Live Website Traffic Analysis Integrated with Improved Performance for Small Files using Hadoop

Navyashree B L[#1], Rakshitha H[#2], Ranjitha M[#3], T Auntin Jose [#4]
1,2,3 Bachelor of Engineering Students, 4 Associate Professor
Department Of Computer Science And Engineering,
RAJARAJESWARI COLLEGE OF ENGINEERING
BENGALURU-560074, KARNATAKA, INDIA
navyashree333@gmail.com, rakshitha.h@gmail.com, ranjitha200008@gmail.com, auntin123@yahoo.com

Abstract— Hadoop, an open source java framework deals with big data. It has HDFS (Hadoop distributed file system) and MapReduce. HDFS is designed to handle large amount files through clusters and suffers performance penalty while dealing with large number of small files. These large numbers of small files pose a heavy burden on the NameNode of HDFS and an increase execution time for MapReduce. Secondly, as an application part Traffic analyzer implemented with the combination of Hadoop and Map-Reduce paradigm, which makes it possible to analyse the any website programmatically. A web ranking metric, web analytics or simply web measurement refers to a system used to measure factors that affect a website‘s exposure and traffic on the web. The proposed approach is done to handle small files. In proposed approach, ―merging‖ of small file is done using MapReduce programming model on Hadoop. This approach improves the performance of Hadoop in handling of small files and also reduces the memory required by NameNode to store them. Traffic analysis gives the rank, number of views, visitors, index number so on any website which indicates the true analysis of the website in frequent basis using Hadoop.
Keywords— *MapReduce; Hadoop; HDFS; Small Files; Traffic Analyzer;*

I. INTRODUCTION

A.Hadoop

As Internet is growing rapidly, data is exploding from and growing beyond limit of expectation, the traditional techniques like RDBMS miserably fail to handle large files. Apache Hadoop [1][2] was created out of necessity by Doug Cutting. Published in the papers by Google, Hadoop was inspired in storing, processing and analyzing hundreds of terabytes, and even petabytes of data [3]. No data is too big for Hadoop.

The architecture of Hadoop is as shown in the figure1.

i. HDFS

a. Name Node

Apache Hadoop includes HDFS (Hadoop Distributed File System) is distributed file system designed .This enables the data to be processed in parallel using all of the machines in the cluster. HDFS is a portable file system written in Java for the Hadoop framework.

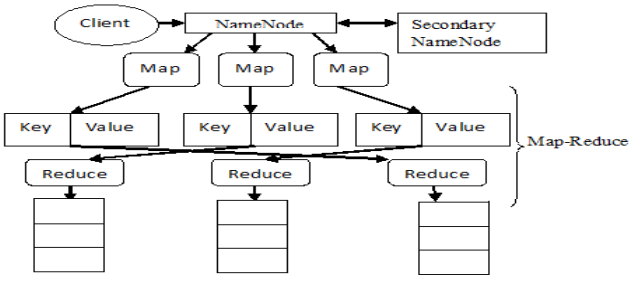

Figure1: Basic architecture of Hadoop

The design of the HDFS is such that it is used to store large files. The architecture of HDFS design stores large cluster of data in form of NameNode and DataNode which is taken from client. It consists of a metadata in the form of the NameNode and a large number of I/O nodes called DataNode.

b. Data Node

The DataNode works as the slave on which the actual data resides. To indicate its presence in the system, the DataNode keeps on sending status signal to the NameNode at regular intervals. The DataNode is responsible for serving the read and write request for the client. The daemon named as TaskTracker runs on the DataNode which is responsible for executing the individual tasks assigned by the JobTracker.

The DataNode service all read/write and file replication requests based on job given from the NameNode. Because Hadoop keeps all file system metadata in main memory, it is necessary for the NameNode to be its own server, this way file access is not slowed because of strain on the NameNode from serving metadata requests.
In order to keep the replication high and to rebalance the data, the DataNodes interact with one another and moves and copies the data around.

c. HDFS Client

Client machines have Hadoop installed with all the settings, but are neither a Master nor a Slave. Client machine loads the data into the Hadoop cluster, along with this it also invokes the map reduce , which says the

processing of data and also view or retrieved the results when the process is done. Client can read, write and delete files and also perform the operations to create and delete directories with contacting to NameNode. Based on the Client request only the NameNode process the job.

ii . Map Reduce

MapReduce is another component of Hadoop. MapReduce is regarded as the heart of Hadoop. In MapReduce, there are two job that Hadoop performs i.e. map() function and reduce() function. The map() function is designed to take the set of data and breaks into number of different tuples in the form of key-value pair. Since there are multiple DataNodes where the huge data is stored, processing the required file among huge set of file is a time consuming task, as solution to this the key-value pair is generated where searching of data becomes easier.

The reduce() function on the other hand considers the output from the map as an input and then combines those data tuples into a smaller set of tuples. The map job is always performed prior to the reduce job.

B.Traffic Analyzer

A website tells a lot about a business. It shows how much thought the business puts into its brand and whether it values having a website. Unfortunately, far too many companies don‘t really value their websites and don‘t get the full benefit out of them. They neglect design, website copy, and other important essentials. They put the focus only on making sales.

This results in a really bad website and leaves visitors unsure if the company is the best one to do business with. This is why it‘s so important to have a value proposition. A website needs to tell visitors in a couple of sentences or less why their business is the best choice for the visitor, instead of sending a bunch of different messages that won‘t be received.

Unlike other marketing venues, visitors to a website are typically anonymous. Web traffic analyzer helps to understand the behavioral data using Alexa server. The analyzer draws the data from the server calculates the result and generate the files which contains the calculated result.

II. PROBLEM ANALYSIS

Hadoop is a better approach is mainly designed to handle the huge amount of data in form of terabytes anta petabytes. But it has a major performance issue for handling with large number of small files. The data is stored using HDFS in multiple nodes, the default block size of HDFS is 64MB designed to access large files. If there is a file whose size is less than 64MB it is stored on one entire block. Consider the case which is shown in the diagram, where Client request for analysis of few set of websites. Traffic analyser pull the registered website from the database, it communicates through the Alexa server and fetches the information from server and generates the report in the form of a file for given websites. These files might be of size 1KB. If there are several files generates of these size is put on to the Hadoop cluster. These files are processed and each file is stored different blocks, where the remaining memory is being unused. This results in inefficiency of memory management.

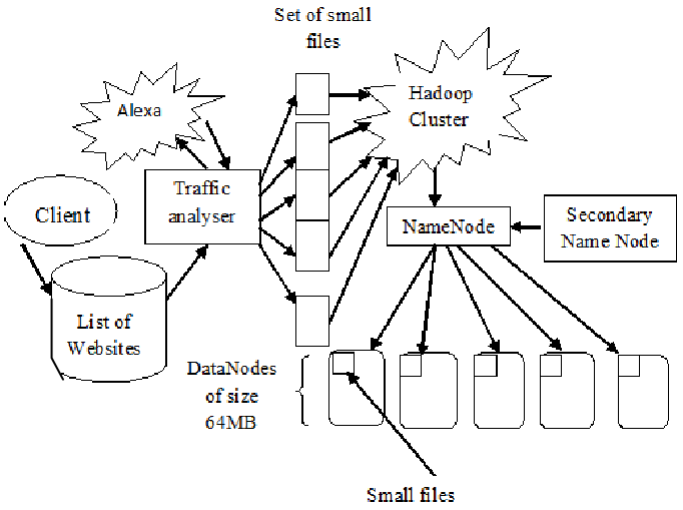

Figure 2: Problem in HDFS

Along with the memory being inefficient, small files pose a heavy burden on the NameNode. To read or access the data from a small file causes hopping from DataNode to DataNode and a lot many seeks are required in search of data from the small file. In case there are large number of small files, then each map task processes very little input and for every data there is a map tasks.

It imposes a high overhead in the system. In the NameNode, capacity of namespace of the system is limited by physical memory. This causes metadata to occupy larger portion of the memory.

So, if there are thousands of small files than the efficiency of HDFS fail to manage the files via performance, memory management becomes disaster.

III. PROPOSED APPROACH

According to the problem analysis, Hadoop suffers from performance issues from large number of small files. To overcome this problem, ―Merging‖ solution is taken. But merging algorithm is time consuming. In order to reduce the time consumption, small files can be combined parallel using the MapReduce algorithm. In map-Reduce algorithm threshold value can be set, whose value will be slightly lesser than block size of HDFS. In this algorithm, Map() function will fetch the file, generates the key-value pair, where the key is file size and value is filename . This key-value pair will be given to Reduce() function. The Map() would keep on adding the files until the default block size is reached and then pass it to Reduce(). The Reducer will merge the files. This process will then be carried out parallel till all the files are combined to the default block size. Since it is carried parallel, this reduces

the time for merging and executing the files. If the file is larger than threshold value it ignores those files.

Consider the case mentioned in the problem analysis, according to the solution mentioned the architecture diagram is shown in figure 2.

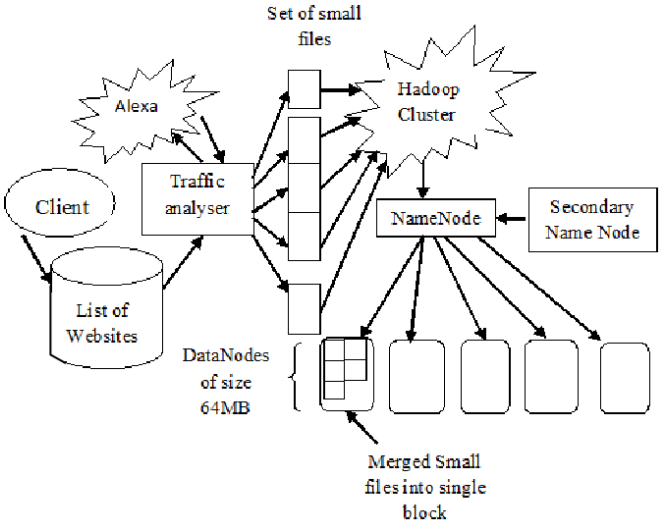

Figure 2: Problem in HDFS

In this approach, the files being sent to the NameNode generates the key-value pair. If the key i.e. the size is relatively lesser than the threshold being set in the algorithm, then the files are merged till it reaches the threshold value and kept in the DataNode. This clearly shows that the memory is efficiently used after processing. NameNode burden for execution also reduces. Traffic analyser is also proposed using Map-reduce algorithm this helps to have the minimum response time. The computing capacity of the website analysis also reduces.

Through this approach the execution time will reduce which is shown through graph in figure 3.

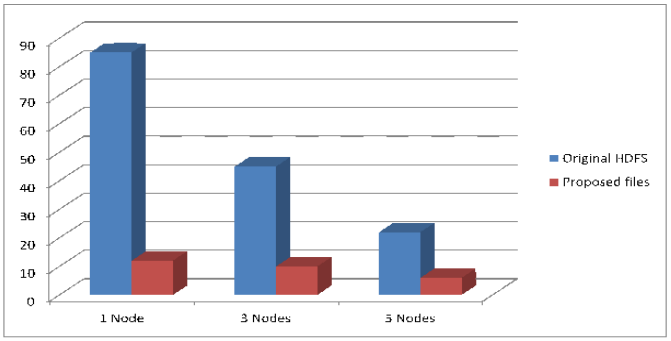

Figure 4: Comparison of Different Approaches of execution time on different number of nodes

Original HDFS takes a lot of time to execute the application for small files as it is the rule of thumb, Hadoop that each file is separately stored in individual block and each block is mapped to one map. So an increase in the number of map() function, increases the execution time. It is depicted in figure 4.

Consider the case to having 20000 small files. The size of these files range from 10KB to 126MB. The cumulative size of all the files is approximately 525MB.

According to proposed approach they are treated as individual file and store in its metadata into main memory. So it requires very less amount of memory than original HDFS. It is depicted in figure 5.[3]

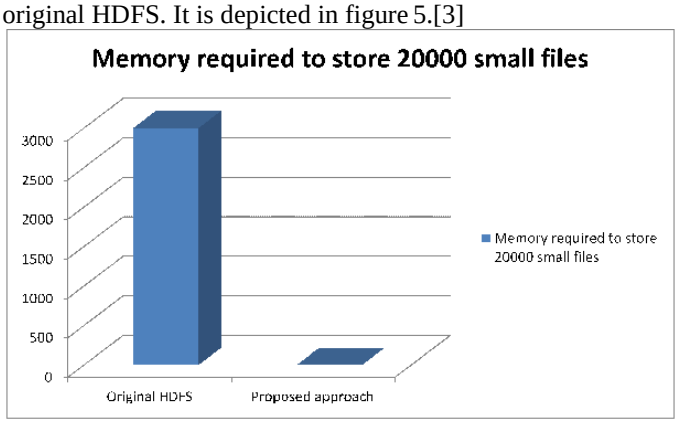

Figure 5: Memory required for storing small files

IV. CONCLUSION AND FUTURE WORK

As such Hadoop being wide area of research and one of the topics is chosen for research is handling of small files in HDFS, so the following research focuses on a MapReduce approach to handle small files, considering mainly two parameters. Firstly, execution time to run file on Hadoop Cluster and secondly the memory utilization by NameNode. By considering these parameters, proposed algorithm improves the result compared to existing approaches. Also the analysis of website traffic using Map-Reduce paradigm reduces the computing time and helps for the parallel processing. Thus, the overall system helps to increases the efficiency of the Hadoop for generated small files.

As for future work, small file storage solutions on HDFS will be mainly studied for other types of files as well. Based on file type‘s analysis, small files are classified as multiple types, and customized approaches will be supplied to different types to further improve the efficiency. The Solution might also work on the merging of different types of files and obtaining the efficiency of the algorithm. Image files also can be considered because it is also a small file whose performance is low in the HDFS.

REFERENCES

[1] http://hadoop.apache.org/core/.

[2] Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simplied Data Processing on Large Clusters, Google, I nC.2004

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

[4] Parth Gohil, Bakul Panchal, J S. Dhobi –A Novel Approach to Improve the Performance of Hadoop in Handling of Small Files‖ 2015 IEEE

[5] Grant Mackey, Saba Sehrish, Jun Wang "Improving Metadata Management for Small Files in HDFS" 2009 IEEE

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

[7] JaiPrakesh Varma, Bamkim Patel, Atul Patel, –Big Data Analysis: Recommendation System with Hadoop Framework‖ , in 2015 IEEE International Conference on Computational Intelligence & Communication Technology

[8] Liu Jiang, Bing Li , Meina Song "THE OPTIMIZATION OF HDFS BASED ON SMALL‖ Proceedings of IC-BNMT 2010

[9] http://www.cloudera.com/blog/2009/02/02/the-small-files-problem.

[10] J. Venner, Pro Hadoop. Apress, June 22, 2009.

[11] Http://issues.apache.org/jira/browse/HADOOP-1687.

[12] Hadoop archives, http://hadoop.apache.org/common/docs/current/hadoop_archives.html.