Study and Experimental Setup of Testing of Performance Parameters on Various Distributed File Systems

Madhavi Vaidya
Asst Professor, Computer Science Department
VES College, Mumbai, India

Shrinivas Deshpande, PhD
Associate Professor and Head
Computer Science Department
HVPM, Amravati, India

ABSTRACT
Distributed File System (DFS) is acting as an extension to file system which manages files and data on multiple storage devices and provides more performance and reliability using various modern techniques. Outside world only sees the distributed file system as a single storage device and it is nothing but an interface to a great extent. In case of failure or heavy load very few Distributed file systems provide location transparency and redundancy to improve the data availability. Significant challenges for such a distributed file system are extended to a large number of storage nodes and providing reasonably. In this paper, there is been a few performance collaborative parameters have been studied viz. replication, fault tolerance and load balancing experimentally on various DFS.

Keywords
DFS; Hadoop; Ceph; Balancing

1. INTRODUCTION
In this paper, there is a summary given on the literature and on the glitches in connection of two or more file systems to one another, in addition to few collaborative parameters have been studied here, named Fault Tolerance which means if any node goes down then how to perform recovery, along with few more performance parameters such as replication and Load balancing have been elaborated using implementation of those parameters using commands and their explanation of respective file systems. As a part of studying the data analysis on various distributed file systems, the study done and implemented various distributed file systems viz. Hadoop Distributed File System, Ceph File system, Glusterfs and zfs.

1.1 Glusterfs - GlusterFS is a scalable network file-system suitable for the bulky data processing tasks and this data processing can be done on cloud storage. GlusterFS is free and open source software. It is an open source distributed file system which provides replication over multiple storage nodes and it uses user space, i.e. File System in User Space and is known as **FUSE**. As a part of implementation, the nodes have been mounted on various nodes, which are combined into storage volumes using fstab in CentOS.

1.2 Hadoop Distributed File System – Hadoop Distributed File System is based on the Google File System. The main components in HDFS are the NameNode that manages the HDFS namespace and a collection of DataNodes that store the actual data in HDFS files. MapReduce utilizes the Google File System(GFS) as an underlying storage layer to read input and store output[[1]. MapReduce, which has been popularized by Google, is a scalable and fault-tolerant data processing tool that enables to process a massive volume of data in parallel with many low-end computing nodes [2,3].

1.3 Ceph file system - Ceph is designed to be a fault-tolerant, scalable storage system. In CephFS, the Metadata Server (MDS) plays a role in solving this problem. Metadata management is completely distributed, using a cluster of MDSs to handle metadata request from clients. The operation is adapted dynamically based on the workload generated by the clients (e.g., moving and replicating metadata depending on how often a file is accessed). Ceph cluster has POOLS, pools are the logical group for storing objects. These pools are made up of Placement Groups were the data is placed from the nodes in the cluster. Ceph is based on an object storage paradigm, where file data is stored by object storage devices (OSDs) and metadata is stored by metadata servers (MDSs). In comparison with few other distributed file systems, might rely on 'dumb' OSDs, the Ceph OSDs have responsibilities for data migration, replication and failure handling and communicate between each other.

1.4 zFS is designed as a distributed file system that offers comprehensive scalability by separating storage management from file management. It has been built in 20014 by Sun Microsystems which is free and open source. It provides the logical volume manager for using it in their Solaris Operating Systems [4]. Storage management is carried out using Object Store Devices (OSDs), and file management is distributed over a set of cooperative machines [5] All storage allocation and management in *zFS* is delegated to the OSDs. When a file is created and written to, the data blocks are sent to the OSD, which allocates space on the physical disk and writes the data on the allocated space.

2. LITERATURE REVIEW
Data replication consists of maintaining multiple copies of critical data, called *replicas*, on separate computers. It is a critical enabling technology of distributed services, improving both their availability and performance. Availability is improved by allowing access to the data even when some of the replicas are unavailable. This paper surveys and displays the replication fundamentals of each of the distributed file system along with balancing techniques used by them. In general, load balancing provides an ability to avoid the situation where some resources of the systems are overloaded while others remain idle or under loaded. It is well understood that excessively overloading a portion of resources

substantially reduces the overall performance of the systems. Existing load balancing approaches are classified into two broad categories-Static and Dynamic.

Glusterfs is an open source distributed file system which provides replication over multiple storage nodes and it uses user space, i.e. File System in User Space and is

Known as **FUSE**.

Replication simply takes a file and stores multiple replicas of it on multiple bricks. The failure of a brick is transparent for the user with a replicating volume except where it affects the last remaining brick. The user can continue working while the repair procedures take place. After the work is completed, GlusterFS can reintegrate the brick and then automatically start data synchronization.

A large amount of work has been done for static load-balancing schemes that do not rely on the current state of hosts. Based on this study, Kim and Kameda [6] have proposed two static load balancing algorithms, which are quite effective to improve system performance.

In contrast, the dynamic load balancing approaches provides an ability to improve the quality of load distribution at run time at a reasonable cost of communication and processing overhead. McCann et al [7] have studied a dynamic scheduling strategy, which is aware of resource requirements of submitted tasks. Their design is based on a centralized manager that handles scheduling on each processor. Condor [8] has been developed to harvest the idle cycles of a cluster of computers by distributing batch jobs to idle workstations. One of the very important goal of Condor is the guarantee that other clients will become available for their owners when the owners are about to access the machines. This goal is approached by detecting an owner's activity at an idle machine, and migrating background jobs to other idle machines when the owner begins accessing his/her machine. Even Condor has elaborated on making periodically checkpoints on tasks, thereby making it possible to restore and resume jobs in presence of software and hardware failures for reliability purposes. In addition, Condor offers a flexible mechanism where each machine's owner specify conditions under which the machine is considered idle. In centralized approach, the global state information is collected or estimated at a single host (server) which makes request task distribution decisions based on the collected information. This approach may impose fewer overheads for maintaining the state information, but has lower reliability. Failure of the central server makes load sharing inoperable.

3. STUDY AND EXPERIMENTAL SETUP GLUSTERFS

In case of Glustefs, the replicated volume stripes across bricks in a volume. This means that the each file is split up into multiple parts and stored on different bricks. Striped replicated volumes stripes data across replicated bricks in the cluster. For best results, striped replicated volumes can be used in highly concurrent environments where there is parallel access of very large files and performance is critical. The subtle difference between stripe-replication and distributed replication is that stripe replication will partition files and replicate the partitions of the files for added redundancy along with concurrency already present in striped volumes. On the other hand distributed-replicated drives simply replicate complete files. Glusterfs can handle failover mechanism very easily and simple manner using replicated Gluster. Making a copy of data in real time is replication which has been observed here. When the glusterfs file system is mounted from any one of the server, the server actually provides a file that contains details about all nodes taking part in the storage. When the glusterfs file system is mounted from any one of the server, the server actually provides a file that contains details about all nodes taking part in the storage. It has been experienced, this way the failover is pretty seamless, as if one of the server stops responding another node is selected from the cluster. The performance changes and depends on the kind of storage has been used in the Glusterfs installation. When the stripped volume was used, it has been experienced that the performance was much better as read and write will be distributed across nodes.As per given in the output attached, the files have been striped and replicated across nodes, they have been divided on gluster1 and gluster2 as the reads and writes will be distributed across the nodes. The two bricks are added, originally they were 2 in number and in later output there are 4 bricks. After implementation of rebalancing operation, it has been seen that the files have been automatically spread across new nodes which is seen in Figure 1 and 2. Performance depends upon the kind of storage has been used. It has been experienced that if a stripped volume has been used, it has been found that performance is much better as read and write will be distributed across nodes.

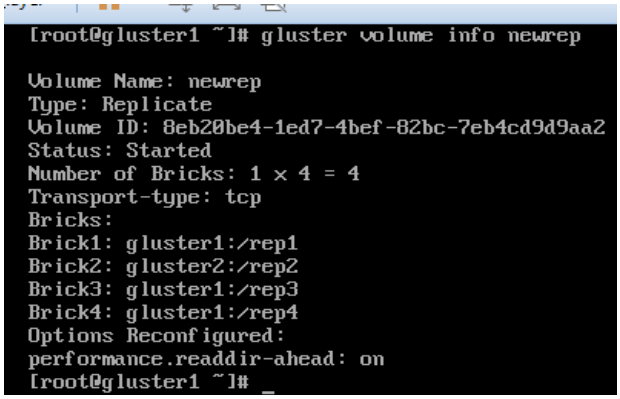

Fig 1: Creation of Volume and Bricks on Glusterfs

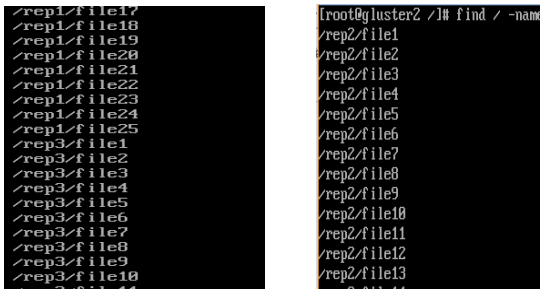

Fig 2: Stripe Replication on gluster1 and gluster2

#gluster volume rebalance distribute start

Rebalancing in Glusterfs is possible by rebalance statement

It has been experienced that, the files on gluster1 and gluster2 have been scattered properly with few files on gluster1 and remaining on gluster2.

3.1 Ceph File System–

Distributing metadata for balance in Ceph, tries to spread metadata evenly across the metadata cluster. The benefit of this approach is that clients can contact different servers for their metadata in parallel. CEPH RADOS is a very important aspect that manages the data storage as well as replication among the cluster nodes. It consists of two daemons: the

CEPH OSD Daemon that runs on each storage node as given in Figure 3 and handles the storage operations, and the CEPH Monitor that keeps track of the cluster state, i.e. the map of active nodes and their roles.

```
#to find the replication level

root@newceph22# ceph osd dump | grep –i pool-D

pool 5 'pool-D' replicated size 1 min-size 1….

root@newceph22# ceph osd pool set pool-D size 3

Set pool 5 size 3

#changing replication level

root@newceph31#ceph osd dump | grep –I pool-D

pool 5 'pool-D' replicated size 3 min-size 1…….
```

Fig 3 : Replication on Ceph

RADOS offers the possibility to divide the logical storage area into different pools, and allows pool-based access control mechanisms. RADOS automatically monitors usage statistics in order to perform load-balancing between the nodes composing the cluster

Many metadata balancers distribute metadata for complete balance by hashing a unique identifier, like the inode or filename[9]. Ceph is designed to handle two things: 1) it enables fault tolerance by distributing data (replicated or erasure-coded) across a cluster of nodes, and 2) it provides user access to the data [10].

Although conventional storage systems often use various tricks to ensure redundancy over replication, the subject of replication (in combination with high data availability) is almost inherently incorporated in the design at Ceph. It acknowledges the failure of a hard drive after a set time (default setting is five minutes) and then copies all missing

```
root@newceph22#ceph osd pool create pool-D
128

root@newceph22#ceph osd lspools

0 data, 1 metadata, 2 rbd, 3 pool-A, 4 pool-B,5-
pool-D

root@newceph31# ceph osd lspools

0 data, 1 metadata, 2 rbd, 3 pool-A, 4 pool-B,5-
pool-D
```

Fig 4 : Load Balancing in Ceph

objects and their replicas to other OSDs. This way, Ceph ensures that the admin's requirements are consistently met – except for the wait immediately after the failure of a disk.

Controlled Replication Under Scalable Hashing *(*CRUSH) is the algorithm that Ceph uses to determine how and where to place data to satisfy replication and resiliency rules. The CRUSH Map gives CRUSH a view of what the cluster physically looks like, and the replication rules for each node [11].

Ceph cluster has POOLs , pools are the logical group for storing objects. These pools are made up of Placement Groups. In our case, in the production environment, it is expected that at a minimum, there will be three Ceph nodes in a cluster. The output generated by them has been given in Figure 4.

This means all the objects of pool-D will be replicated 3 times on 3 different OSD's. Ceph is designed in such a way that, everything in ceph is designed in terms of objects. Hence ceph cluster known as Object Storage cluster. The objects are mapped to placement groups and their copies are scattered across different OSDs.

3.2 Zfs

ZFS offers superb data integrity as well as compression, raid-like redundancy and de-duplication. When data integrity is the priority the zfs is the solution as it is the file system which is ready to meet and take up the demands of huge redundant data volumes [12].

zFS is a scalable distributed file system that uses Object Store Devices (OSDs) for storage management and a set of cooperative machines for distributed file management. It offers extended scalability by separating storage management from file management. Storage management is done by OSDS [13].

Clone, Snapshot, and replication are the most powerful features of ZFS. Cloning is used to create a duplicate dataset, Snapshots are used to create point-in-time copies of file systems or volumes, and replication is used to replicate a dataset from one datapool to another datapool on the same machine or on two different machines the replicas in datapool's can be created between two or more different machines.

Snapshot is one of the most powerful features of ZFS, a snapshot which is executed and given in Figure 5, which provides a read-only, point-in-time copy of a file system

```
#zpool create mypool mirror /dev/sdb/ dev/sdc

#zfs list –r mypool

#zfs snapshot mypool/docs@version1

#zfs list –t snapshot

For replication

#zfs send mypool/docs@today | zfs receive
backuppool/backup

#ls /backuppool/backup
```

Fig 5 : creating snapshot on zfs

or volume that does not consume extra space in the ZFS pool. The snapshot uses only space when the block references are changed. Snapshots preserve disk space by recording only the differences between the current dataset and a previous version.

3.3 Hadoop Distributed File System

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
        <name>dfs.replication</name>
        <value>2</value>
</property>
<property>
        <name>dfs.name.dir</name>
        <value>file:///home/hadoop/hadoop/hdfs/nam
enode</value>
</property>
<property>
        <name>dfs.data.dir</name>
        <value>file:///home/hadoop/hadoop/hdfs/data
node</value>
</property>
<property>
        <name>dfs.replication</name>
        <value>2</value>
</property>
</configuration>
```

Fig 6 : Replication factor on Hadoop

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. There is a different scenario in Hadoop Distributed File System. HDFS has a

```
[root@h1g1 bin]# ./hadoop balancer
Time Stamp               Iteration#
Bytes Already Moved  Bytes Left To
Move  Bytes Being Moved
16/11/03 09:48:36 INFO
net.NetworkTopology: Adding a new
node: /default-
rack/192.168.25.129:50010
16/11/03 09:48:36 INFO
net.NetworkTopology: Adding a new
node: /default-
rack/192.168.25.247:50010
16/11/03 09:48:36 INFO
net.NetworkTopology: Adding a new
node: /default-
rack/192.168.25.245:50010
16/11/03 09:48:36 INFO
balancer.Balancer: 0 over utilized
nodes:
16/11/03 09:48:36 INFO
balancer.Balancer: 0 under utilized
nodes:
The cluster is balanced. Exiting...
Balancing took 4.817 seconds
```

Fig 7 : Node Balancing on Hadoop

Master Slave Architecture. HDFS cluster consists of a single NameNode. In addition, there are a number of DataNodes, usually one per node in the cluster. The cluster manages storage attached to the nodes that they run on. HDFS exposes allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes.

The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode. In case the JobTracker, which runs on namenode does not receive any heartbeat from a TaskTracker, which is executed on datanode for a specified period of time (by default, it is set to 10 minutes), the JobTracker understands that the worker associated to a specific datanode's TaskTracker has been failed [14].

The replication factor has been mentioned in the xml file which are situated in /conf directory of Hadoop directory as mentioned in Figure 6, performed Hadoop installation. Modifying the dfs.replication property in hdfs-site.xml will change the default replication for all files placed in HDFS.

The block size setting is used by HDFS to divide files into blocks and then distribute those blocks across the cluster. The load balancing on the hadoop nodes is done using balancer command, elaborated an experimental setup in Figure 7. Availability of the node is managed by maintaining multiple replicas of each block in an HDFS file, recognizing failure in a DataNode or corruption of a block, and having mechanisms to replace a failed DataNode or a corrupt block.

4. COMPARATIVE APPROACH

Distributed file systems like GlusterFS, zfs and HDFS can spread a single file system namespace across multiple servers. In fact, various set ups are made for those DFS on each three nodes of Virtual Machines on two different setups In Ceph's case, a single metadata server (MDS), maintained in working memory of a single node, keeps track of the data across all the storage nodes, each of which is managed by an object storage daemon (OSD). If a node falls out, or more nodes are added, the changes are managed by the MDS. First of all being that GlusterFS distributes files and works on top of existing file-systems. It is completely transparent to the system and applications.

Ceph and Gluster have similar data distribution capabilities. Ceph stripes data across large node-sets, like most object storage software. This aims to prevent bottlenecks in storage accesses. Advantages of Glusterfs includes a global namespace, compression on write. It can scale to petabytes with thousands of clients. Ceph supports both replication and easy to deploy. GlusterFS is easy and quick to set up a basic constellation, and it ships out-of-the-box with most distribution repositories. It's understandable very quickly by any regular Linux administrator.

5. CONCLUSION

In this manner, the comparison of various distributed file systems especially on collaborative and performance parameters such as Load Balancing, Fault Tolerance and Replication has been provided. The wide study is been provided, based on the concepts and the commands which are executed and experimental setup is displayed for all the listed distributed file systems.

In particular, Ceph looks quite promising when stability and performance issues will be solved, but currently Glusterfs

remains the best system in few performance tests which were carried out. The Glusterfs was the most easiest distributed file system to install and to work on it, able to make working file-system by simply adding a new volume to the persistent volumes and see the replicated data very easily on other nodes of cluster. ZFS offers superb data integrity as well as compression, raid-like redundancy and de-duplication. Hadoop Distributed File System instead appears the more stable and reliable storage system, performs quite well. The future plan is to perform the study on unstructured files on all listed distributed file systems.

6. REFERENCES

[1] S. Ghemawat et al, 2003, The Google File System. ACM SIGOPS Operating Systems Review, 37(5):29–43.

[2] J. Dean et al. 2008, MapReduce Simplified Data Processing on Large Clusters. Communications of the ACM, 51(1):107–113.

[3] Jeffrey Dean et al., 2004, Mapreduce: Simplified Data Processing on Large Clusters. In In Proceedings of the 6th USENIX OSDI, pages 137–150.

[4] Rodeh and Teperman, 2003,A Scalable Distributed File System Using Object Disks. Proceedings of the 20th IEEE Conference on Mass Storage Systems and Technology (MSS'03), San Diego, CA, pp. 207–218.

[5] http://www.lustre.org/docs/whitepaper.pdf

[6] C. Kim and H. Kameda, March 1992, An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems. IEEE Trans. Compute., 41(3):381–384.

[7] C. McCann, R. Vaswani, and J. Zarhojan, May 1993, A Dynamic Processor Allocation Policy for Multiprogrammed Shared-Memory Multiprocessors. ACM Transactions on Computer Systems (TOCS), vol. 11, no. 2, pp. 146–178.

[8] M. Litzkow, M. Livny, and M. Mutka, Condor, June 1988 A Hunter of Idle Workstations. International Conference on Distributed Computing Systems, pp. 104-111.

[9] Sevilla, Michael A., et al., 2015, Mantle: A Programmable Metadata Load Balancer for the Ceph File System. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. ACM.

[10]Petros Koutoupis, June 2016, http://www.linuxjournal.com/content/understanding-ceph-and-its-place-market,

[11]Weil, Sage A., et al. Ceph: A Scalable, High-Performance Distributed , 2006, File System. Proceedings of the 7th symposium on Operating Systems Design and Implementation. USENIX Association.

[12] Teperman, A. and A. Weit, 2004, Improving Performance of a Distributed File System using OSDs and Cooperative Cach. IBM Journal of Research and Development.

[13] O. Rodeh and A. Teperman, 2003, zFS, A Scalable Distributed File System using Object Disks. InProceedings of the IEEE Mass Storage Systems and Technologies Conference, pp 207-218,San Diego, CA, USA.

[14] Apache Hadoop Documentation