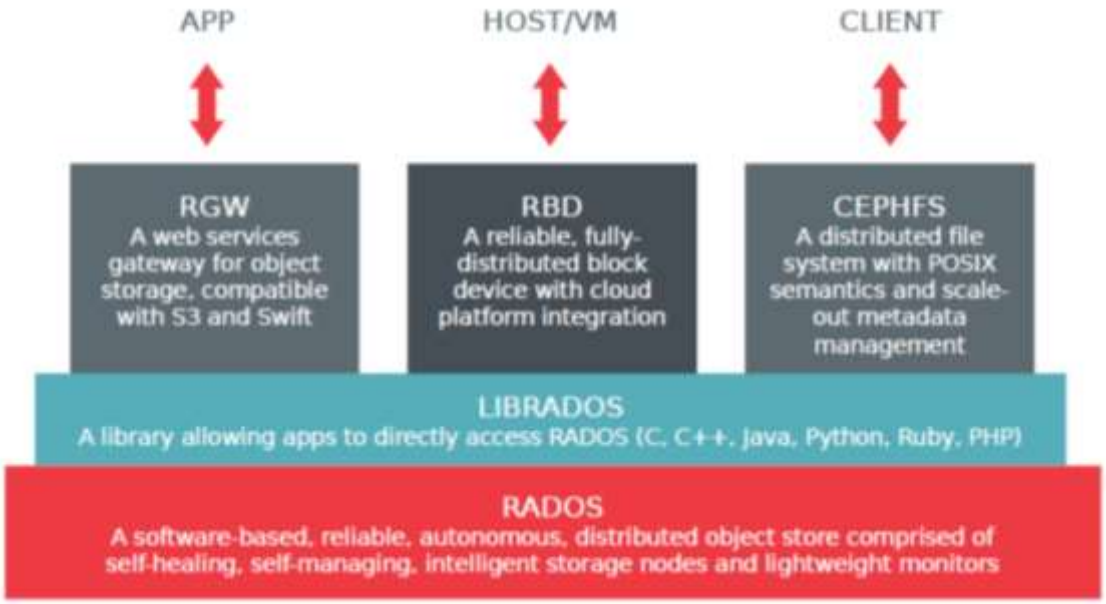

Fig 4: CephFS Architecture (Source: John, 2015)

Areas of Application of CephFS

1. CephFS is used in storing large files efficiently.
2. They are used in the Field of Telecommunication for storage and distribution.

GlusterFS File System

GlusterFS is a distributed network file system that has the capability to handle growing amount of work. It is developed using C programming language (Shyam and Sudarshan, 2015). By means of conventional standard hardware, GFS can produce massive, storage solutions that are geographically dispersed. It can be used for media streaming, data transformation and modeling, and other data and jobs that affects the maximum data transfer rate of a network or Internet connection. GlusterFS is an unrestricted and source code that anyone can inspect, change, and distribute. It is easy to understand its architecture. Users put it on their computer for personal use. The idea of information about data for storing files on server does not apply in GlusterFS. It have the ability to compute files' address and retrieve files without difficulty. GFS utilizes the idea of bricks and volumes to separate various user's data on small allocation units for a file within a file system (Shyam and Sudarshan, 2015). GlusterFS have four major notions:

Bricks - the part of a computer in which information is stored. It is made up of a server and route that points to a file system location by following the directory tree hierarchy (i.e., server:/export)

Translators - components that are joined to transport data from point A to point B

Trusted Storage Pool – a reliable interconnection of servers that will as the central repository of data and programs that are shared by users in a network

Volumes - group of bricks having the same condition for redundancy.

Architecture of GlusterFS file System

GlusterFS system architecture is easy to comprehend and it is equally a robust file system written in user space that employs FUSE to attach itself to programs that forms an interface between an operating system's kernel and a more concrete file system. GlusterFS file system architecture is in different levels. This makes it possible to add or delete features. GlusterFS file system works on ext3, ext4, xfs.etc.to store data. It allows horizontal growth or the addition of new resources in the network. The conceptual model that defines the structure of GlusterFS is depicted in figure 5 below.

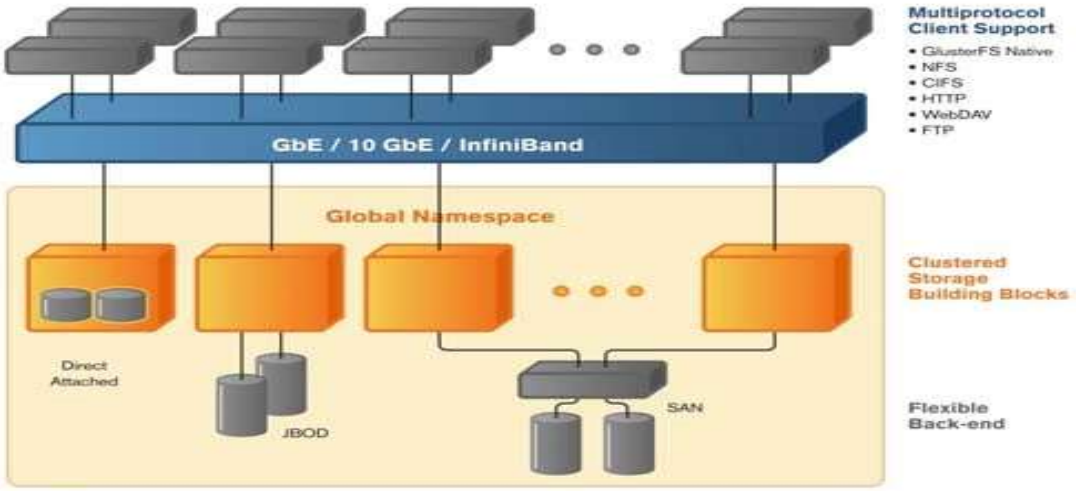

Fig 5: GlusterFS System Architecture (Source: Shyam and Sudarshan, 2015)

Areas of Application of GlusterFS

GlusterFS cluster have found relevance in computer storage of large and voluminous data, electronic device that can be used to store data (e.g. hard drives, CDs, DVDs, Floppy Disks, USB drives, ZIP disks, magnetic tapes and SD cards).

The Oracle Cluster File System (OCFS)

The OCFS is a file system which is shared by being installed on many servers concurrently. Oracle makes balancing and automatically and seamlessly switch to a highly reliable backup solutions such as OCFS2 and ACFS possible by allowing shared disk clustered file system architecture (Burleson, 2017). A cluster comprises of two or more autonomous, but interconnected, servers. Many hardware merchants have made available cluster capacity for some years to satisfy many needs. Some clusters were meant only to offer the users superior availability by permitting the relocation of work to an auxiliary node if the operational node crashes. The reason for designing others was to provide scalability by making it possible for user connections or work to be distributed across the nodes. Additional popular attribute of a cluster is that it should seem to an application that it is one server. Likewise, many servers should be managed as much as possible in the same manner as a single server is being managed (Burleson, 2017). The software that balances workload to reduce bottlenecks in the cluster makes this transparency possible. In other for the nodes to appear as one server, files need to be kept in a way that they can be located by the particular node that requests for them. We have existence today several softwares that define the type and state of each node in the cluster and the relation between them. This solves the data retrieval problem though it relies on the prim goal of the person that designed the cluster. The connection is a computer network topology employed as a way of transmitting information between each node of the cluster. A cluster is made up of many connected computers or servers that look as though they are one server to end users and applications. Oracle Real Application Clusters (Hupfeld, *et al.*, 2008). Some of the benefits of OCFS are Advanced Security (POSIX ACLs and SELinux), REFLINK Snapshots with Copy-On-Write, in-built Clusterstack with a Distributed Lock Manager, file Size Scalability up to 16 TB, and cluster Scalability up to 32 Nodes.

Architecture of Oracle Cluster File System

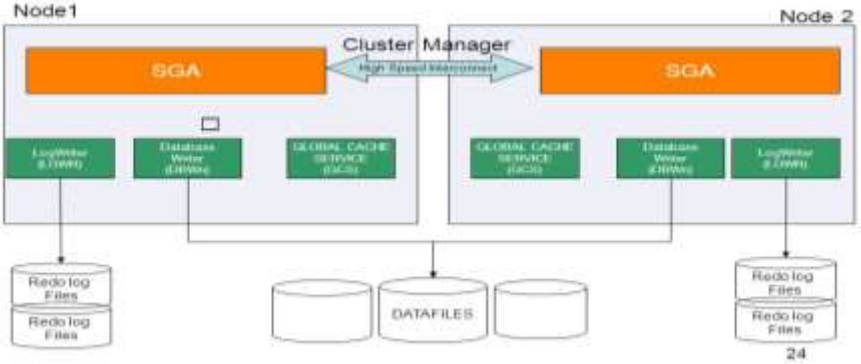

Fig. 6: Oracle Cluster File Architecture (Source: Carlos, 2008).

Area of Application of Oracle Cluster file system

According to Burleson (2017), Oracle Cluster File System is applicable in the areas explained below:

i. Metadata caching
ii. Metadata journaling
iii. Asynchronous and direct I/O support for database files for enhanced database workload, throughput, resources, optimization, and contention.
iv. Oracle Database has found application in storing files related to software programs designed to accumulate, manage and disseminate information efficiently like CAD, medical images, invoice images, documents, etc. The SQL usual data type, BLOB (and CLOB) is used by applications to store files in the database. Oracle Database offers superior security, accessibility, ability to operate without breakdown despite degradation in transactions, and capacity to be handle increase in file size compared to conventional file systems. Each time files are stored in the database, they are copied or achieved, harmonized to the failure recuperation site by means of Data Guard, and regained together with the relational data in the database (Burleson, 2017).

TidyFS

The Tiny File Systems is a small uncomplicated client/server-based application that permits clients to retrieve and process data stored on the server as if it were on their own. TidyFS offer the abstractions required for concurrent processing of data on clusters (Fetterly, *et al.*, 2010). There has been a sudden increase of research in computing using large numbers of already-available computing components for parallel computing, to get the best number of useful computation at reduced cost. Often, the high number of reads and writes, causing latency and bottlenecks for such clusters is generated by a software systems that runs on a cluster of networked computers and looks like one dependable machine that offers huge collective volume of computational and I/O performance. Examples include Map Reduce, Hadoop or Dryad. They have a high-throughput, they are sequential, and are read-mostly (Fetterly, *et al.*, 2010). Unlike the other DFS that we discussed in the previous sections, TidyFS is very simple. The system does not have any complicated duplication protocols and read/write code paths by taking advantage of workload properties which include the nonexistence of simultaneous writes to a file by many clients, and the presence of end-to-end fault tolerance in the execution engine (Fetterly, *et al.*, 2010).

Some of the advantages of TidyFS include: it permits applications to carryout I/O by means of whatsoever access patterns and techniques for reducing the number of bits required to represent data. This can save storage capacity, speed up file transfer, and decrease costs for storage hardware and network bandwidth. It makes migrating data and tools from obsolete technologies to modern ones easier. It removes the need for an additional layer of indirection by means of TidyFS interfaces, ensuring that clients can realize the highest obtainable I/O performance of the indigenous system (Sajjad and Harirbaf, 2013).

Architecture of TidyFS

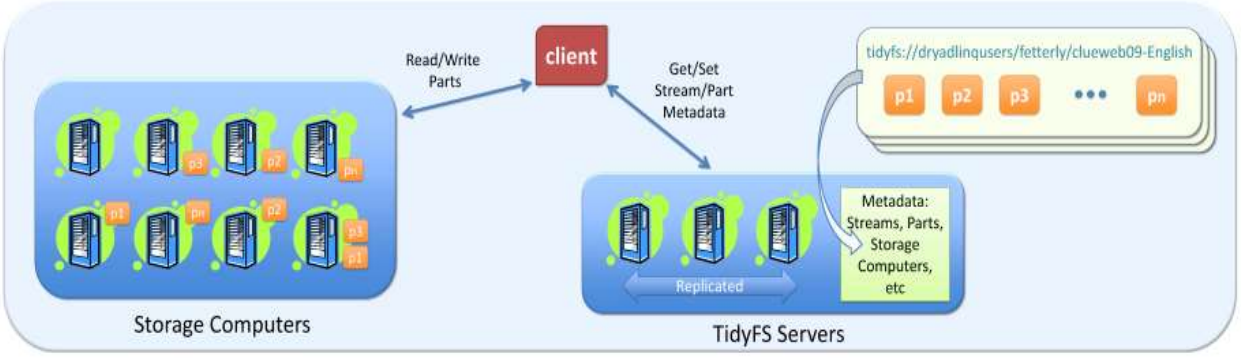

Fig. 7: TidyFS System Architecture (Source: Ghemawat et al., 2003)

The TidyFS storage system is made up of three sections: a metadata server; a node service that performs housekeeping tasks running on each cluster computer that stores data; and the TidyFS Explorer, a GUI which permits users to observe the status of the system

Area of Application

TidyFS is mostly used by TinyOS file system.

The table 1 below is a comparison of the different distributed file systems we studied in this paper.

Table 1: Comparative table of characteristics of distributed file systems under study

	Data Scalability	**Fault Tolerance**	**Data Access Concurrency**	**Data Striping**	**Supported OS**	**Reference**
HDFS	Yes	Block Replications. Secondary Namenode.	Files have strictly one write at any time	Block size of 128MB	Linux and Windows are the supported , but BSD, Mac OS/X, and Open Solaris are known to work	Suralkar *et al.*, 2013
Google FS	Yes	Chunk Replication. Meta data replication	Optimised for concurrent 'appends'	64MB Chunks	Linux	Gemayel, 2016
Luster FS	Yes	Meta-data replication by a single server. Data is stored on reliable nodes	Many seeks and read-and-write operations of small amounts of data	800 MB/sec of disk bandwidth	Linux and provides a POSIX-compliant UNIX file system interface.	Paul *et al.*, 2013
CephFS	Yes	Metadata is replicated across multiple MDS nodes.	Allow aggregate I/O performance to scale with the size of the OSD cluster.	14-node cluster of OSDs (around 58MB)	Linux VFS and page cache.	Sage *et al.*, 2006
GlusterFS	Yes	Replication means single file can be cloned and placed on multiple nodes.	Lowest level translator, stores and accesses data from local file system	Atleast 16KB file size	Linux but supports redhat. CentOs, Fedoro. Ubuntu flavors of Linux operating	Shyam and Sudarshan, 2015
Oracle Cluster FS	Yes	Oracle Net connect-time failover and connection load balancing.	It supports row level locking	Can handle 4.3 billion fully consistent reads and 1.2 fully transactio nal writes per minute	Windows, Red Hat Linux and United Linux	Burleson, 2017
Tidy FS	Yes	Automatic replication of read-only database parts	Uses native interfaces to read and write data	Not available	Windows	Fetterly, *et al.*, 2010

CONCLUSION

A comparative study of some important features of seven distributed file storage systems were considered in this paper. We first discussed the different types of distributed file systems, their architectures and areas of application. At the end, we drew a table of comparison (Table 1) whose each column's header is a vital attribute of a DFS system and each line's header parallels one of the seven DFS systems we studied. At the intersection of each row and column, we indicate whether the attribute is implemented by the system in addition to the distinctiveness of the implementation. It is obvious from our analysis that the foremost mutual interest of these systems is scalability. These systems are aimed at efficiently managing the large volume of data that kept on increasing on daily basis. Many drawbacks are associated with centralised storage systems. Their maintenance is complex and costly. Scalability in any DFS should with lowest cost and labour.

Moreover, availability of data and fault tolerance continue to be some of the main concerns of DFS. Several systems are apt to employ cheap hardware for storage. Such situation will the systems vulnerable to periodic failures. This challenge is corrected by means of replication, versioning, snapshots and others which have the goal of restoring the system state, in most cases spontaneously, once a fault or total loss occurs at any node. In addition to these mechanisms, data striping and lock mechanisms are included to control and enhance simultaneous access to the data. The development of concurrent access is very crucial for systems that manage huge files in substantial quantities. Locking a whole file to modify a portion of it can stop the access to this file for an undefined duration of time. Implementing solutions that will simply lock the byte range involved in the alteration is therefore principal. The ability of the DFS to work on multiple operating systems can be a great plus to its performance. Among the seven DFS we studied, the HDFS is the one offering the highest collection of operating systems that can support its implementation.

REFERENCE

Akram Elomari, Larbi Hassouni, Abderrahim Maizate (2017). The Main Characteristics of Five Distributed File Systems Required for Big Data: A Comparatively Study. Advances in Science, Technology and Engineering Systems Journal, vol. 2, No. 4, pp. 78-91.

Burleson (2017). Oracle Cluster File System (OCFS) Tips. Available at http://www.dba-oracle.com/disk_ocfs.htm

Carlos Fernando Gamboa (2008). Atlas LCG 3D Oracle cluster migration strategy at BNL, Gris Group, RACF Facility, Brookhaven National Lab, WLCG Collaboration Workshop. Available at http://slideplayer.com/slide/8285174/

D. Fetterly, M. Haridasan, M. Isard, and S. Sundararaman, (2011). TidyFS: A Simple and Small Distributed File System, in USENIX ATC'11, Available at http://research.microsoft.com/pubs/148515/tidyfs.pdf .

DataFlair Team, (2017). Hadoop HDFS Architecture Explanation and Assumptions. HDFS Tutorials. Available at https://data-flair.training/blogs/hadoop-hdfs-architecture/.

Feiyi W. Mark N. Sarp O. Dong F. (2013). Ceph Parallel File System Evaluation Report. Oak Ridge National LaboratoryOak Ridge, Tennesse.

Felix Hupfeld, Toni Cortes, Bj¨orn Kolbeck, Jan Stender, Erich Focht, Matthias Hess, Jesus Malo, Jonathan Marti, Eugenio Cesario. (2008). The XtreemFS architecture – a case for object-based file systems in Grids, Concurrency And Computation: Practice And Experience Concurrency Computat.: Pract. Exper.: 8:1–12.

Giacinto Donvito, Giovanni Marzulli, Domenico Diacono (2014).Testing of several distributed File-systems (HDFS,Ceph and GlusterFS) for supporting the HEP experiments analysis. Journal of Physics: Conference Series 513 (2014) 042014 doi:10.1088/1742-6596/513/4/042014.

Hooman Peiro Sajjad and Mahmoud Hakimzadeh Harirbaf. (2013). Maintaining Strong Consistency Semantics in a Horizontally Scalable and Highly Available Implementation of HDFS,Master Thesis, KTH Royal Institute of Technology.

Hurwitz J., Nugent A., Halper F. (2013). Big Data for Dummies. John Willey and Sons Inc, USA IBM (2018). Apache MapReduce. Retrieved May 7, 2018, from: https://www.ibm.com/analytics/hadoop/MapReduce

John Spray (2015). CephFS Development Update. Available at events.linuxfoundation.org/sites/events/files/slides/CephFS-Vault.pdf

Kaushiki, (2012). Google File System. Available at kaushiki-gfs.blogspot.com/

Knowledge Base (2018). About Luster file systems. Indiana University. Available at: https://kb.iu.edu/d/ayfh

Kuchipudi Sravanthi and Tatireddy Subba Reddy (2015). Applications of Big data in Various Fields. (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (5) , 2015, 4629-4632.

Madhavi Vaidya, Shrinivas Deshpande (2016). Comparative Analysis of Various Distributed File Systems & Performance Evaluation using Map Reduce Implementation, IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2016), December 23-25, Jaipur, India, pp.

Nader Gemayel (2016). Analyzing Google File System and Hadoop Distributed File System. Research Journal of Information Technology, 8: 66-74.

Pooja S.H. (2014). The Hadoop Distributed File System. International Journal of Computer Science and Information Technology, vol. 5(5), 6238-6243.

R.Vijayakumari, R.Kirankumar, K.Gangadhara R. (2014). Comparative analysis of Google File System and Hadoop Distributed File System. International Journal of Advanced Trends in Computer Science and Engineering, vol. 3 , No.1, pp. 553– 558.

Ray Walshe (2015). Google File System - DCU School of Computing. Available at https://www.computing.dcu.ie/~ray/teaching/CA485/notes/LectGFS.pdf.

Sage A. Weil, Kristal T. Pollack, Scott A. Brandt, and Ethan L. Miller (2004). Dynamic Metadata Management for Petabyte-Scale File Systems, Proceedings of the 2004 ACM/IEEE Conference on Supercomputing (SC '04), Pittsburgh, PA, November 2004.

Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D.E. Long, Carlos Maltzahn (2006). Ceph: A Scalable, High-Performance Distributed File System, Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI), Seattle, WA, November 2006.

Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung.(2003). The Google File System, SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles.

Shvachko, Kuang, Radia, Chansler (2010). The Hadoop distributed file system, Proceedings of the 26th Symposium on Mass Storage Systems and Technologies (MSST '10), Lake Tahoe, NV, pp. 1–10.

Shyam C Deshmukh, Sudarshan S Deshmukh. (2015). Simple Application of GlusterFs: Distributed file system for Academics. International Journal of Computer Science and Information Technologies, vol. 6 (3) , pp. 2972-2974.

Sunita Suralkar, Ashwini Mujumdar, Gayatri Masiwal, Manasi Kulkarni (2013). Review of Distributed File Systems: Case Studies, International Journal of Engineering Research and A pplications (IJERA), vol. 3, Issue 1, pp. 1293-1298.

Torben Kling Petersen (2018). Inside The Luster File System - An introduction to the inner workings of the world's most scalable and popular open source HPC file system Technology Paper, Seagate. pp. 1-14.