A Selective Approach for Storing Small Files in Respective Blocks of Hadoop

Chethan.R , ChandanKumar, Jayanth Kumar.S Girish.H.J , Prof. Mangala.C.N
UG Scholar, Associate Professor
Department of Computer Science and Engineering
chethanr95@gmail.com, kchandan488@gmail.com,sjayanth317@gmail.com,girishgowdahj@gmail.com
mangalacn.02@gmail.com

Abstract – Hadoop is an open source framework used for processing the data in big data. This hadoop majorly consists of two components 1. HDFS(Hadoop Distributed File System) 2. Map Reduce Component. HDFS is used for storing the files in hadoop and Map Reduce is used to process the data stored in HDFS. Hadoop doesn‘t performs well for storing the small files that is, it provides individual block of DataNode to individual file and hence reduces the performance. This research work gives an introduction of HDFS and the existing ways for solving the problem of small files. In this proposed approach, we merge the small files of into same block using Map Reduce programming model on Hadoop and hence provide different key value for files of different format. Hence this approach reduces the inefficient usage of memory from NameNode to access the DataNode and in turn it improves the efficiency of Hadoop by storing selective small files in respective blocks of Hadoop.We also propose a Traffic analyser with MapReduce paradigm that provides batch analysis in minimum response time and helps to process the log files in efficient and stable way

Keywords - Hadoop; NameNode; MapReduce; Small Files; Traffic Analyzer.

I. INTRODUCTION

It is a known fact that Hadoop has been specially created to manage Big Data. We know that Google is the world known popular search engine. To provide search results for users, Google had to store huge amount of data. Hence in 1990‘s,they started searching for the different ways of storing these huge amount of data and finally in the year 2003 they came up with Google File System(GFS) to store these huge amount of data and in 2004 they provided another technique called MapReduce for processing the data present in GFS.

But these techniques were presented to the world as a description and was just stored theoritically in GFS. So people had knowledge of the technique but there was no working model or code provided. Then in the year 2006 another major search engine, Yahoo came up with techniques called HDFS and MapReduce based on the descriptions given by Google. So, finally HDFS and MapReduce became the two core components of Hadoop.

Hadoop was actually created by Doug Cutting. Doug Cutting[2][7] choosed the logo of hadoop as an elephant. The reason behind it is that, the elephant is symbolic representation and a good solution for Big Data.

This paper covers many sections. Section II covers about the Hadoop distributed file structure, and the MapReduce component. Section III discusses the small file problems and the existing approach. Section IV discusses the proposed approach. Section V covers experimental setup. Section VI discusses conclusion and future work and then the paper is concerned with the acknowledgement for the constant support provided to us.

II. HADOOP COMPONENTS

The two main components of Hadoop are HDFS and MapReduce.HDFS(Hadoop Distributed File Structure) is used for storing the files in Hadoop and also consists of two nodes called NameNode and DataNode to split the given data into blocks and then store them in respective blocks. MapReduce component is used to process the data stored in the HDFS that is it involves in processing large amount structured and unstructured data in parallel in order to maintain good performance for the system. The three major components of this Hadoop architecture is as shown below

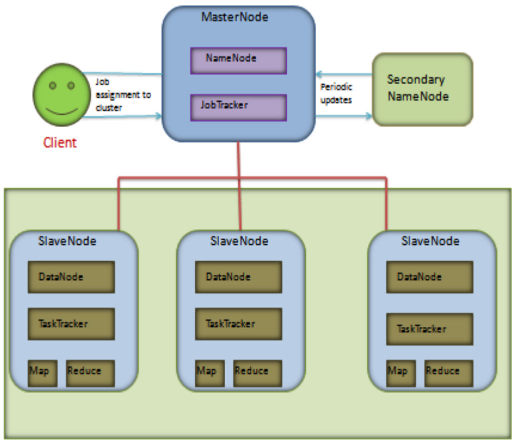


Figure 5: Hadoop Distributed File System Architecture

A.NameNode

NameNode is an center piece of an HDFS file system. It keeps the directory tree of all the system and tracks where across the cluster file data is kept and it does not store the data of these files itself.Client applications talk to NameNode whenever they wish to locate a file,or when they want to add/copy/move/delete a file.The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data lives.Hence these nodes are called Master Nodes.These also consists JobTracker which is a deamon runs on the Name Node.The secondary NameNode is connected to the NameNode which access acts a backup by storing replicas of the metadata of the file systems in local storage.

B.DataNode

A DataNode are refered as slave nodes since they follow the commands given by the NameNodes. They store data in the

Hadoop File System.A functional filesystem has more than one
DataNode, with data replicated across them. On startup,a DataNode connects to the NameNode;spinning until that service comes up and then responds to requests from the NameNode for filesystem operations. Client applications can directly talk to the DataNode,once the NameNode has provided the location of the data. TaskTracker is a deamon process on the DataNode which indeed should be deployed on the same servers that host DataNode instances.

C.HDFS Client

HDFS Clients are neither master nor slave,

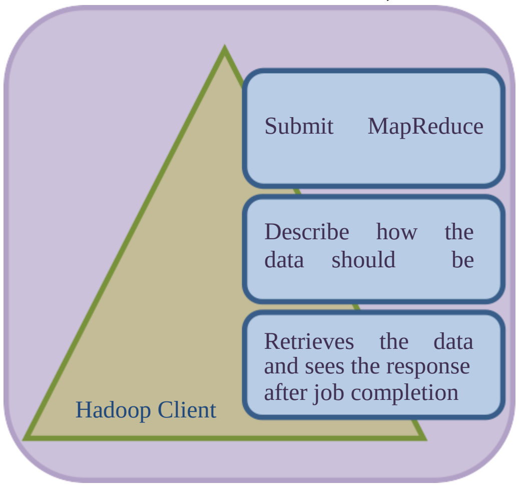

Submit MapReduce
Describe how the data should be
Retrieves the data and sees the response after job completion
Hadoop Client

Figure 2:HDFS Client structure

rather play role of loading the data into cluster, submit MapReduce jobs describing how the data should be processed and then retrieve the data to see the response after the completion of the given job.

III. SMALL FILE PROBLEMS IN HADOOP AND EXISTING APPROACHES

The Hadoop Distributed File system is a distributed file system mainly designed for batch processing of large volume of data. The default block size of HDFS is 64MB. Storing lot of small files which are extreamly smaller then the block size cannot be efficiently handled by HDFS. When data is represented in files significantly smaller than the block size the performance degrades dramatically[5]. Mainly there are two reasons for producing small files.One reason is some files are pieces of a larger logical file.Other reason is some small files cannot be combined together into one larger file and are essentially small.When small files are used there will be lots of seeks and lots of hopping from DataNode to DataNode to retrieve each small file which is an inefficient data access pattern[6]. Hadoop offers few options to handle these small files problems.They are as follows:

1) CONSOLIDATOR

Consolidator takes a set of files containing records belonging to the same logical file and merges the files together into larger files. It is possible to merge all small files into one large file,but it is not practical as then it would be a terabytes sized file.It would take a longer time to run such a huge file.

2) HAR FILES[6][9]

Hadoop Archieves were introduced to HDFS to alleviate the problem of lots of files putting pressure on the NameNodes memory. HAR files work by building a layered file system on top of HDFS.

A HAR file is created using the hadoop archieve command,which runs a MapReduce job to pack the files being archieved into a small number of HDFS files. The below figure 3 shows the architecture of HAR which is containing two index called *Index* and *MasterIndex*..Reading through files in HAR is comparatively slower than reading the files in HDFS because HAR files requires two _index‘ file reads as well as the data file read.

This is one of the disadvantage of HAR file.
In order to overcome the disadvantage of this HAR the locality of speed in HAR should be improved in order to increase the speed of access.

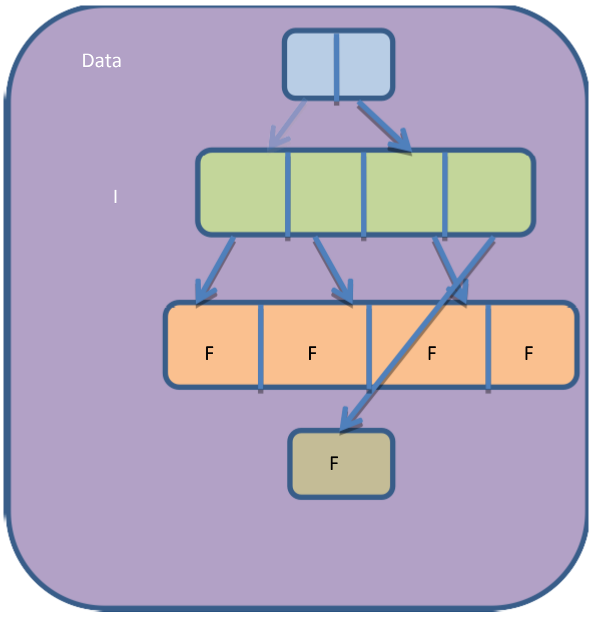


Figure 3: HAR File Layout

3) USING HBASE STORAGE

HBase stores data in MapFiles(indexed Sequence Files) and therefore it is a good choice when it is need to do MapReduce streaming analysiswith the occasional random look up. But the major problem is it doesn‘t allows partial keys completely and allows only one default sort per table.

4) SEQUENCE FILES[6][9]

Sequence files in a Hadoop specific archieve file format to tar and zip. The below figure shows Sequence file layout and the concept behind this is to merge the file set with using a key and a value pair and this created files are known as _Hadoop Sequence Files‘. In this method file name is used as key and file content is used as value but it is very much time consuming to convert existing data into sequence files.

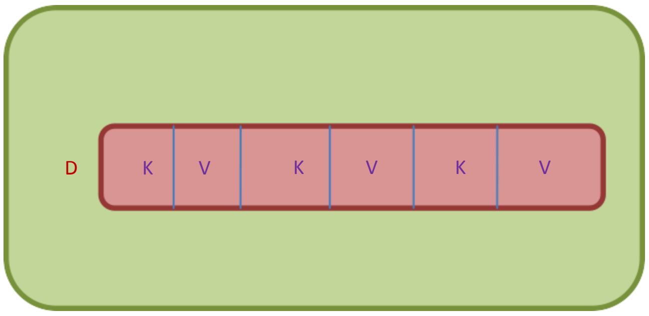


Figure 4: SequenceFile File Layout

IV. PROPOSED APPROACH

In this section, details of the proposed approach are introduced. Initially the idea of the approach is defined and the the algorithm and the mathematical model is described.

Storing large number of small files into HDFS is an overhead in terms of memory usage of NameNode and increase in executing time of MapReduce. According to this problem analysis, the proposed approach was merging the selective small files into respective blocks of Hadoop and hence make them a large file. Hence this will reduce the number of files and saves the memory of HDFS. Before merging all the small files together we determine the files of same format stored in the blocks of Hadoop, then these small files can be combined in parallel using MapReduce paradigm where Mapper will fetch the file and during these mapping technique the Mapper should provide a key and value for these small files inorder to fetch them. In this approach Mapper is made to provide keys for the file which is the byte offset and the value for these files will be the filename. Now the Mapper starts adding the files until it reaches the default block size and then pass it to reducer. The reducer will merge the files. This process is then carried out parallel until all the files are completely merged. This approach will reduce the time required for merging and executing the files and also makes it easier to access a file of particular format in the whole blocks of Hadoop. This approach also reduces the time of execution by ignoring to merge those files whose size is more than the threshold which is set to 80%(0.8) of the block size of Hadoop. This threshold can also be given as an input to the algorithm and it should be a integer number between range 0 to 1.

Algorithm for Map in MapReduce

i. Identify and fetch the small files and place them in a block of Hadoop knowing their filename and filesize.

ii. Consider the respective filename as key and filecontent as value for all the individual files in the block.

iii. If the file size is greater than the threshold ignore to add in the list.

iv. Maintain a list of block name(key given to block) and file names for merging which is to be done by reducer.

v. Pass this entire list to Reducer.

vi. After this the list becomes empty and fetches the new input file.

Algorithm for Reducer in MapReduce

i. Take the input from Mapper.

ii. Merge the files considering the threshold.

Mathematical Model

Let K_r be the set of Keys in $U^{\grave{}r}$, let V_r be the multiset of values $U^{\grave{}r}$, and let $(V_{k,r})$ denote the multiset of values in $U^{\grave{}r}$ that have key K. K_r and V_r $\Theta(n^{1-\varepsilon})$ can be partitioned across machines such that all machines get $O(n^{1-\varepsilon})$ bits, and the pair $(k, V_{k,r})$ gets sent to the same machine.

Proof : For a set of binary strings *B* denoted by *s(B)=bЄB* |*b*| the total space used by the strings in *B*. Since the algorithm is in MRC, by definition,

$s(V_r) + K_r) \leq U^{\grave{}r}) = \quad O(n^{2-2\varepsilon})$

Furthermore , the space of the reducer is restricted to $O(n^{1-\varepsilon})$; therefore ¥ K, $V_{k,r}$) is $O(n^{1-\varepsilon})$.

We can conclude that maximum number of bits mapped to any one machine is no more than the average load per machine plus the maximum size of $(k, V_{k,r})$ pair.

Thus,

$$\leq \frac{s(V_r)+s(K_r)}{\text{number of machines}} + \max(K\in K_r)$$

$$(|K|+s(V_{k,r}))$$

$$\leq \frac{O(n^{2-2\varepsilon})}{\Theta(n^{1-\varepsilon})} + O(n^{1-\varepsilon})$$

$$\leq O(n^{1-\varepsilon})$$

V. EXPERIMENTAL ANALYSIS

In this paper, we use the experiment of WordCount to test the performance HDFS using MapReduce in processing and storing the small files. This experiment can be performed for data of any format as of now we are providing the setup only for text file and also we compare the processing time required by proposed approach and the previous existing approaches for the same experiment.

WordCount

WordCount is a problem which is used to determine the count of repeatation of the words in the given file or data. The experiment was conducted by creating a text document and we appended it with some texts like –*hello how are you hope you are good*‖ and with some more text but the above text shows the data that was split and stored into a block of HDFS.

MapReduce consists of 3 components Mapper, Shuffel and Combine, Reducer. Now this data is sent to Mapper by NameNode and and Mapper takes two input *Key* and *Value* and returns two outputs that is the same *Key* and *Value*.Now Mapper divides each word and text into a single split and each splits are executed parallely. The Key for Mapper is the byte offset and the value is the content.

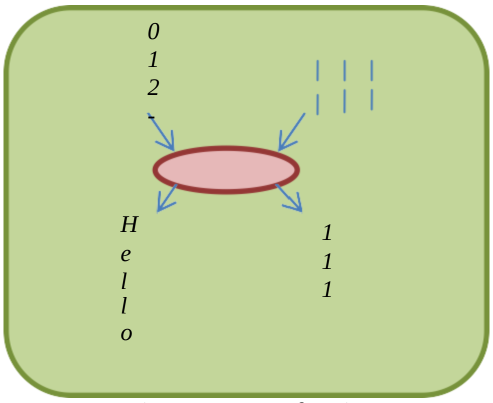
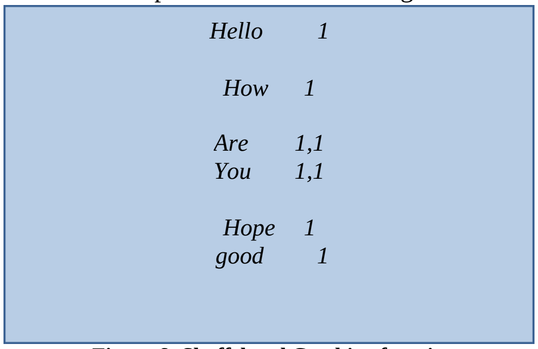


Figure 5: Mapper function

The value 1 which is given as an output indicates the index value for the corresponding splits.Now this output is given to the Shuffle and Combiner of MapReduce which combines the index value of the repeated words into a single block



Hello	1
How	1
Are	1,1
You	1,1
Hope	1
good	1

Figure 6: Shuffel and Combine function

After the completion of the shuffle and combine task the obtained output is sent to the last component of MapReduce that is Reducer which counts the number of index values and replace it by corresponding value and hence this provides the total number of count of a particular word in a given text file for the *WordCount* problem. The below figure shows the function performed by the reducer after taking the input fromthe Shuffel and Combine component.

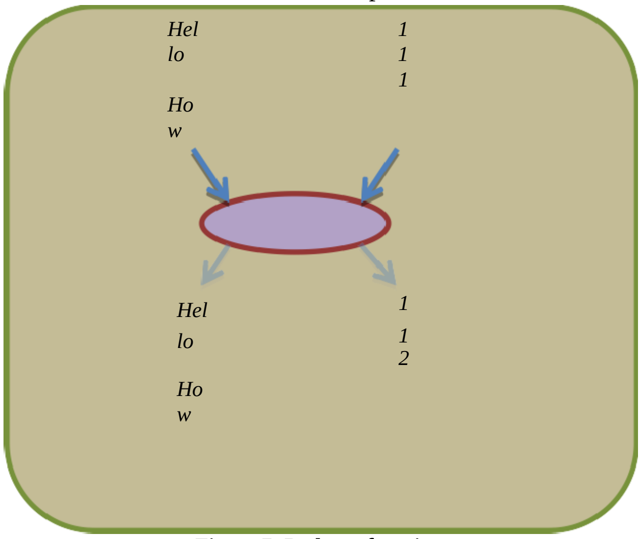


Figure 7: Reducer function

Once after performining the experiment with the proposed approach and the existing approaches we compared the execution time of all these approaches using a graph and then we came to the conclusion that the proposed system can perform more well in terms of execution time and the memory management.

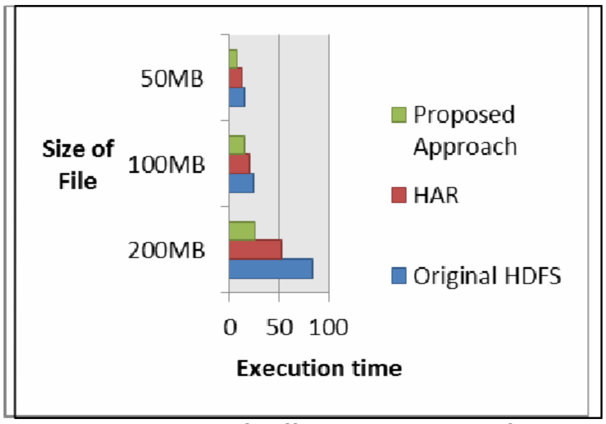


Figure 8: Comparision of different approaches of execution time on different sized files

The above graph in figure 8 depicts that the proposed approach takes less amount of time to execute the small file. In HAR approach there is inconsistency with the block size. Sequence file takes lot much of time to text data to sequence file format therefore we have not considered this approach and Original HDFS also takes lot of time to execute the application for small files and the proposed approach takes less amount of time for its execution since all the splits given to the Mapper by the NameNode are executed parallel and hencr reduces the number of Mappers which in turn increases the performance of the Hadoop by decreasing the execution time required for processing the small files.

VI. CONCLUSION AND FUTUREWORK

Hadoop is being one of the wide area of research in handling of small files in HDFS, hence the following research focuses on MapReduce approach to handle the small files and retrieve them using the key values given to the merged files. The proposed approach also focuses on execution time to run small files on Hadoop Cluster and hence the performance of HDFS. This can handle both sequence file and files of text, pdf etc which is related only to text file efficiently and also avoid files whose size is greater than threshold.

In future, work can be carried out to other files like audio,vedio and image files which are also a kind of small file. These files can be stored in HDFS and they also suffer performance issues which were faced with the small files discussed previously.

ACKNOWLEDGMENT

We would like to thank Mrs.Mangala.C.N, Associate Professor, Department of Computer Science and Engineering, East West institute of technology for her constant support and her guidance and also for providing us the opportunity to contribute on the small file problems faced by Hadoop. At last we also thank all faculty members of Department of Computer Science and Engineering for helping us in this work.

REFERENCES

[1] Manghui Tu, Peng Li, I-Ling Yen, BhavaniThuraisingham, Latifur Khan.Secure Data Objects Replication in Data Grid., IEEE Transactions on Dependable and Secure computing, Vol. 7, No. 1,2010

[2] Apache Hadoop, http: //hadoop.apache.org/

[3] P. H. Cams, W. B. Ligon III, R. B. Ross, and R. Thakur, PVFS: A parallel file system for Linux clusters, in Proc. of 4th Annual Linux Showcaseand Conference, 2000, pp. 317327.

[4] Hong Sha, Yang Shenyuan, Key technology of cloud computing and research on cloud computing model based on Hadoop, Software Guide,2010,9(9): 9 I I.

[5] Xie Guilan, Luo shengxian, Research on applications based on Hadoop MapReduce model, Microcomputer & its applications, 2010, (8): 4 7.

[6] The-small-files-problem, Cloudera http://blog.cloudera.comlblog/ 2009/02/the-small-files-probleml, February 2009

[7] Dhruba Borthakur, The Hadoop Distributed File System: Architecture and Design, ACM 2006

[8] Amazon Web Services, http://aws.amazon.comlec 2/, 2006

[9] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler "The Hadoop Distributed File System" IEEE 2010

[10] Techniques-to-deal-with-small-files-in.html, http://hadoopshares.blogspot.in /2012/09/techniques-to-deal-with-smallfiles- in.html, September 2012

AUTHORS BIOGRAPHY

Mrs.Mangala.C.N received the B.E degree in Computer science and Engineering from NCET, Bangalore, VTU University in 2006 and got M.Tech degree in Computer Science from RVCE, Bangalore, India. She is currently working as Associate Professor in the faculty of CSE,EWIT-Bangalore, India. Her area of interest includes Image Processing, Data Mining and Big Data.

Mr.Chandan Kumar is persuing his B.E in Computer Science and Engineering in East West institute of Technology, Bangalore, India. His area of interest includes Big Data, Data Mining and Network Security.

Mr.Jayanth Kumar.S is persuing his B.E in Computer Science and Engineering in East West institute of Technology, Bangalore, India. His area of interest includes Big Data, Data Mining and Cloud Computing.

Mr.Girish.H.J is persuing his B.E in Computer Science and Engineering in East West institute of Technology, Bangalore, India. His area of interest includes Big Data, Cloud Computing and Network Security.

Mr.Chandan Kumar is persuing his B.E in Computer Science and Engineering in East West institute of Technology, Bangalore, India. His area of interest includes Big Data, Data Mining and Cloud Computing.