Challenges of Different Types of Distributed File Systems

Pragathi Thammaneni, Sridevi Mallipudi

School of Computing and Engineering

University of Missouri Kansas City

***Abstract*— With the massive increase in internet applications the need for storing huge volumes of data has rapidly increased over the past years. To process the immense amount of data, a framework is needed for that Distributed File System provides a centralized storage for data. As the amount of data storage is continuously increasing, the problem of providing efficient and fault tolerant solution arises, and this problem can be overcome with use of DFS. In this paper, an overview about how distributed filesystem is effectively used in network applications and the issues related to design of DFS such as consistency, transparency, replication, fault tolerance and few others are discussed in detail.**

***Keywords*—** *Distributed file systems, Fault tolerance, Consistency, Transparency*

1.OBJECTIVE

To analyze and interpret the current challenges related to design of Distributed File Systems.

2.MOTIVATION

Distributed File System plays a crucial role in managing huge amounts of data on network. In this paper, we discuss evolution of various file systems and various challenges of DFS.

3.INTRODUCTION

Distributed File System is a client-server-based application that stores the data at server and allows client to access it. The requirement for storing huge amounts of data has increased over recent years, the local file systems could not handle the immense amount of data. DFS allows sharing of large amounts of data in a secure and reliable manner over a network of nodes. DFS is a collection of independent nodes that appears to provide centralized file system view. DFS don't directly process or compute the data, but allows clients or applications to store and share data.

When user access or request a file on the server, server sends a copy of the file to the user which is cached in user local machine until the file processing is completed, then the file is returned to the server. In DFS, group of shared folders located at different locations are connected transparently by one or more DFS namespace. DFS namespace is the hierarchical view of the shared folders from different servers. All the files are organized in a global directory so the remoted data can be accessed from different locations. Since the access to the data is done simultaneously by one or more clients at the same time, different mechanisms are in place to provide the current latest version of the data.

4.ARCHITECTURE

There are four types of architectures in distributed file system there are:[1,2]

4.1 Client-Server Architecture

It provides a uniform view of local file system that has a communication protocol which allows clients to access files on server. This makes collection of processes running on different operating systems to share a common file system. The advantage with this system is it is largely independent of local file system. The main issue with this system is it cannot be used in MS_DOS due to its short file names.

4.2 Cluster-Based Distributed File System

It consists of single master server along with multiple chunk servers each divided into 64 MB. The advantage with this system is it allows single master server to control hundreds of chunk servers. One of the example is Google File System. The three important features of clustered-based DFS are Decoupled metadata and data, Reliable Autonomic Distributed Object Storage, and Dynamic Distributed Metadata Management.

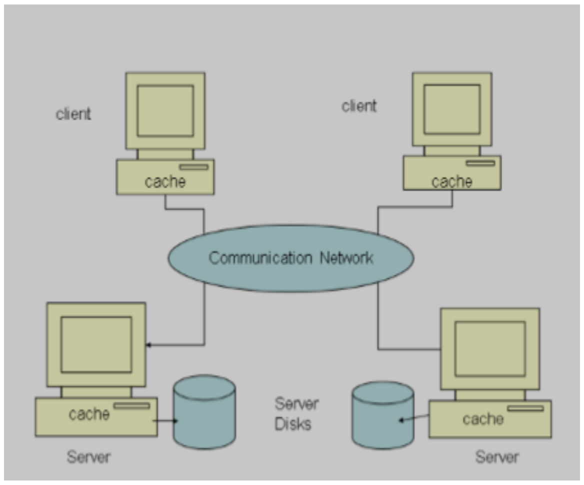

Fig 1. Client-Server Architecture

4.3 Symmetric Architecture

It is based on peer-to-peer technology. It uses key based lookup mechanism combined with DHT based system for distributing data.

4.4 Asymmetric Architecture

In this file system, one or more dedicated managers are present to maintain the file system and its associated disk structure. Supports parallel node access of file at the same time i.e., concurrent read writes operations.

5. DESIRABLE FEATURES OF DFS

Distributed file system provides some of the features listed below:[1,2]

5.1 Transparency; data is migrated between different nodes for improving load balancing. Users must be unaware of the location of services and transfer of data from remote location to local machine.

5.2 Replication; is done for achieving reliability and for increasing performance. Data is usually moved to the different locations closer to client. For reliability, data is duplicated and copied to different nodes to protect against failures.

5.3 Scalability; new nodes can be added to existing system for increasing the DFS capacity.

5.4 Reliability; system is made available to client all the time by using connection-oriented protocols.

5.5 Fault Tolerance; the distributed system should respond appropriately in the face of failure.

5.6 Consistency; different clients must see same directory and file contents at same time.

5.7 Security; user authentication and secure communication must be provided.

6. LITERATURE REVIEW

Some of the existing and new techniques of Distributed File Systems are discussed below:

6.1 Lazy replication technique in HDFS:

In HDFS, fault tolerance is handled by using data replication, where each block of data is copied and stored on multiple DataNodes. The existing implementation of replication in HDFS is performed in pipelined manner which takes much time for replication.

To improve the throughput, an alternative technique for efficient replica placement, called Lazy replication technique, has been proposed. In this technique, the block of data from client is written to the first DataNode, which will send the acknowledgements directly to client without waiting for receiving acknowledgement from other DataNodes.[3]

- Increased throughput and improved data availability.
- The drawback with this technique is efficient storage utilization.

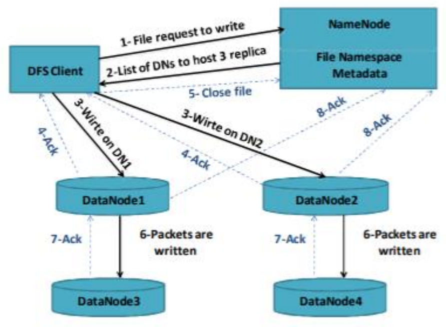

Fig 2. Writing a File on HDFS using Lazy Replication technique

6.2 Evolution and analysis of Distributed File System in Cloud Storage

In this paper, GFS and HDFS file systems on the basis of limitations with respect to the cloud paradigm and discussed proposed solutions for these limitations.

Distributed File System is the most popular file system used for the cloud computing. Both GFS and HDFS are scalable distributed file systems designed to handle large scales of data. But they are not considered suitable for

- Handling files with small sizes
- Frequently changing data
- Low latency data access

Using HBase on the top of HDFS can solve the problem of low latency access to small files from the large data chunks.

Recommendations are proposed for future work to improve caching techniques and to reduce MetaData storage and low latency data access. [4]

6.3 Load Balancing in distributed File System

Minimum response time is one of the factor for measuring the performance of a distributed file system. This can be improved with good design of load balance algorithm. There are static as well as dynamic load balancing strategies are available like Self-acting, load balancing (SALB) for parallel file system, Adaptive loading data migration (ALDM) in distributed file system. [5]

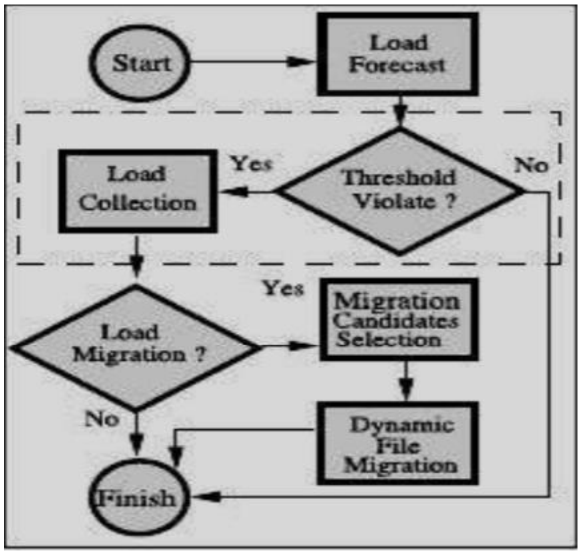

Fig 3. SLAB Flow Chart

In SALB, Online load prediction and load collection algorithm is used to predict the future loads from historical trends. Based on the results Distributed load balancing decision for file migration is taken.

- This distributed load balancer helps to improve scalability of the system.
- This mechanism is based on throughput of each node which may be weak in the case of Hotspot problem.

In ALDM algorithm migration of file to balance the load of the file system is done by calculating network load, disk capacity load, disk I/O load and average load. From the results obtained cost performance ration to decide destination node is calculated.

- Every node sends load to a centralized node for centralized decision of load migration, to some extend it solves the hotspot problem.
- The performance can be affected with failure in centralized node.

7. COMPARISON OF DISTRIBUTED FILE SYSTEMS

The motivation behind comparison is to explore various characteristics of different distributed file systems. This comparison analysis would help the customers to find the most appropriate and suitable file system.

7.1 Architecture

The following are the different DFS architectures that exists Client- Server Architecture, Cluster-Based Distributed, Symmetric Architecture, Asymmetric Architecture, Parallel Architecture. In detail explanation about each type of architecture is given in section 4 of the paper. [1]

7.2 Processes

A process can be either stateful or stateless. A stateful process saves the information about the client session and uses the information next time when a client makes a request. In stateless process, server does not maintain the information about client session.

7.3 Communication

In DFS, Remote Procedure Call(RPC) is used as a communication method as they make the calls independent to the network, operating systems and transport protocols. In RPC approach, there are there are two communication protocols TCP and UDP.

7.4 Naming

Naming plays a crucial role in distributed systems as each object has an associated physical address and associated logical path name. This address of objects is used to access the objects for retrieving the information. In DFS, complete transparent access to a remote file system is provided to clients. There are two approaches to employ this such as *central metadata server* to handle file namespace. Other approach is *metadata distributed to all nodes* this results in disk structure visible to all the nodes.

7.5 Scalability

Scalability is an attribute that describes the ability of a system to grow and manage increased demands. In a distributed system, there can be many CPU's getting added every day. An efficient DFS must be able to handle those needs. DFS systems can be designed in two ways: one is centralized and the other is decentralized architecture. In Centralized architecture scaling up needs more administration in distributed file system, whereas in decentralized architecture scaling can be managed by an administrator itself. [6]

7.6 Synchronization

In DFS, the semantics of read and write are important aspects when sharing a file between multiple users. Apart from semantics there are few approaches available they are file locking system, hybrid approaches, and atomic transactions.

Atomic Transaction is a transaction that either completely occurs or completely fails to occur.

File Locking System it is based on write-once-read-many and Multiple producer/single-consumer access model. Based on these models some systems choose to perform all operations synchronously on the server and some choose to give locks on objects to clients.

Hybrid Model it uses leases to lock an object to the client and controls parallel access to distributed file system. [8]

7.7 Consistency and Replication

To achieve consistency, most DFS systems use checksum to authenticate the data after sending through the communication network. Furthermore, replication and caching plays a crucial role in DFS. Replication is a process of storing data in more than one location this improves data availability and data consistency is maintained using caching. This can be done as client-side caching and server-side replication. Client-side caching verifies whether the data cached is valid and validates if the requested content is served with the desired information. In server-side replication every operation is notified by the server for example file modification. There are two types of replication should be considered in DFS such as data object replication and metadata replication. Availability and recoverability of data is maintained in DFS by providing proxy of metadata server with transaction logs along with snapshots of metadata. [6,7]

7.8 Fault Tolerance

Fault-tolerant techniques like replication and redundant arrays of inexpensive disk (RAID) are used in DFS. Transparency of failure and availability is provided using replication. There are two approaches for fault tolerance *failure as exception* and *failure as norm*. In failure as exception the failure node is eliminated so that system can be easily recoverable to last normal running state. Failure as norm systems replicates all kinds of data and replication is executed whenever replication ratio is unsafe. [4,5]

7.9 Security

Authentication and access control are the key security issues in DFS. Security are provided in DFS using authentication, authorization and privacy. Whereas some DFS does not employ security mechanisms based on trust between the nodes and client. [2]

8. FINDINGS

Based on the comparative analysis with different parameters the following findings can be helpful to select the appropriate DFS according to the requirements.

GFS:

GFS provides reliable and efficient access to data using large clusters-based architecture of commodity hardware. Files are divided into fixed-size g4 bit chunks. GFS provides chunk and file handling mechanism which make it stateless with fast, simple recovery. GFS implements TCP connections for communication. To achieve reliability each block of data is replicated on multiple chunk servers. Simultaneous access to same chunk of data is handled by locking mechanisms. Re-replication and rebalancing allows to spread replicas across racks to reduce correlated failures. GFS is fault tolerant and records high component failure rates due to inexpensive commodity, replicates the data to multiple chunk servers. Google has different datacenters in various locations which are most unmarked for protection. [9]

HDFS:

HDFS is open source version of GFS which runs on clusters with thousands of nodes which is a cluster-based architecture, it is specially designed to be highly fault tolerant. HDFS takes the data and breaks the information down in to separate blocks which can be distributed as different nodes in cluster, uses write-once-read-many access model for files. HDFS follows RPC based protocol on top of TCP /IP. HDFS provides high throughput access to data and is especially designed for applications that store large data sets. HDFS replicates each piece of data among multiple times and distributes among nodes, placing at least one copy on a different server rack. HDFS security is based on POSIX model. [3,11,15]

CFS:

Cassandra is open source, scalable, distributed, column-oriented database. CFS provides peer to peer distributed architecture where all nodes will be same which can handle the requests in efficient manner and fault tolerant. CFS uses a gossip protocol to discover state and location information about other nodes involving in the Cassandra cluster. It is non-relational and uses Big Table Data Model. Cassandra allows low latency operations with asynchronous master less replication. But the limitations with CFS is authorization is implemented at client – side. Cassandra supplies, “always on” availability with no single point failure. CFS is reliable and failure resistant because it applies replication and allows choosing the required replication parameters. Cassandra initiates a key space read repair to update inconsistent data.

NFS:

NFS was one of the most widely used distributed file system which follows the client -server architecture and uses RPC network services. Initially completely stateless which is operated over UDP, but modern implementations use TCP/IP which make its stateful. It is highly compatible with multiple platforms and operating systems. Server instantiates NFS volume for server file system / replication on top of remote local directories. Local hard drives managed by concrete file systems (EXT, ResierFS). As server code runs entirely in user space, so that running several copies will produces delay in synchronization issues. Limitations with NFS is performance is affected in case of network traffic and has security issues. [10]

AFS:

AFS is developed to run on UNIX operating system which is stateful with low network load and transparent. AFS use Remote procedure call method to communicate which make the underlying operating systems independent and supports TCP protocol. AFS is highly scalable which supports 50,000 clients at enterprise level with at most high security. Kerberos authentication provides rich set of access control bits than UNIX [11]. Writes/Reads operate on locally cached copy. High availability through replicas allows read-only copies of filesystem volumes. Copies are guaranteed to be atomic checkpoints of entire file system at time of read - only copy generation. It also employs client-side caching with proactive cache-invalidation. But drawback with AFS is complex and difficult to setup.

Lustre:

Lustre is a parallel distributed file system used for large -scale cluster computing. Metadata server that has one or more metadata target stores namespace, such as filenames, access permissions and file layout. Lustre is recognized as a stateful system by having some focus on resiliency because this system is big, and clients died all the time. Lustre Network (LNet) can use several types of network interconnects for communication includes InfiniBand verbs, TCP/IP on Ethernet, Omni -path and other network technologies Cray Gemini interconnect [12].It is an open-standard based system Commonly used for computing large scale clusters. It has compatibility with network components and is suitable for general purposes file systems. But it is currently available for only Linux. It also improves the metadata server code which provides hybrid locking to remove the internal locking bottlenecks on nodes with many CPU cores. Lustre uses Shared Secret key security flavors the same mechanism as Kerberos to provide client and server authentication, RPC message integrity and security encryption.

Panasas:

Panasas file system uses parallel and redundant access to object storage devices (OSDs), per-file RAID, distributed metadata management, consistent client caching, file locking services, and internal cluster management to provide a scalable, fault tolerant, high performance distributed file system. Implements the TCP/OSD protocols for the communication. It is suitable for general purposes file systems, implements file system in hardware. Panasas uses ActiveScale Storage cluster to improve storage utilization. Cluster scales dramatically in both performance and capacity [13]. Panasas object-based storage provides additional scalability and security over block -based SAN systems. Clients are responsible for writing data and its parity. Component objects include file data and file parity for reconstruction. This file attributes are replicated with two component objects on the server side. Fault tolerance is based on a local transaction log that is replicated to a backup on different manager node. Security mechanisms of OSD protocol let us enforce access control over files as client access storage nodes directly.

Table 1: Comparative analysis of different Distributed File Systems

File System	**Architecture**	**Processes**	**Communication**	**Naming(**metdata)	**Synchronization**	**Consistency and Replication**	**Fault tolerance**	**Security**
GFS	Clustered	Stateless	RPC/TCP	Central server	Client objects	Server-side	Norm	No
HDFS	Clustered	Stateful	RPC&TCP/IP	Central server	Client objects	Server - Side	Norm	Yes
CFS	Peer to Peer	Stateful	Gossip	inode column family	Synchronous/ asynchronous	Key Space	No Single Point	Yes
NFS	Client-server	Stateful	RPC/TCP	Remote - local directories	Delayed commit	Server-side	Exception	No
AFS	Client-server	Stateful	RPC/TCP & IP	Identical	Server lock	Client-side	Exception	Yes
Lustre	Clustered	Stateful	Independent	Metadata server	Hybrid locking	Server-side	Exception	Yes
Panasas	Clustered	Stateful	RPC/TCP/OSD	Central server	Client objects	Server-side	Exception	Yes
PVFS2	Clustered	Stateless	RPC/TCP/IP &BMI	All nodes	Metadata Locking	No	Exception	Yes
KFS	Clustered	Stateful	RPC/TCP	Central server	Client objects	Server-side	Norm	No
RGFS	Clusterd	Stateful	RPC/TCP	All nodes	Client objects	No	Exception	Yes

GFS – Google File System HDFS – Hadoop Distributed File System CFS – Cassandra File System NFS – Network File System AFS – Andrew File System PVFS2 - Parallel Virtual File System KFS – Kosmos File System RGFS -Redhat Global File System
RPC /TCP – Modern Communication mechanisms that most DFs follows
Security mechanisms - Authentication, authorization, encryption, data auditing (Dedicated -Yes/No Dedicated-No)

PVFS2:

PVFS2 is a virtual parallel file system for Linux clusters. Provides an interface for the nodes in the system to communicate with the storage device management. For network communication a Buffered Messaging Interface (BMI) is used along with the general protocols like RPC/TCP.PVFS2 endures restrictions introduced by TCP/IP independence, such as limits on the number of simultaneous open system sockets and network traffic overhead inherent in the TCP/IP protocol. PVFS2 is best suited for I/O-intensive applications. PVFS2 allows all nodes to behave typically. A single node will serve as a management node, while a subset of the nodes will be compute nodes and another subset will serve as I/O nodes. It is also possible to use all nodes as both I/O compute nodes. In PVFS2 synchronization is achieved with file or metadata locking [14]. PVFS2 can be fault tolerant and reliable Linux cluster when it adopts to provide more access security, data redundancy and management node redundancy. PVFS2 provides a rather unsophisticated security model, which is intended for protected cluster networks. [15]

KFS:

Kosmos Distributed File System (KFS) provides high reliability and availability which is C++ implementation of the Google file system which includes the functionality of GFS. Communication is done via basic RPC/TCP protocols. KFS is mostly used for data intensive apps such as data mining, search engines, grid computing etc. KFS has been deployed in production settings on large clusters to manage multiple petabytes of storage. [15]Replication is to provide availability due to chunk server failures. Generally, in KFS replications is done in three ways which is per file degree of replication, re-replication and re-balancing. KFS follows standard model when an application creates a file, the filename becomes part of the central server file system namespace. For performance writes are cached at the KFS client's library. Leases are used to support cache consistency.

RGFS:

Red Hat GFS allows a cluster of Linux servers to share data in a common pool of storage. RGFS is suitable for general purposes file systems. It is open-standard system that has great compatibility with networking components. But is currently available only for Linux Operating system. A cluster file system like Red Hat GFS can be used with an IP network block -sharing protocol like iSCSI to provide scalable file serving at low cost and follow the modern communication TCP protocol. RGFS provides a single consistent view of the file system namespace across all the nodes. Two GFS file systems are available with Red hat cluster suite -GFS/GFS2 [16]. RGFS scale clusters seamlessly, adding storage or servers on the fly. As data is shared in a common pool reduces the need for redundant copies of data. Provides decrease over all storage needs by reducing data duplication. RGFS data infrastructure implies backup and disaster recovery tasks.

.

9. RECOMMENDATIONS

From the comparative table, we can notice that each of the distributed file systems has both advantages and disadvantages based for certain parameters.

The best recommended file systems based on analysis with different parameters are HDFS and GFS both are similar in many aspects, but GFS provides better error monitoring, auto-recovery compared to Hadoop.

PVFS2 has the poor performance as it does not handle the challenges such as synchronization and replication.

10. CONCLUSION

In summary, DFS is the most important and widely used form for file sharing through which multiple users can store and share resources. Synchronization, Consistency, Replication, Fault tolerance and security are the key issues that need to be taken into consideration while designing the DFS. In this paper, a comprehensive survey of various file systems features, and comparative analysis of their characteristics are presented.

11. REFERENCES

[1] Akram Elomari, "*The main characteristics of five distributed file systems required for big data: A comparative study*", RITM-ESTC / CED-ENSEM, University Hassan II, ZIP Code 8012, Morocco,2017.

[2] Akram Elomari, "*The main characteristics of five distributed file systems required for big data: A comparative study*", RITM-ESTC / CED-ENSEM, University Hassan II, ZIP Code 8012, Morocco,2017.

[3] Eman S.Abead, Mohamed H. Khafagy, Fatma A. Omara , "An Efficient Replication Technique for Hadoop Distributed File System", International Journal of Scientific & Engineering Research, Volume 7, Issue 1, January-2016

[4] Dharavath Ramesh, "*Evolution and Analysis of Distributed File Systems in Cloud Storage: Analytical Survey* ", Department of Computer Science and Engineering Indian School of Mines, Dhanbad, India,2016.

[5] Shyam C. Deshmukh, " *A Survey: Load Balancing for Distributed File System*," ME Computer SP Pune University PCCOE, Pune,2015.

[6] Yongwei Wu, " *Modeling of Distributed File Systems for Practical Performance Analysis*," 2014.

[7] Satyanarayanan, M., "*A Survey of Distributed File Systems*," Technical Report CMU-CS-89-116, Department of Computer Science, Camegie Mellon University, 1989

[8] L.Sudha Rani, *"Distributed File Systems: A Survey",* Assistant Professor, Computer Science Department, GPREC, Kurnool, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3716-3721

[9] Ghemawat, S., Gobioff, H., Leung, S.T., "*The Google file system*", ACM SIGOPS Operating Systems Review, Volume 37, Issue 5, pp. 29-43, December 2003

[10] Omkar Kulkarni "*Study of Network File System(NFS) And Its Variations*"- IJERA, 2018

[11] Monali Mavani, "*Comparative Analysis of Andrew Files System and Hadoop Distributed File System*", May 2013

[12] Sun MicroSytems "*LUSTRE™ FILE SYSTEM High-Performance Storage Architecture and Scalable Cluster File System"* White Paper, December 2007

[13] Brent Welch "Scalable Performance of the Panasas Parallel File System "San Jose, California — February 26 - 29, 2008

[14] Yu, W., Liang, Sh., Panda, D.K., "*High performance support of parallel virtual file system (PVFS2) over Quadrics*", Proceedings of the 19th annual international conference on Supercomputing, pp. 323-331, 2005.

[15] Wittawat Tantisiriro, "*Data-intensive file systems for Internet services",* CMU-PDL-08-114 October 2008

[16] Tony Brown, "*Optimizing Red Hat GFS2® on SAS® Grid*" , SAS1929-2018