Improving Efficiency in Handling Big Volume Data in Distributed File Systems

Mrs Swapna Vanguru , Mrs Dhomala Aswani
Assistant Professor,ACE Engineering College,Hyderabad.
Assistant Professor, ACE Engineering College, Hyderabad

Abstract: Now-a-days, Big Data has become an emerging technology in modern technical world. With the increasing usage of this enormous volume of data through online based services the growth of data is tremendous. As the size increases there comes the problem of handling the huge volume of data. The aim of this paper is to handle the large volume of data in an efficient way. For this we propose an *efficient distributed file system based on easy storage and fast retrieval method.* This method shows that it is much efficient and better than using a file system and is applicable for all file sizes and frequency of use.

Keywords – Big data, Distributed file system,Hadoop.

I. INTRODUCTION

The haddop framework provides the many required tools to implement big data. The Hadoop Distributed File System(HDFS) default file system used in hadoop framework. It provides support for other file systems to be used and other distributed file systems are easily understood the hadoop framework. This framework also provides a way to use different file system according to the usage .So we have particularly focused on storing and retrieving the data in to the file system and from the file system.

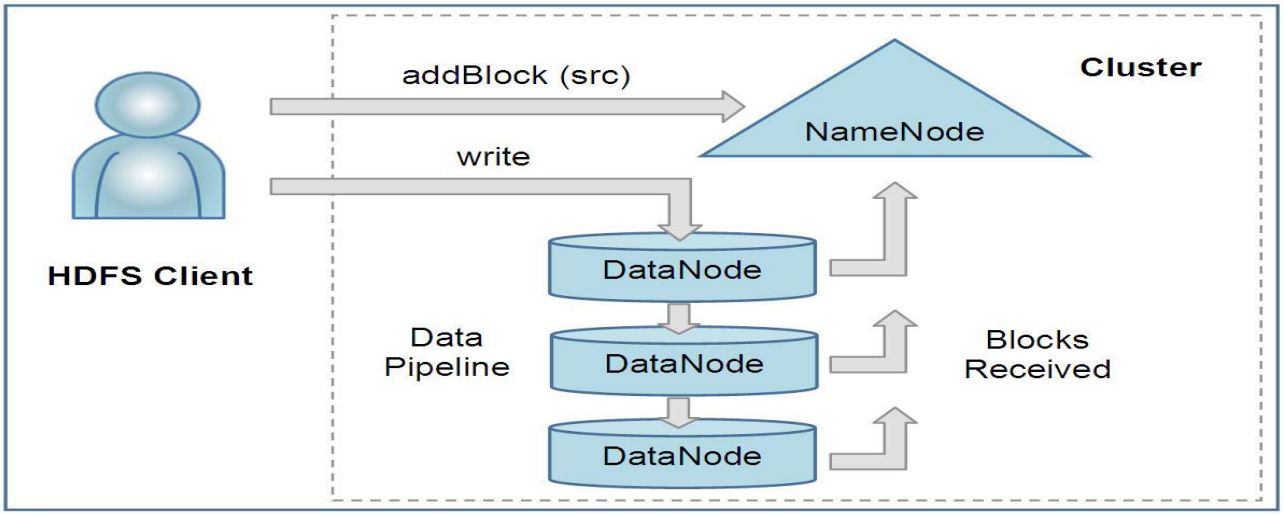
II. DISTRIBUTED FILE SYSTEMs

Normally a Distributed File System provides data storage over a distributed network environment. The main characteristics of DFS are transparency, fault tolerance and scalability. The other general characteristics include file mobility, network transparency, and user mobility. The design considerations include the name space, state full or stateless operation, semantics of sharing and remote access methods.

III. Hadoop Distributed File System

Now, HDFS stores metadata on a dedicated server, called the NameNode. Application data are stored on other servers called DataNodes. The architecture has a single server for namenode with data blocks distributed and replicated over the datanodes. The basic components are Namenode,Checkpoint node, Jobtracker node, Subordinate nodes, Tasktracker node and

Datanode.HDFS is optimized to provide support for very large file sizes ranging from gigabytes to petabytes.



A.NameNode :

Basically HDFS consists of files and directories. Files and directories are represented on the NameNode by inodes, which record attributes like permissions, modification and access times. The file content is split into large blocks (typically 128 megabytes, but user selectable file-by-file) and each block of the file is independently replicated at multiple DataNodes (typically three, but user selectable file-by-file depending on number of nodes). The NameNode maintains the namespace tree and the mapping of file blocks to DataNodes (the physical location of file data).

If HDFS client wants to read a file first contacts the NameNode for the locations of data blocks comprising the file and then reads block contents from the DataNode closest to the client. When writing data, the client requests the NameNode to nominate a suite of three DataNodes to host the block replicas. The client then writes data to the DataNodes in a pipeline fashion. The current design has a single NameNode for each cluster. The cluster can have thousands of DataNodes and tens of thousands of HDFS clients per cluster, as each DataNode may execute multiple application tasks concurrently.

B.DataNodes

Each block replica on a DataNode is represented by two files in the local host's native file system. The first file contains the data itself and the second file is block's metadata including checksums for the block data and the block's generation stamp. The size of the data file equals the

actual length of the block and does not require extra space to round it up to the nominal block size as in traditional file systems. Thus, if a block is half full it needs only half of the space of the full

block on the local drives.

DataNodes are not directly called by Namenodes. It uses replies to send instructions to the DataNodes. The instructions include commands to:

- replicate blocks to other nodes;
- remove local block replicas;
- re-register or to shut down the node;
- send an immediate block report

C BackupNode

A recently introduced feature of HDFS is the BackupNode. Like a CheckpointNode, the BackupNode is capable of creating periodic checkpoints, but in addition it maintains an in memory, up to-date image of the file system namespace that is always synchronized with the state of the NameNode.

The BackupNode accepts the journal stream of namespace transactions from the active NameNode, saves them to its own storage directories, and applies these transactions to its own namespace image in memory. The NameNode treats the BackupNode as a journal store the same as it treats journal files in its storage directories. If the NameNode fails, the BackupNode’s image in memory and the checkpoint on disk is a record of the latest namespace state.

IV. RELATED WORK

Various works for improving the storage efficiency and a better way to manage volume in big data are introduced but they have drawbacks. We are proposing a system with the name Efficient Hadoop Distributed File System (EHDFS) is one such attempt that increases the efficiency of HDFS for small files as well as for large files. The files are stored in one file is known as combined file on Datanode or client. The Constituent FileMap is most important in EHDFS for efficient management. Indexing mechanism is used to access individual files from combined file. To minimize load on NameNode and I/O performance improvement index prefetching is used. The footprint in NameNode is reduced then the file system becomes more efficient. EHDFS will minimize the time required for processing , Another work it improves the performance of data servers for hadoop distributed file systems by improving the performance for small files. It uses LSM structure based key- value store. The improvement in the throughput is up to 78% for small files. The logic is that the throughput gets increased when the large files and small files are grouped together in a separate cluster.

Ceph

It is a completely distributed file system. It provides excellent performance, reliability, and sclability. It provides a dynamic didtributed metadata management with a metadata cluster (MDS) and stores data and metadata in Object Storage Devices (OSD) which provides scalability. The API and client access is through a function called CRUSH [3]. It uses near POSIX system which allows concurrent read and write access with mechanism called capabilities. Load balancing is ensured in data and metadata level. It implements a counter for each metadata to know their frequency of access to move overloaded MDS to another one. A dynamic distributed metadata cluster provides extremely efficient metadata management and seamlessly adapts to a wide range of general purpose and scientific computing file system workloads. Performance measurements under a variety of workloads show that Ceph has excellent I/O performance and scalable metadata management, supportingmore than 250,000metadata operations per second.

V. PRAPOSED WORK

The aim of our work is to reduce the time taken to store and retrive the files from file system. Our proposed work uses different types of distributed file systems that are plugged into the

Hadoop framework. Different types of file system are suitable for different types of data and uses. So taking advantage of this heterogeneous nature of the file systems we select the DFS that is suitable for the incoming big data volume based on the file size and frequency of access. This is done based on the various DFS comparison [12] (show in table 1) and the suitability for the respective data.

We have made challenges in the file system layer of the Hadoop architecture with our customized data handler in the master node and respective file system in the clusters. HDFS and Lustre maintain an index where a physical location is associated to a file name. A meta data is maintained in the master node which is used by Ceph and GlusterFS for calculating file location

VII. CONCLUSION

EHDFS addresses three critical challenges of storage systems—scalability, performance, and reliability by occupying a unique point in the design space. With this approach we can maintain a buffer to store all the files at each datanode. When any client want to get the file which is avails bile in the buffer they can access directly by the address of that file. With this approach we can improve the efficiency and performance of HDFS

REFERENCES

[1] Abad,C.L., Lu, Y., & Campbell, R.H. (2011, September). DARE: "Adaptive data replication for efficient cluster scheduling," In Cluster Computing (CLUSTER), 2011 IEEE International Conference on (pp, 159 – 168)., IEEE.

[2] Cheng, Z., Luan, Z., Meng, Y., Xu, Y., Qian, D., Roy, A., … & Guan, G. (2012, September). Erms: An alastic replication management system for HDFS. In Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on (pp. 32- 40). IEEE.

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

Mrs. Swapna Vanguru has completed her B.Tech Computer Science and Engineering and M.Tech Computer Science and Engineering at Jawaharlal Nehru Technological University Hyderabad. The M.Tech was focused mainly on research of Data Preprocessing in Networks and in Cloud based on clustering techniques.In the same said area published National and International paper in other familier journals and proposed new algorithms.

Mrs. D.Aswani has completed her B.Tech Computer Science and Engineering and M.Tech Computer Science and Engineering at Jawaharlal Nehru Technological University Hyderabad. And she is interested to do research work in Bigdata Hadoop file syste,.