ISSN: 2321-7782 (Online)
e-ISJN: A4372-3114
Impact Factor: 7.327

Volume 5, Issue 11, November 2017

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Efficient Way to handle small files using Cloud DataProc in Hadoop

K. P. Jayakar[1]
Department & PVPIT Bavdhan
University of Pune
Pune – India

Y. B. Gurav[2]
Computer Department & PVPIT Bavdhan
University of Pune
Pune – India

Abstract: ***Hadoop is a framework for running applications on large clusters built of commodity hardware. It is a software framework for distributed processing of large datasets across large clusters of computers. The Hadoop framework transparently provides applications both reliability and data motion. Hadoop implements a computational paradigm named Map/Reduce. Hadoop's Distributed File System (HDFS) is designed to reliably store very large files across machines in a large cluster. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS has a master/slave architecture. An HDFS cluster consists of a single Name Node, a master server that manages the file system namespace and regulates access to files by clients. HDFS suffers a performance penalty with increased number of small files. Storing and managing a large number of small files imposes a heavy burden on the Name Node. Mainly Yahoo, Facebook, Netflix, and Amazon uses Hadoop to store huge data, Hadoop focus and framework works on unstructured data analysis and structured. HDFS is for storing huge, large files, but when large number of small files needs to be stored in HDFS, it faces some problem because all data stored and managed by single Name Node and it is burden on Name Node storage. Dataproc is a fully managed, cloud service with fast and easy access for running clusters Apache Spark and Apache Hadoop clusters in a simpler, more cost-efficient way. It is Flexible Virtual Machines, integrated and highly available.***

Keywords: DataProc, Apache Spark Apache Hadoop, Map/Reduce, HDFS.

I. INTRODUCTION

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

II. EXISTING SYSTEM RELATED WORK

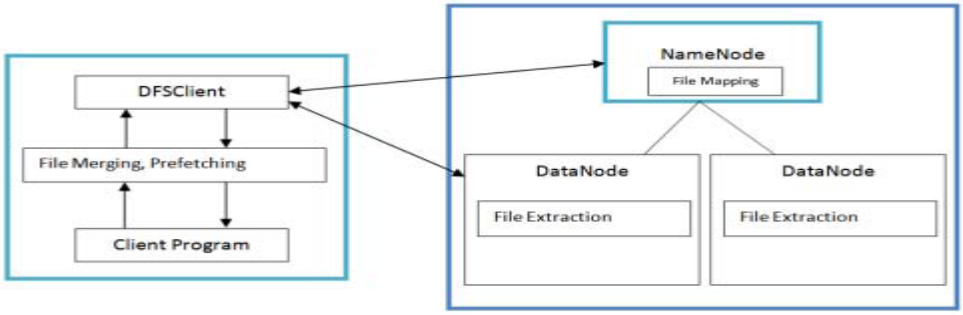

Fig.1 HDFS Architecture

A. *File Merging*

NameNode maintains two types of metadata for every file in HDFS namely, the file metadata and the block metadata. File metadata comprises of information about th file such as name of the file, location of the file in the name space tree, file size, modification time, access time, ownership details and file permissions. Block metadata comprises of information about the list of blocks that hold the file data and the location of these blocks. The file merging technique reduces both the file metadata and the block metadata maintained by the Name Node for small files.

B. File Mapping

File mapping is the process of mapping the small file name to the block of the combined file that contains this file. This is carried out by the NameNode. File mapping technique comes into play when the user wants to read a small file from the combined file. The user has to explicitly specify the name of the combined file and the small file while initiating the read operation. A request is sent to the NameNode, along with these two file names, for obtaining the location of the desired small file NameNode maintains a data structure called Constituent FileMap, for each combined file.

C. Prefetching

The file merging technique only reduces the metadata footprint in the NameNode. It does not improve the performance of read operations. As mentioned before, HDFS is designed on "write once, read many times" pattern. Improving the speed of file read operation is more significant than improving the speed of write operation.

D. File Extraction

File extraction involves the process of extracting the desired file contents from a block. This operation is carried out by the Data Node. While reading a file, the client specifies both the name of the small file and the name of the combined file.

III. PROPOSED SYSTEM

One of the best cloud based cloud based service offered on Google Cloud Platform which is cloud based managed spark and Hadoop is Google Cloud Dataproc. Dataproc is a fully managed, cloud service with fast and easy access for running clusters Apache Spark and Apache Hadoop clusters in a simpler, more cost-efficient way. It is Flexible Virtual Machine, availability of cluster is high, integrated.

- *Management of cluster* -Automatic cluster management, focus on data only, not on cluster. Clusters are stable, speedy, scalable
- *Economical affordable*- Google Cloud Platform is economical affordable. Cloud Dataproc has a reasonable cost and an easy to understand and manageable price structure, measured by the second, based on actual use. It contains instances which is lower cost preemptible, in lower cost system gives powerful clusters
- *Availability high*-Clusters run with many master nodes and after failure, it sets jobs to restart, it shows availability of your jobs and clusters are high.
- *Open Source* -Tools, libraries, documentation provided by spark and Hadoop system that can leverage with Dataproc. You can start without learn new API, tools, you can continue projects or ETL without redevelopment.
- *Fast*-Scalable and fast-creation and resizing of Dataproc cluster is easy and fast. You can create multiple node in short period of time, no need to worry about cluster data pipeline which outgrow cluster.it takes time less than 90 seconds.

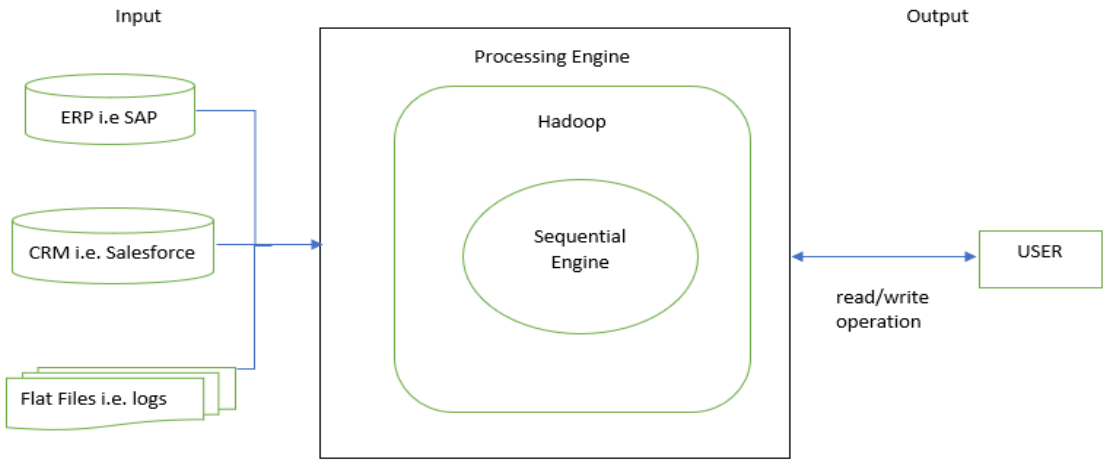

Fig.2 Proposed system architecture

In the Fig. architecture of proposed system shows flow of the system. All the files from various sources i.e. CRM, Flat files, ERP are given as input to the processing engine. By default, the block size in Hadoop is 64mb.for large file this block size is best to do execution, and for storage also. But for small file just like 20kb-50kb, current block is very large to store single small file. So, to avoid this problem, we use processing engine. Processing engine contain Hadoop and sequence engine is part of Hadoop

IV. Implementation Plan

Execution of processing engine is shown in fig. flow of processing engine, according to that, map reduce is performed on all files, after that merging of all files performed. Then sequential file is generated. MapReduce of the sequential file is performed, do actual mapping and reduction of sequential file. The sequential file is compressed, so compressed sequential file is generated and this operation performed by Name node only, and compressed sequential file store at data node

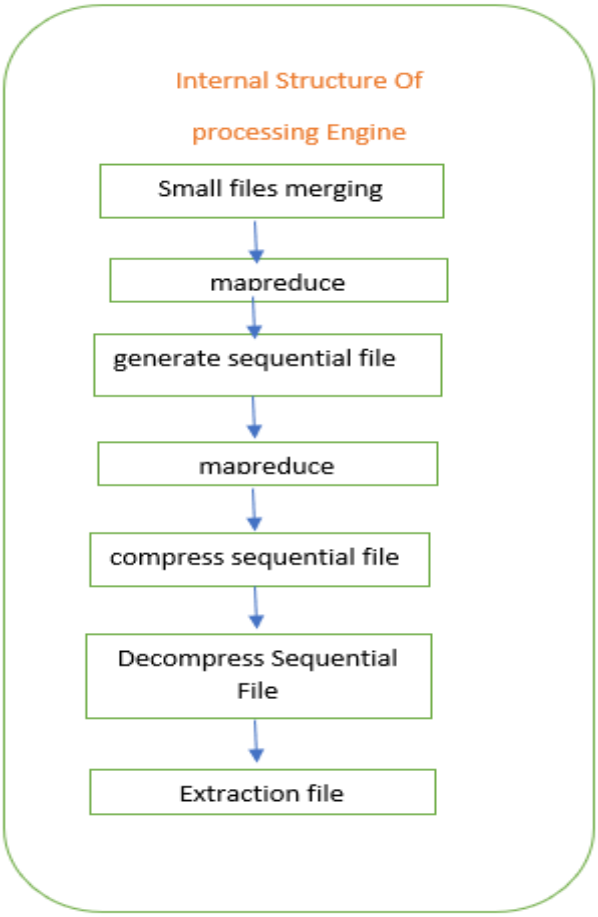

Fig.3 Flow of Processing Engine

When any request raises for specific file which is part of compressed sequential file to perform read or write operation in specific file, its necessary to decompress the file. So, file is decompressed, and all files are ready to access, we can easily access the any file by extraction of the file. In Extraction process extracting the desired file contents from sequential file.

V. RESULT

The problem of large numbers of small file in Hadoop is solve using cloud Dataproc, it is Google cloud service which include Map/reduce processing framework and Hadoop distributed Filesystem

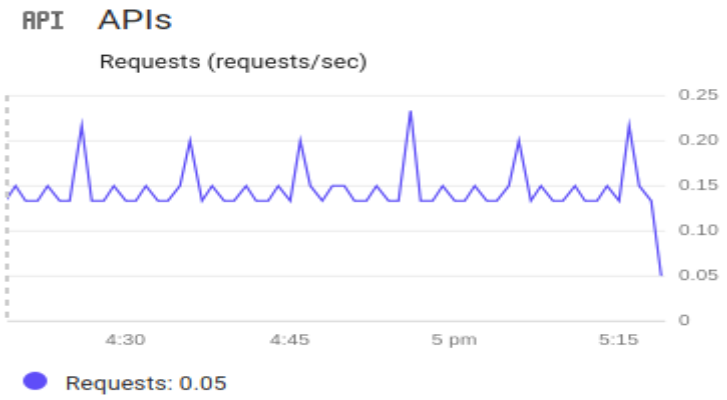

Fig.4 API Utilization Cycle

Fig.API Utilization Cycle shows number of API request executed per second. It is varied according to the input

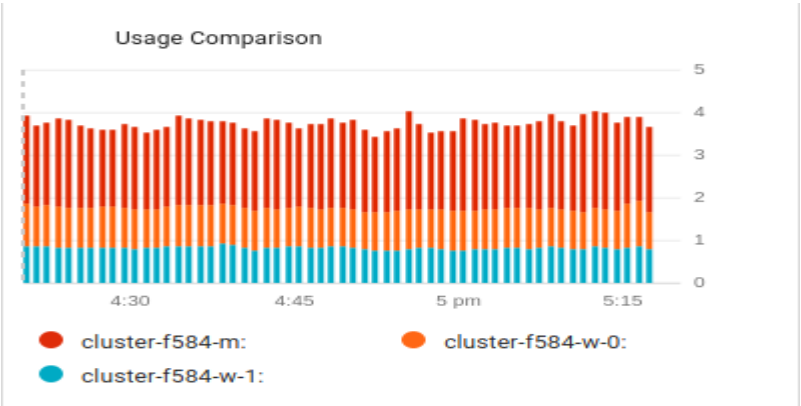

Fig.5 Usage Utilization

Fig. Usage Utilization Cycle shows the output i.e. number of cluster utilizing Name node memory for generating the output. Currently we have use three cluster, which simultaneously run the job. In above Fig. the current job is divided in three clusters. It also shows when job run and how much duration it will take.

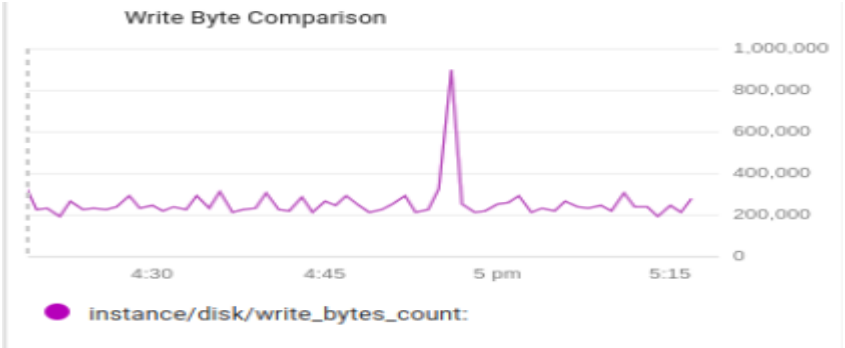

Fig.6 Write Byte Comparison

Fig.Write Byte Comparison shows time taken by hadoop to write sequential output file, it is the final output to the disk.

VI. CONCLUSION

The system provides a more advanced prefetching framework. This framework should support better file correlation analysis for prefetching and a better file merging process that takes this file correlation information into consideration. Append operation can be provided by Hadoop, to add files into an existing sequential file. Compression of sequential file is also providing, to minimize utilization of storage area. This will help to minimize the burden on Namenode memory.

References

1. Ahmed Eldawy, Mohamed F. Mokbel "A Demonstration of SpatialHadoop:An Efficient MapReduce Framework for Spatial Data" Proceedings of the VLDB Endowment, Vol. 6, No. 12 Copyright 2013 VLDB Endowment 21508097/13/10
2. Kenn Slagter • Ching-Hsien Hsu "An improved partitioning mechanism for optimizing massive data analysis using MapReduce" Published online: 11 April 2013
3. B. Dong, J. Qiu, Q. Zheng, X. Zhong, J. Li, Y. Li. "A Novel Approach to Improving the Efficiency of Storing and Accessing Small Files on Hadoop: A Case Study by PowerPoint Files". In Proceedings of IEEE International Conference on Services Computing, Miami, Florida, USA, July 2010, pp. 65
4. S. Ghemawat, H. Gobioff, S. Leung. "The Google File System". In Proceedings of ACM Symposium on Operating Systems Principles, Lake George, NY, October 2003, pp. 29 43.
5. A Novel Indexing Scheme for Efficient Handling of Small Files in Hadoop Distributed File System Chandrasekar S, Dakshinamurthy R, Seshakumar P G, Prabavathy B, Chitra Babu Department of Computer Science and Engineering SSN College of Engineering Kalavakkam, Tamilnadu, India
6. Hadoop: A Software Framework for Data Intensive Computing Applications Ravi Mukkamala Department of Computer Science Old Dominion University
7. J. Dean and S. Ghemawat, ``MapReduce: Simplied Data Processing on Large Clusters,'' OSDI 2004. (Google)
8. Jason Venner, Pro Hadoop, 1st ed. Apress, Jun. 2009, pp. 4 17
9. HDFS Architecture Guide
10. Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein "Online Aggregation and Continuous Query support in MapReduce" SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA. Copyright 2010 ACM 978-1-4503-00322/10/06.

