International Journal of Advanced Research in IT and Engineering

ISSN: 2278-6244
Impact Factor: 6.111

Map Reduce based Analysis of Live Website Traffic Integrated with Improved Performance for Small Files using Hadoop

Roshini R, PG Student, Dept of Computer Science and Engg, AIET, Moodbidri.

Manjunath Raikar, Assistant Professor, Dept of Computer Science and Engg, AIET, Moodbidri.

Abstract: Hadoop is an open source java framework deals with big data which is used to store and process huge amount of data. It has mainly two main components namely HDFS (Hadoop distributed file system) which stores large amount of data in a reliable manner and another is Map Reduce which is a programming model which processes the data in parallel and distributed manner. Hadoop does not perform well for little records as countless small documents produces a burden on the Name Node of HDFS and an decrease in execution time for Map Reduce is experienced. Hadoop is intended to handle large sized documents and consequently shows performance degradation while managing vast number of small records. This exploration work gives a presentation about HDFS, little document issue and existing approaches to manage these issues along with proposed way to deal with handle little records. In proposed approach the merging of small files is done using improved version of map reduce algorithm which mergers the small files saving the block size of the Hadoop.

I INTRODUCTION

Hadoop is an open source framework which enables the distributed parallel processing of large amount of data, the data sets are queried by map reduce programming model and divides it into subparts and runs parallely over multiple nodes. The design of the HDFS is such that it is used to store large files but inefficiency lies in storing a large number of small files because of high memory usage and unacceptable access cost. A small file is a file whose size is less than the HDFS block size. For example, 10 million files, each using a block, would use about 3 gigabytes of memory and therefore, storing and managing huge number of small files is a big challenge to HDFS. In this paper, a novel approach is represented to improve small files on HDFS, to reduce the memory overhead on the centre piece of Hadoop Distributed file system which is a name node It keeps the directory tree of all files in the file system, and tracks the cluster where data is kept across. It does not store the data of these files itself and to improve the performance of accessing small files in Hadoop.

Log files are generating at a record rate. In a day thousands of Petabyte or Terabytes of log files are generated by a data centre. It is very challenging to store and analyze these large volumes of log files. The problem of log files analysis is complicated because of not only its volume but also its disparate structure. Hadoop Map Reduce tunes up the task faster and load data faster than DBMS. Hadoop Map Reduce is applicable and used in analysis of big data. As log files is one of the type of big data which growth rapidly in such a way that Hadoop is used to store the log files and map reduce is used to analyse the same.

Hadoop is an open source project created by Doug Cutting and administered by the Apache Software Foundation. Hadoop enables applications to work with thousands of nodes and Terabytes or Petabyte of data. Large volume of commodity computers are connected in parallel to work on huge of data where Hadoop is used to store and process the same. Hadoop breaks up log files into numbers of blocks and these blocks are evenly distributed over cluster of thousands of nodes. Reliability and fault tolerance features implemented on account of replication of these blocks over the plot files are generating at a record rate.

In a day, thousands of Petabyte or Terabytes of log files are generated by a data centre. It is very challenging to store and analyze these large volumes of log files. The problem of log files analysis is complicated because of not only its volume but also its disparate structure. In 2009, Erik Paulson and Andrew Pavlo compared the Hadoop Map Reduce and SQL DBMS suggested that Hadoop Map Reduce tunes up the task faster and load data faster than DBMS.

Hadoop Map Reduce is applicable and used in analysis of big data. The log files are stored using Hadoop and parallel implementation of Map Reduce program for analyzing them. Hadoop enables applications to work with thousands of nodes and Terabytes or Petabyte of data. As described by Tom White in Hadoop cluster, there are thousands of nodes or machine which store multiple blocks of log files.

Hadoop breaks up log files into numbers of blocks and these blocks are evenly distributed over cluster of thousands of nodes. Reliability and fault tolerance features implemented on account of replication of these blocks over the Map Reduce paradigms are designed to compute large volumes of data in a parallel fashion, in which the workload is divided across a large number of machines or nodes. Map Reduce works by breaking the processing into map and reduce phase .Each phase has key value pairs as

input and output, the types of which may be chosen by the programmer.

II ARCHITECTURAL DIAGRAM FOR HADOOP DISTRIBUTED FILE SYSTEM

The design of the HDFS is such that it is used to store large files but inefficiency lies in storing a large number of small files because of high memory usage and unacceptable access cost. A small file is a file whose size is less than the HDFS block size. For example, 10 million files, each using a block, would use about 3 gigabytes of memory and therefore, storing and managing huge number of small files is a big challenge to HDFS. In this project, a novel approach is represented to improve small files on HDFS, to reduce the memory overhead of Name Node and to improve the performance of accessing small files in Hadoop

The objective of this project is to give introduction about HDFS, small file problem and existing ways to deal with it these problems along with proposed approach to handle small files. In proposed approach, merging of small file is done using Map Reduce programming model on Hadoop. This approach improves the performance of Hadoop in handling of small files by ignoring the files whose size is larger than the block size of Hadoop and also reduces the memory required by Name Node to store them.

The system architecture consists of the components as shown in the FIG 4.1,

a) Client Layer.
b) Service Layer
c) Communication Layer.
d) Hadoop Framework.
e) Map Reduce Framework.

- **Client Layer**: The Client Layer is the front end, which is a collection of html pages. It contains Account Access View, URL Registration View, Result View, and Configuration View
  - **Account Access View:** The user can create, login using his username and password, delete and logout the account.
  - **URL Registration View:** The user can give the desired number of websites to be analysed as an input.
  - **Result View:** The graphical representation of the analysed results will be shown to the user.
  - **Configuration View:** The user can decide the frequency at which traffic has to be analysed whether on a weekly or a daily basis. The analysis report will be sent to the user on a daily or weekly basis.
- **Service Layer**: The Service Layer includes Account Access Service, Url Registration Service, URL Registration Service Provider, Configuration Service, Result Formatter and JAX-B Libraries.
  - **Account Access Service:** Account Access View invokes the Account Access Service which stores the email id and password given by the user during registration.
  - **URL Registration Service Provider:** URL Registration Service Provider processes the request of URL Registration View.
  - **Configuration Service Provider:** Configuration Service Provider processes the request of Configuration View.
  - **Result Formatter:** Result Formatter processes the request of Result View to display the result in graphical manner.
  - **JAX-B Libraries:** JAX-B Libraries allows the java developer to map java classes to xml representation.
- **Communication Layer:** The Communication Layer consists of Traffic Data Pull, Database Service Operation, and Map Reduce Invoker.
  - **Traffic Data Pull:** Traffic Data Pull is an application programming interface used to pull the traffic from the mentioned website.
  - **Database Service Operation:** Database Service Operation is a channel through which users and applications have access to the data stored in the database.
  - **Map Reduce Invoker:** Map Reduce Invoker is used to invoke the Map Reduce Framework.
- **Hadoop Framework:** Hadoop is a free, Java-based programming framework that supports the processing of large data sets . The two main components of Hadoop are HDFS and Map Reduce, where HDFS is used to store the data and Map Reduce is used to process the data. The processing of the data in a distributed manner.

There are three major categories of machine roles in a Hadoop deployment. They are Name Node, Data Node and Client machines.

- **Name Node**: NameNode works as the master node which stores the file system metadata i.e. keeps the track of which file and blocks are to be stored on which Data Node. All information is stored in RAM. The secondary Name Node is connected to the Name Node which keeps on taking snapshot the metadata of the file system in local storage, if main Name Node fails Secondary Name Node acts as a backup by avoiding the single point of failure. It consists of Job Tracker which is the daemon process running background it partitions the job and provided it to the Task Tracker which is a daemon process running in the Data Node.
- **Data Node:** The Data Node works as the slave on which the actual data resides. To indicate its presence in the system, the Data Node keeps on

sending the heartbeat signal to the Name Node at regular intervals. In order to keep the replication high and to rebalance the data, the Data Nodes interact with one another and moves and copies the data around. The Data Node is responsible for serving the read and write request for the client. The daemon named as Task Tracker runs on the Data Node which is responsible for executing the individual tasks assigned by the Job Tracker.

- **HDFS Client:** Hadoop is installed in all the client machines with all the cluster settings. But are neither a Master nor a Slave. Instead. The loading of the data into the cluster is done by the client machine. Map Reduce jobs process the data and retrieve the results of the job when it is processed. Client can read, write and delete files and also perform the operations to create and delete directories by contacting to Name Node. Map Reduce is regarded as the heart of Hadoop. In MapReduce, there are two separate and distinct tasks which the Hadoop program performs. The first is the map job and other is the reduce job. The map function takes an input a set of data and converts in another set of data, where the individual portions of data is broken in small set of tuples in the form of key-value pair. The reduceO 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.

- **Map Reduce Framework** The Map Reduce Framework consists of Map and Reduce phase.
  - **Mapper:** Mapper takes the set of key/value pairs as an input where the entire file is broken down into records, it is converted into key value pairs as an input and the output will be an key value pair with the purpose served and will act as an input to an reducer.
  - **Reducer:** Reducer takes the input as an key value pair from an Mapper and the logic of the reducer is to combine all the values associated with the same key.
  - **Partioner:** Partioner partiones the key according to the hash key, for every key a single space is given to the hash key calculated.
  - **Combiner:** It is a java class specified in Map Reduce driver to process the output.
  - **Driver:** The Driver class defines the Mapper, Reducer and it shows the path from where the files are loaded from the local file system to the Linux machine.
  - **Writable:** Writable Interface provides the necessary methods for the required for the Mapper and the Reducer classes.

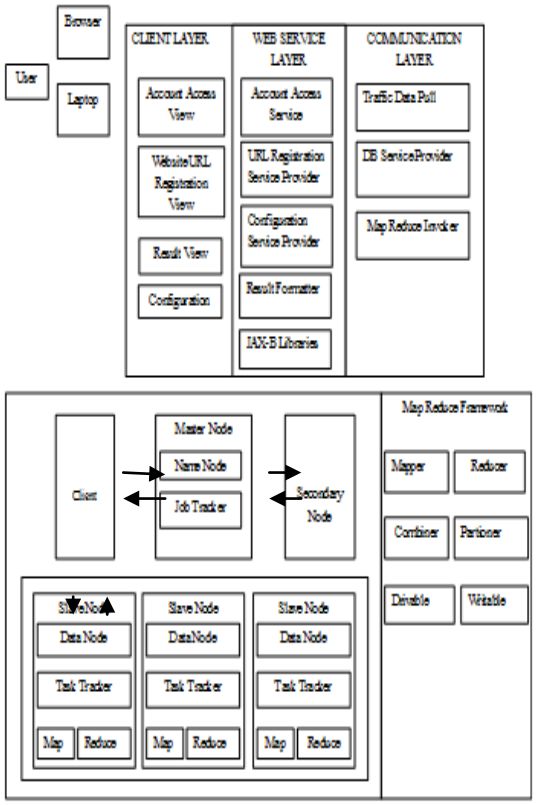

Fig 4.1 Architectural Diagram

III IMPROVED VERSION OF MAP REDUCE ALGORITHM

The overhead of HDFS is to store a large number of small which in turn affects the of memory usage of Name Node and increase in executing time of Map Reduce. According to this problem analysis, the proposed approached was merging small files into larger ones. This will reduce the number of files. This is similar to merging solution which is already being proposed. But this technique was time consuming, so in order to reduce the time consumption, small files can be combined in parallel using the Map Reduce paradigm where the Mapper will fetch the file, consider the key as file size and value as filename, pass it to Reducer. The Mapper would keep on adding the files until the default block size is reached and then pass it to Reducer. The Reducer will merge the files. This process will then be carried out parallel till all the files are combined. This will reduce the time in merging and executing the files. In this approach, it ignores those files to merge which are already larger than threshold and default threshold for this algorithm is set to (0.8) 80% of block size of Hadoop. It is also possible to give threshold as parameter to input to the algorithm as per the requirement. It must be any integer between ranges of 0 to l.

Algorithm of Map in Map Reduce

- Read small file name and file size.
- It considers the file name as key and file content as value.

- If file size is greater than threshold, ignore to add in list. (Default Threshold value = 80% of HDFS block size, default block size is 64 MB)
- It maintains list of filename for merging which is to be done by reducer. (Output limit).
- Pass the entire list of files to Reducer.
- After this the list becomes empty and fetches the new input file.

Algorithm of Reducer in Map Reduce

- Take the input from the Mapper.
- Merge the files considering threshold.

IV Results and Analysis

The results will be the analysis of n number of websites given by the user, the analysis part consists of the website rank, yahoo index, Google index, Alexa rank, no of visitors visiting the websites etc, and the main aim of the paper implementation is to solve the small file problem faced by the Hadoop that can be shown in the console, since Hadoop by default has an block size of 64 mb, if we upload n number of small files the Hadoop block size will be wasted so we have merged all the small files using improved version of map reduce algorithm and Hadoop block size will be consumed and even get the analysis of many websites.

V Conclusions

Log analysis helps to boost the business methods also on generate applied math reports. Hadoop MapReduce primarily based log file analysis tool can offer U.S. graphical reports showing hits for websites, user's page read activity, traffic, attack etc. From these reports business communities will get to their website rank and their improvement areas on behalf that are the potential customers from that IP or space or region web site is obtaining most hits, etc, which is able to be facilitate in planning future business and selling plans. Hadoop MapReduce framework provides parallel distributed computing and reliable information storage by replicating information for giant volumes of log files. Firstly, information get hold on block wise in rack on many nodes during a cluster so interval needed will be reduced that saves a lot of the interval and enhance performance. Here Hadoop is characteristic of moving computation to rather moving data to computation helps to boost time interval. Secondly, MapReduce with success works distributed for giant datasets giving the additional economical results, in and of itself Hadoop being wide space of analysis and one among the topics of analysis is handling of tiny files in HDFS, therefore the following analysis focuses on a MapReduce approach to handle tiny files, considering chiefly 2 parameters. Firstly, execution time to run file on Hadoop Cluster and second the memory utilization by NameNode. By considering these parameters, planned algorithmic program improves the result compared to existing approaches. It will handle each sequence file and document with efficiency and additionally avoid files whose size is larger than threshold. In future, work will be carried forward with image files. Image files also are quite tiny file. Image files will be hold on in HDFS and that they additionally suffer performance problems that were sweet-faced with tiny text files.

References

[1] L. Gatzoulis and I. Iakovidis, "Wearable and portable E-health systems," IEEE Eng. Med. Biol. Mag., vol. 26, no. 5, pp. 51–56, Sep.-Oct. 2007.

[2] I. Iakovidis, "Towards personal health record: current situation, obstacles and trends in implementation of electronic healthcare records in Europe," Int. J. Med. Inf., vol. 52, no. 1, pp. 105–115,

[3] E. Villalba, M. T. Arredondo, S. Guillen, and E. Hoyo-Barbolla, "A new solution for a heart failure monitoring system based on wearable and information technologies in," in Proc. Int. Workshop Wearable Implantable Body Sens. Netw., Apr. 2006, pp. 150–153.

[4] R. Lu and Z. Cao, "Efficient remote user authentication scheme using smart card", Comput. Netw, vol. 49, no. 4, pp. 535–540, 2005.

[5] M. D. N. Huda, N. Sonehara, and S. Yamada, "A privacy management architecture for patient-controlled personal health record system," J. Eng. Sci. Technol., vol. 4, no. 2, pp. 154–170,2009

[6] S. Schechter, T. Parnell, and A. Harte mink, "Anonymous authentication of membership in dynamic groups in," in Proc. 3rd Int. Conf. Financial Cryptography, 1999, pp. 184–195.

[7] D. Slamanig, C. Stingl, C. Menard, M. Heiligenbrunner, and J. Thierry, "Anonymity and application privacy in context of mobile computing in eHealth," in Mobile Response, New York, NY, USA: Springer, 2009 pp. 148–157.

[8] J. Zhou and Z. Cao, "TIS: A threshold incentive scheme for secure and reliable data forwarding in vehicular delay tolerant networks," in Proc. IEEE Global Commun. Conf., 2012, pp. 985–990.

[9] S. Yu, K. Ren, and W. Lou, "FDAC: Toward fine-grained distributed data access control in wireless sensor networks," in Proc. IEEE Conf. Comput. Commun. 2009, pp. 963–971.

[10] F. W. Dilemma and S. Lupetti, "Rendezvous-based access control for medical records in the pre-hospital environment," in Proc. 1st ACM SIGMOBILE Int. Workshop Syst. Netw. Support Healthcare Assisted Living, 2007, pp. 1–6.

[11] J. Sun, Y. Fang, and X. Zhu, "Privacy and emergency response in e-healthcare leveraging wireless body sensor networks," IEEE Wireless Commun., vol. 17, no. 1, pp. 66–73, Feb. 2010.