A Comparative Study on Performance of Hadoop File System with MapR File System to Process Big Data Records

Dr. T. Suryakanthi[1] and V. S. J. Pallapolu[2]

[1] Fellow, Faculty of Computing, Botho University
Gaborone, Botswana

[2] Lecturer, Department of Accounting and Finance, University of Botswana
Gaborone, Botswana

Abstract

Big Data is a buzz word heard everywhere and many organizations are generating huge amounts of data. The data is growing at faster pace. Variety of data stored is posing a new challenge for the organizations. Organizations need a new set of tools and techniques which can efficiently process, analyze and visualize the data for better decision making. Distributed systems developed by the developers can run on various nodes to an extent can solve the problem of data processing. Development of cloud applications is an advantage to the organizations to process the huge data on the cloud. Hadoop and its ecosystem will help to efficiently process the data by using commodity hardware. MapReduce is a framework for writing programmes for Hadoop system. Hadoop Distributed File System (HDFS) is the storage system for storing large data on commodity hardware. Hadoop file system still faces few challenges. Recently MapR has developed the MapR file system to distribute the large data sets and it overcomes the challenges faced by the Hadoop file system. In this paper we first study about the Hadoop file system, its limitations and then make a comparative study of MapR file system. Also we analyze how the MapR system is more efficient in distributing the data than Hadoop file system. We also analyze how MapR system overcomes the limitations of Hadoop File System.

Keywords: *Big Data, Cloud, Hadoop, HDFS, MapR.*

1. Introduction

Hadoop [1] prominence has grown a lot in recent years and it is considered as standard framework for big data analytics [2, 3]. One of the reasons for this is hadoop scalability feature. Hadoop File system provides good features for accessing large data. It also provides effective fault tolerance. It provides faster access to huge data. HDFS [4] is designed for sequential read of the data from beginning of each block. It is highly optimized for parallel sequential readers which reduces the seek time and speed up the process. But over a period of time it has been observed that there are number of limitations with respect to HDFS was observed. These limitations will be discussed in section III of this paper. So it is highly essential to have a new system which implements Hadoop API. MapR [5] has introduced entirely new architectural components which will resolve most of the limitations by HDFS. This paper is organized as follows. Section II focus on literature review on Hadoop File System, Section III discusses some of the limitations by HDFS, Section IV focus on MapR-FS. In Section V we compare the HDFS system with MapR system to analyze which system provides the efficient features in today Big data world. Analysis can be made to check whether the new system overcomes the limitations of HDFS. Section VI focus on architectural components of MapR-FS. Section VII results of our study is discussed. We conclude the study in section VIII.

2. Literature Review of Hadoop File System

Reliable data storage of very large data sets and to efficiently stream those data sets a file system called HDFS [4] is designed. Data and computation is distributed across many servers. One of the important characteristic of Hadoop is that it can scale to thousands of hosts and data can be partitioned in many nodes and computation is performed. Application data and file system Meta data is stored separately in HDFS. Name node is an dedicated server on which HDFS stores the metadata. Data Node is another server on which application data is stored. All the servers communicate each other using TCP [6] based protocols. Fig 1 shows the HDFS architecture. HDFS supports various operations on files like all the conventional file system. The major operations which are supported by hadoop includes read, write and delete files. It also supports to create and delete directories.

The data storage in the file system is the responsibility of Hadoop HDFS and HBase. Primary data storage is HDFS. There is an computation for the huge data analysis called MapReduce framework. This consists of a single master

job-tracker and one slave task-tracker per cluster node. This follows the master-slave technology. Master will schedule the jobs for the slaves and slaves execute the tasks as given by the master. MapReduce framework and HDFS run on the same set of nodes. The data is already present and tasks are executed on the scheduled nodes. Here we are moving the code to the data rather data to the code which allows for faster computation. There are many languages in Hadoop ecosystem which will enable the huge data transfer between the clusters.

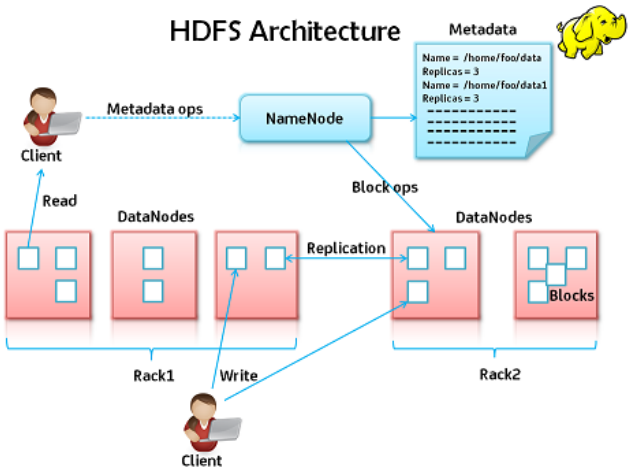

Fig 1. HDFS Architecture

The HDFS architecture is composed of data nodes and name node. Files and directories are hierarchically stored as HDFS namespace. They are represented by inodes. Various attributes like file permissions, file modification, access time, namespace and disk space is recorded by the inode.

Contents in the file are break into chunks and each chunk of file is replicated in the multiple datanodes. HDFS client is responsible for user applications to access the file system. Once the application starts the file operation like reading a file, a list of datanode is requested by the HDFS client through the namenode. Conventional file systems like NTFS, FAT systems do not provide any interface to find the location of file blocks. HDFS provides an Application Program Interface to locate the file blocks.

The file operations in HDFS like read and write can be implemented through single-writer and multi-reader mode. It means that once a file is created and data is written on it, the data cannot be altered or removed. We can write only the new data by appending to an existing file after opening that in read mode. HDFS contains blocks. Unique block with ID is created by the Namenode whenever required. It also lists replication datanodes.

3. Limitations of HDFS

The distributed file system like NFS [7] is built for general applications, while Hadoop file system is built for the specific applications. Hadoop system is designed for sequential file read. If there are four blocks of data and the data in the third block has to be read, it start from first block and read all the blocks in sequence. Random seeking is a difficult task to perform with HDFS. Mechanism for local caching of data is not provided with HDFS where as other file system provide it. In HDFS system once the data is written to a file it can only be read for several times. Once the file is created and closed it cannot be updated with a new data. Updates to the files after they have already been closed are not supported. Hadoop provides better recovery for hardware failures but the limitation here is the system performance is lost in proportion to number of nodes failed. Table 1 shows the limitations of HDFS.

Table 1: Limitations of HDFS

Feature	***Limitation***
Reliability	NameNode leaves data and performance vulnerable
Mutability	Write once/read only; Data immutable; Source data changes data needs to be reloaded into cluster
Block size	Same size for I/O, replication, sharding not optimized for different requirements
POSIX semantics	Must use ‘hadoop fs’ to access data
Availability	No snapshot or built-in mirroring capability
Scalability	NameNode only scales to 100M files
Performance	Written in Java and runs on block device

4. MapR File System

Limitations found in the Hadoop file system need to overcome with the advancement of technology. MapR has introduced an enterprise grade distribution framework for Apache Hadoop. This framework improves reliability, performance and usability. This file system provides complete Hadoop ecosystem which includes MapR-FS [8] the file system, Map Reduce [9] and MapR Control system user interface. Fig 2 shows the MapR distribution for Hadoop.

Data protection in MapR is carried out with a special feature called Snapshots [10]. This will enable us to rollback to known good data set. It is a read-only image of a volume which provides recovery by point-in-time. In order to increase the efficiency of cluster disk resources, snapshots store changes to data stored in volume. Snapshots can be created manually or can be automated.

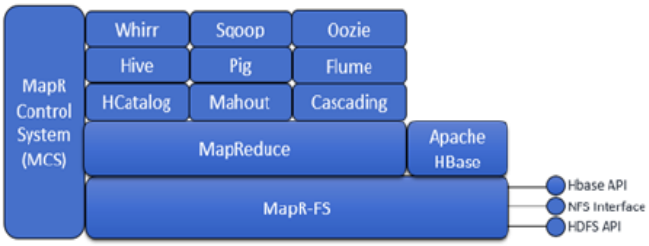

Fig 2. MapR distribution (Source: doc.mapr.com)

MapR provides security features in order to protect the data in the cluster. Users can be protected through various mechanisms like Kerberos, LDAP/AD, and NIS. To authorize the users MapR provides Access Control Lists for cluster. MapR also provides the wire level security to encrypt the data transmission for traffic within the cluster. In order to recover any data disasters, MapR provides a disaster recovery by providing built in mirroring. Between clusters we can create local or remote mirror volumes to mirror the data. These data mirrors can also be created in data centers or on public cloud [11]. The mirroring framework is shown in Fig3.

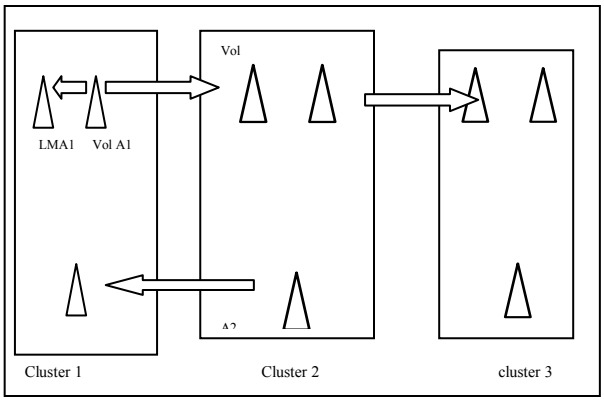

Fig 3: Mirroring for data recovery

Cluster 1 has the volume A1 data and local mirror of A1. It has also remote mirror for volume A2. Similarly cluster 2 will have remote mirror of A1, and volume A2 data. Cluster 3 will have remote mirror of volume A2, local mirror of Volume A3 and so on. Network file system protocol is used by MapR file system to enable read write data. Applications and various tools can access the MapR FS storage layer using NFS.

Graphical control panel for cluster administration [12] in MapR is provided by the MapR Control System (MCS) [13]. MCS dashboard will provide the detailed information about the clusters, the cluster utilization, disk space, and MapReduce jobs.

5. Comparative Study of HDFS and MapR-FS

In this section we look into detail comparative study of Hadoop Distributed File system with MapR-FS. First of all both are distributed file system. MapR is an implementation of HDFS API. The applications which are written for HDFS will also run with MapR-FS. Both the systems have replication factor which are resilient to failure. Let us now look into performance of each of the system. HDFS has Name node performance bottleneck whereas in MapR-FS as there is no name node, and location metadata is distributed across the cluster. It provides higher performance. In MapR-FS full read-write access is allowed. In HDFS it is not allowed. The block size in HDFS is split into equal 64MB. In MapR-FS the chunks are split into 256 MB each. The unit of replication with HDFS is block and each block is 64 MB. In MapR FS the unit of replication is container and each container holds 32 GB.

MapR-FS contains various such components which makes it efficiently process the data. It also overcomes the challenges faced by the Hadoop system.

6. Architectural Components of MapR-FS

In this section the architectural components of the MapR-FS is discussed. The MapR-FS architectural components are shown in the Table 2.

Table 2: Architectural components of MapR-FS (source:doc.mapr.com)

Component	***Description***
Storage Pools	These are group of disks to which MapR-FS writes the data
Containers	Files and directories stored as abstract entity
CLDB	Container location can be tracked by this service
Volumes	These are management entity stores and organize containers
Snapshots	It is a read-only image of a volume at specific point of time
Direct Access NFS	This enables application to read data and write data directly to the cluster

Storage pools are made up of number of disks that are grouped together. Each storage pool has 3 disks. There are multiple storage pools on each of the nodes. Containers are used to store data. These containers reside in storage pools.

7. Results

In this paper, we analyze how the MapR system is more efficient than Hadoop file system based on the study of these two systems. The systems are analyzed for efficiency in their availability, scalability and performance. In HDFS even though there is capability to provide the read-only images of the disks, it is not consistent. In MapR system snapshots are highly consistent and built in mirroring provided will help for load balancing and backup in case of any disasters. In HDFS the name node scales to 100 million files per cluster. In MapR-FS there is no limitation of files per cluster, but it is only limited by the availability of the disk space. HDFS is written in JAVA and it is slower as JVM needs to be always present. But MapR system is written in C program which provides high performance. In HDFS name node coordinates the files and directories and their contents which affects the performance, whereas in MapR-FS there is no name node and metadata for files and directories are fully distributed. Write performance is improved by write operations within storage pool across disks in MapR-FS.

8. Conclusion

The distributed file system and MapReduce computation model has been successful in handling massive data processing, Hadoop has gained more attention in industry for the big data. Hadoop is an open source framework which supports massive data processing. This enables distributed processing on large clusters using commodity hardware. Hadoop has major features like scalability, cost efficiency, flexibility and fault tolerance. Even though Hadoop has features which handle data efficiently, it also have some limitations. In this paper we have studied limitations by Hadoop system. We have also made a study between HDFS and MapR-FS a file system from MapR. We also studied the components of the MapR-FS, and how MapR-FS overcome the limitations of HDFS. We also analyzed the MapR-FS efficiency in terms of the scalability and performance. We conclude that MapR-FS is more efficient as it avoids the limitations faced by HDFS.

References

[1] T. White, Hadoop: The definitive guide, Sebastopol, CA, USA, O'Reily Meida, 2012.
[2] J. Manyika et.al, Big data: The next frontier for innovation, competition, and productivity, SanFrancisco, CA, USA: Mc.kinskey global institute, 2011. Pp 1-137.
[3] D. Fisher, R.D. Line, M.Czerwinski, S.Drucker, "Interactions with big data analytics", Interactions vol 19, no3, pp-50-59, May 2012.
[4] K. Shavachko, H. Kuang, S.Radia, and R. Chansler, "The Hadoop Distributed File System (HDFS)', Proc of MSST 2010, pp1-10, 2010.
[5] Map Reduce Architecture Guide available at doc.mapr.com/Architecture+guide
[6] Huston G., Telstra (2000), The future of TCP, The internet protocol Journal, Cisco Systems, Vol.3, No.3
[7] Russell Sandbaerg, "Design and Implementation or the SUD Network File system", Sun Microsystems.
[8] http://doc.mapr.com/display/MapR/Working+with+MapR-FS.
[9] J. Dean, S.Ghemwat, "MapReduce Simplified data processing on large clusters", Commun, ACM, vol 51, no.1 pp 107-113, 2008.
[10] The MapR Distribution for Apache Hadoop, extracted from www.mapr.com/sites/default/files/mapr_dist_ white_paper.pdf.
[11] Miller, M. (2009) Understanding Cloud Computing. Retrieved February 17, 2012 from www.informit.com/ articles/article.aspx?p=1321170
[12] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In EuroSys, 2010.
[13] Map Reduce Control System available at doc.mapr.com /display/MapR/MapR+Control+System

Dr. T. Suryakanthi earned her master's degree in computer applications in 2002 from Andhra University, Visakhapatnam, India and doctoral degree in 2014 from Lingaya's University, Faridabad, India. She has worked for around 2 years in software industry and has been teaching for 2 and half years. She was Assistant Professor of Computer Applications at Lingaya's University and is currently associated with Botho University, Gaborone, Botswana. She is a member of IEEE, ACM, IAEng. She has 11 research papers to her credit in international conferences and journals. Her current research interests include Artificial Intelligence, Natural Language Processing, Machine Translation, Big data analytics and Theory of automata.

Mrs. V. S. J. Pallapolu earned her master's degree in computer applications in 2008 from Acharya Nagarjuna University, Guntur, India. She is in teaching from the past 4 years. She is currently associated with University of Botswana, Gaborone, Botswana. She has not only taught in fulltime courses and also contributed to Distance programs in the University of Botswana .She has published a paper in International Journal and presented two (2) papers in international conferences. Her current research interests include big data analytics, business intelligence, and Information management.