HMR LOG ANALYZER: ANALYZE WEB APPLICATION LOGS OVER HADOOP MAPREDUCE

Sayalee Narkhede[1] and Tripti Baraskar[2]

Department of Information Technology, MIT-Pune,University of Pune, Pune
sayleenarkhede@gmail.com
Department of Information Technology, MIT-Pune, University of Pune, Pune
baraskartn@gmail.com

ABSTRACT

In today's Internet world, log file analysis is becoming a necessary task for analyzing the customer's behavior in order to improve advertising and sales as well as for datasets like environment, medical, banking system it is important to analyze the log data to get required knowledge from it. Web mining is the process of discovering the knowledge from the web data. Log files are getting generated very fast at the rate of 1-10 Mb/s per machine, a single data center can generate tens of terabytes of log data in a day. These datasets are huge. In order to analyze such large datasets we need parallel processing system and reliable data storage mechanism. Virtual database system is an effective solution for integrating the data but it becomes inefficient for large datasets. The Hadoop framework provides reliable data storage by Hadoop Distributed File System and MapReduce programming model which is a parallel processing system for large datasets. Hadoop distributed file system breaks up input data and sends fractions of the original data to several machines in hadoop cluster to hold blocks of data. This mechanism helps to process log data in parallel using all the machines in the hadoop cluster and computes result efficiently. The dominant approach provided by hadoop to "Store first query later", loads the data to the Hadoop Distributed File System and then executes queries written in Pig Latin. This approach reduces the response time as well as the load on to the end system. This paper proposes a log analysis system using Hadoop MapReduce which will provide accurate results in minimum response time.

KEYWORDS

Hadoop, MapReduce, Log Files, Parallel Processing, Hadoop Distributed File System.

1. INTRODUCTION

As per the need of today's world, everything is going online. Each and every field is having their own way of putting their applications, business online on Internet. Seating at home we can do shopping, banking related work; we get weather information, and many more services. And in such a competitive environment, service providers are eager to know about are they providing best service in the market, whether people are purchasing their product, are they finding application interesting and friendly to use, or in the field of banking they need to know about how many customers are looking forward to our bank scheme. In similar way, they also need to know about problems that have been occurred, how to resolve them, how to make websites or web application interesting, which products people are not purchasing and in that case how to improve advertising strategies to attract customer, what will be the future marketing plans. To answer these entire questions, log files are helpful. Log files contain list of actions that have been occurred whenever someone accesses to your website or web application. These log files resides in web servers. Each individual request is listed on a separate line in a log file, called a log entry. It is automatically created every time someone makes a request to your web site. The point of a log file is to keep track of what is happening with the web server. Log files are also used to keep track of complex systems, so that when a problem does occur, it is easy to pinpoint and fix. But there are times when log files are too difficult to read or make sense of, and it is then that log file analysis is necessary. These log files have tons of useful information for service providers, analyzing these log files can give lots of insights that help understand website traffic patterns, user activity, there interest etc[10][11]. Thus, through the log file analysis we can get the information about all the above questions as log is the record of people interaction with websites and applications.

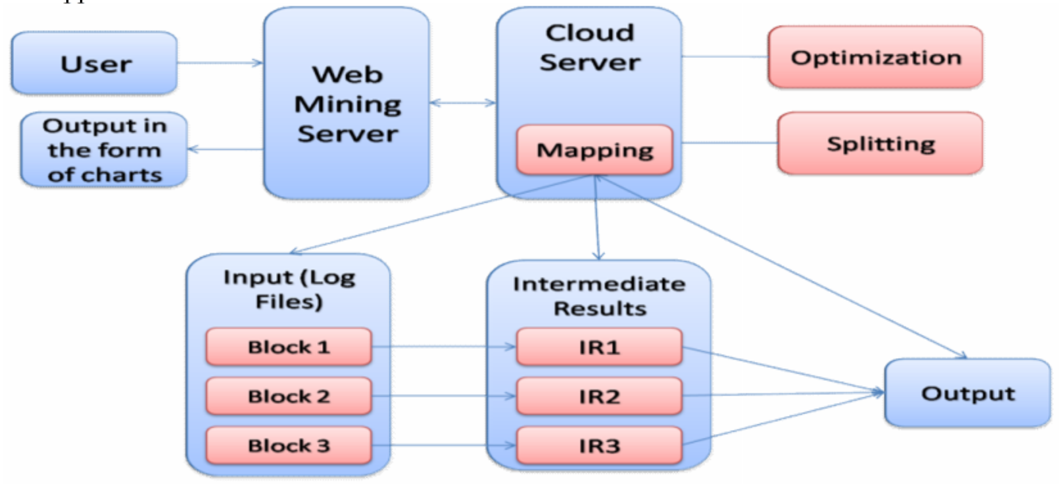

Figure 1. WorkFlow of The System

1.1. Background

Log files are generating at a record rate. Thousands of terabytes or petabytes of log files are generated by a data center in a day. It is very challenging to store and analyze these large volumes of log files. The problem of analyzing log files is complicated not only because of its volume but also because of the disparate structure of log files. Conventional database solutions are not suitable for analyzing such log files because they are not capable of handling such a large volume of logs efficiently. . Andrew Pavlo and Erik Paulson in 2009 [13] compared the SQL DBMS and Hadoop MapReduce and suggested that Hadoop MapReduce tunes up with the task faster and also load data faster than DBMS. Also traditional DBMS cannot handle large datasets. This is where big data technologies come to the rescue[8]. Hadoop-MapReduce[6][8][17] is applicable in many areas for Big Data analysis. As log files is one of the type of big data so Hadoop is the best suitable platform for storing log files and parallel implementation of MapReduce[3] program for analyzing them. Apache Hadoop is a new way for enterprises to store and analyze data. Hadoop is an open-source project created by Doug Cutting[17], administered by the Apache Software Foundation. It enables applications to work with thousands of nodes and petabytes of data. While it can be used on a single machine, its true power lies in its ability to scale to hundreds or thousands of computers, each with several processor cores. As described by Tom White [6] in Hadoop cluster, there are thousands of nodes which store multiple blocks of log files. Hadoop is specially designed to work on large volume of information by connecting commodity computers to work I parallel. Hadoop breaks up log files into blocks and these blocks are evenly distributed over thousands of nodes in a Hadoop cluster. Also it does the replication of these blocks over the multiple nodes so as to provide features like reliability and fault tolerance. Parallel computation of MapReduce improves performance for large log files by breaking job into number of tasks.

1.2. Special Issues

1.2.1. Data Distribution

Performing computation on large volumes of log files have been done before but what makes Hadoop different from them is its simplified programming model and its efficient, automatic distribution of data and work across the machines. Condor does not provide automatic distribution of data; separate SAN must be managed in addition to the compute cluster.

1.2.2. Isolation of Processes

Each individual record is processed by a task in isolation with one another, limiting the communication overhead between the processes by Hadoop. This makes the whole framework more reliable. In MapReduce, Mapper tasks process records in isolation. Individual node failure can be worked around by restarting tasks on other machine. Due to isolation other nodes continue to operate as nothing went wrong.

1.2.3. Type of Data

Log files are a plain text files consisting of semi-structured or unstructured records. Traditional RDBMS has a pre-defined schema; you need to fit all the log data in that schema only. For this Trans-Log algorithm is used which converts such unstructured logs to structured one by transforming simple text log file into oracle table[12]. But again traditional RDBMS has limitation on the size of data. Hadoop is compatible for any type of data; it works well for simple text files too.

1.2.4. Fault Tolerance

In a Hadoop cluster, problem of data loss is resolved. Blocks of input file are replicated by factor of three on multiple machines in the Hadoop cluster[4]. So even if any machine goes down, other machine where same block is residing will take care of further processing. Thus failure of some nodes in the cluster does not affect the performance of the system.

1.2.5. Data Locality and Network Bandwidth

Log files are spread across HDFS as blocks, so compute process running on a node operates on a subset of files. Which data operated upon by a node is chosen by the locality of a node i.e. reading from the local system, reducing the strain on network bandwidth and preventing unnecessary network transfer. This property of moving computation near to the data makes Hadoop different from other systems[4][6]. Data locality is achieved by this property of Hadoop which results into providing high performance.

2. SYSTEM ARCHITECTURE

Following system architecture shown in Figure 2. consists of major components like Web servers, Cloud Framework implementing Hadoop storage and MapReduce programming model and user interface.

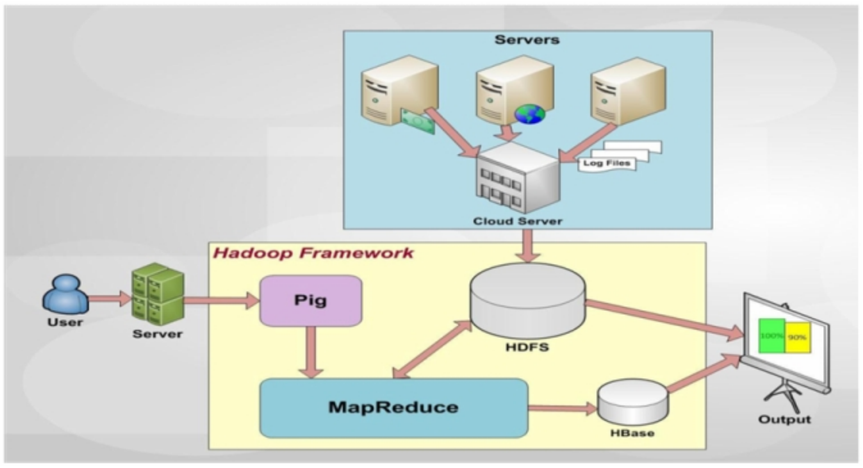

Figure 2. System Architecture

2.1. Web Servers

This Module consists of multiple web servers from which log files are collected. As log files reside in a web server we need to collect them from these servers and collected log files may require pre-processing to be done before storing log files to HDFS[4][6]. Pre-processing consists on cleaning log files, removing redundancy, etc. Because we need quality of data, pre-processing has to be performed. So these servers are responsible for fetching relevant log files which are required to be processed.

2.2. Cloud Framework

Cloud consists of storage module and processing MapReduce[3] model. Many virtual servers configured with Hadoop stores log files in distributed manner in HDFS[4][6]. Dividing log files into blocks of size 64MB or above we can store them on multiple virtual servers in a Hadoop cluster. Workers in the MapReduce are assigned with Map and Reduce tasks. Workers do parallel computation of Map tasks. So it does analysis of log files in just two phases Map and Reduce wherein the Map tasks it generate intermediate results (Key,value) pairs and Reduce task provides with the summarized value for a particular key. Pig[5] installed on Hadoop virtual servers map user query to MapReduce jobs because working out how to fit log processing into pattern of MapReduce is challenge[14]. Evaluated results of log analysis are stored back onto virtual servers of HDFS.

2.3. User Interface

This module communicate between the user and HMR system allowing user to interact with the system specifying processing query, evaluate results and also get visualize results of log analysis in different form of graphical reports.

3. IMPLEMENTATION OF HMR LOG PROCESSOR

HMR log processor is implemented in three phases. It includes log pre-processing, interacting with HDFS and implementation of MapReduce programming model.

3.1. Log Pre-processing

In any analytical tool, pre-processing is necessary, because Log file may contain noisy & ambiguous data which may affect result of analysis process. Log pre-processing is an important step to filter and organize only appropriate information before applying MapReduce algorithm. Pre-processing reduces size of log file also it increases quality of available data. The purpose of log pre-processing is to improve log quality and increase accuracy of results.

3.1.1. Individual Field In The Log

Log file contains many fields like IP address, URL, Date, Hit, Age, Country, State, City, etc.But as log file is a simple text file we need to separate out each field in each log entry. This could be done by using the separator like ‘space’ or ‘;’ or’#’. We have used here ‘#’ to separate fields in the log entry.

3.1.2. Removing Redundant Log Entries

It happens many times that a person visits same page many times. So even if a person had visited particular page 10 times processor should take this as 1 hit to that page. This we can check by using IP address and URL. If IP address and URL is same for 10 log entries then other 9 entries must be removed leaving only 1 entry of that log. This improves the accuracy of the results of the system.

3.1.3. Cleaning

Cleaning contains removing unnecessary log entries from the log file i.e. removing multimedia files, image, page styles with extensions like .jpg, .gif, .css from any log. These fields are unnecessary for application logs so we need to remove them so that we can get log file with quality logs. This will make log file size to be reduced somewhat.

3.2. Interacting With HDFS

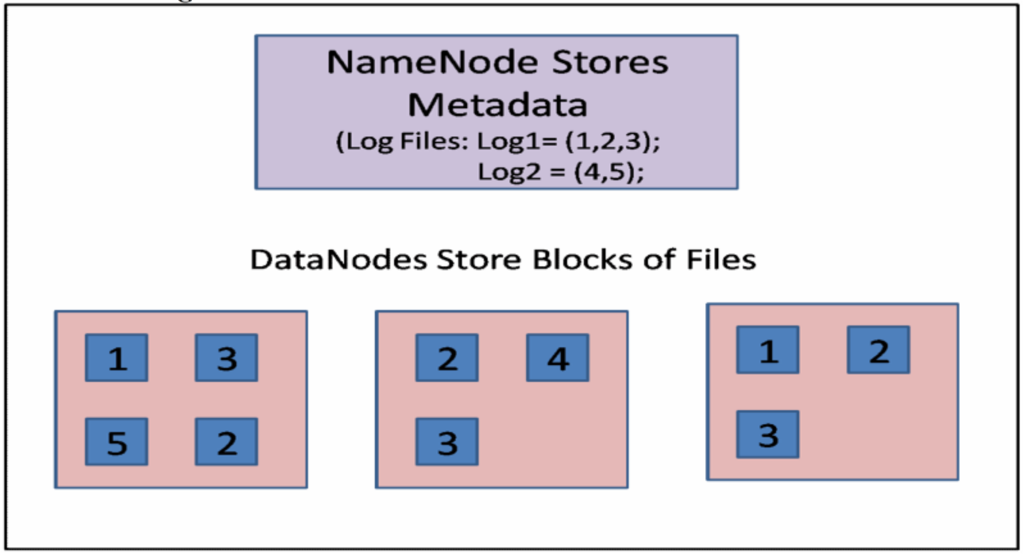

Figure 3. Hadoop Distributed File System

Hadoop Distributed File System holds a large log files in a redundant way across multiple machines to achieve high availability for parallel processing and durability during failures. It also provides high throughput access to log files. It is block-structured file system as it breaks up log files into small blocks of fixed size. Default size of block is 64 MB as given by Hadoop but we can also set block size of our choice. These blocks are replicated over multiple machines across a Hadoop cluster. Replication factor is 3 so Hadoop replicates each block 3 times so even if in the failure of any node there should be no data loss. Hadoop storage is shown in the Figure 3. In Hadoop, log file data is accessed in Write once, Read many times manner. HDFS is a powerful companion to Hadoop MapReduce. Hadoop MapReduce jobs automatically draws their input log files from HDFS by setting the fs.default.name configuration option to point to the NameNode.

3.3. MapReduce Framework

MapReduce is a simple programming model for parallel processing of large volume of data. This data could be anything but it is specifically designed to process lists of data. Fundamental concept of MapReduce is to transform lists of input data to lists of output data. It happens many times that input data is not in readable format; it could be the difficult task to understand large input datasets. In that case, we need a model that can mold input data lists into readable, understandable output lists. MapReduce does this conversion twice for the two major tasks: Map and Reduce just by dividing whole workload into number of tasks and distributing them over different machines in the Hadoop cluster.As we know, logs in the log files are also in the form of lists. Log file consists of thousands of records i.e. logs which are in the text format. Nowadays business servers are generating large volumes of log files of the size of terabytes in a day. According to business perspective, there is a need to process these log files so that we can have appropriate reports of how our business is going. For this application MapReduce implementation in Hadoop is one of the best solutions. MapReduce is divided into two phases: Map phase and Reduce phase.

3.3.1. Map Phase

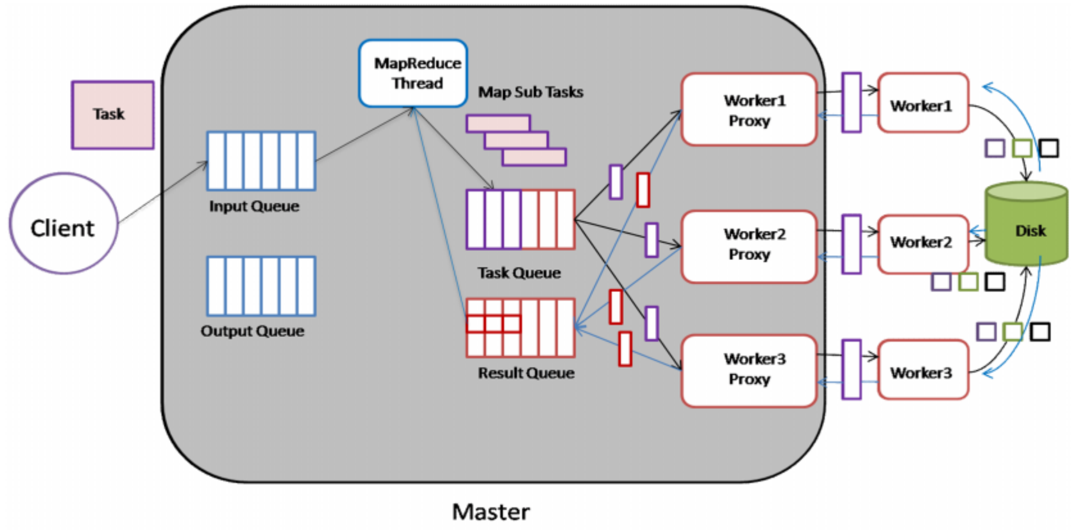

Figure 4. Architecture of Map Phase

Input to the MapReduce is log file, each record in log file is considered as an input to a Map task. Map function takes a key-value pair as an input thus producing intermediate result in terms of key-value pair. It takes each attribute in the record as a key and Maps each value in a record to its key generating intermediate output as key-value pair. Map reads each log from simple text file, breaks each log into the sequence of keys (x1, x2, …., xn) and emits value for each key which is always 1. If key appears n times among all records then there will be n key-value pairs (x, 1) among its output.
Map: (x1, v1) $\rightarrow$ [(x2, v2)]

3.3.2. Grouping

After all the Map tasks have completed successfully, the master controller merges the intermediate results file from each Map task that are destined for a particular Reduce task and feeds the merged file to that process as a sequence of key-value pairs. That is, for each key x, the input to the Reduce task that handles key x is a pair of the form (x, [v1, v2, . . . , vn]), where (x, v1), (x, v2), . . . , (x, vn) are all the key-value pairs with key x coming from all the Map tasks.

3.3.3. Reduce Phase

Reduce task takes key and its list of associated values as an input. It combines values for input key by reducing list of values as single value which is the count of occurrences of each key in the log file, thus generating output in the form of key-value pair (x, sum).
Reduce: (x2, [v2]) $\rightarrow$ (x3, v3)

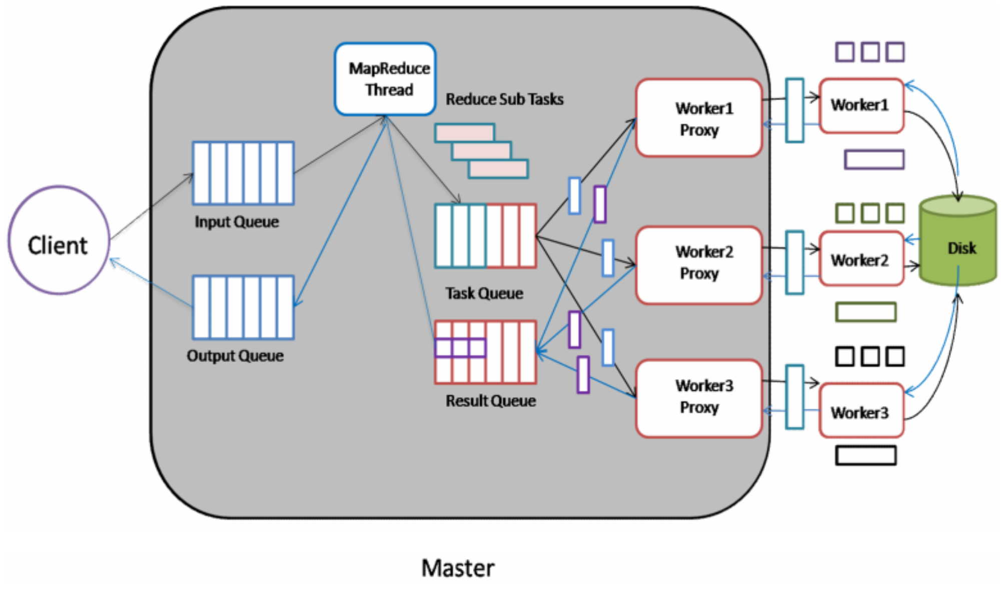

Figure 5. Architecture of Reduce Phase

4. IN DEPTH MAPREDUCE DATA FLOW

4.1. Input Data

Input to the MapReduce comes from HDFS where log files are stored on the processing cluster. By dividing log files into small blocks we can distribute them over nodes of Hadoop cluster. The format of input files to MapReduce is arbitrary but it is line-based for log files. As each line is considered as a one record as we can say one log. InputFormat is a class that defines splitting of input files and how to read these files. FileInputFormat is the abstract class; all other InputFormats that operate on files inherits functionality from this class. While starting MapReduce job, FileInputFormat reads log files from the directory and splits it into the chunks. By calling setInputFormat() method from JobConf object we can set any InputFormat provided by Hadoop. TextInputFormat treats each line in the input file as a one record and best suited for unformatted data. For the line-based log files, TextInputFormat is useful.While implementing MapReduce whole job is broken up into pieces to operate over blocks of input files. These pieces are Map tasks which operate on input blocks or whole log file. By default file breaks up into blocks of size 64MB but we can choose parameter to break file as per our need by setting mapred.min.split.size parameter. So for the large log files it is helpful to improve performance by parallelizing Map tasks. Due to the significance of allowance of scheduling tasks on the different nodes of the cluster, where log files actually reside, it is possible to process log files locally.

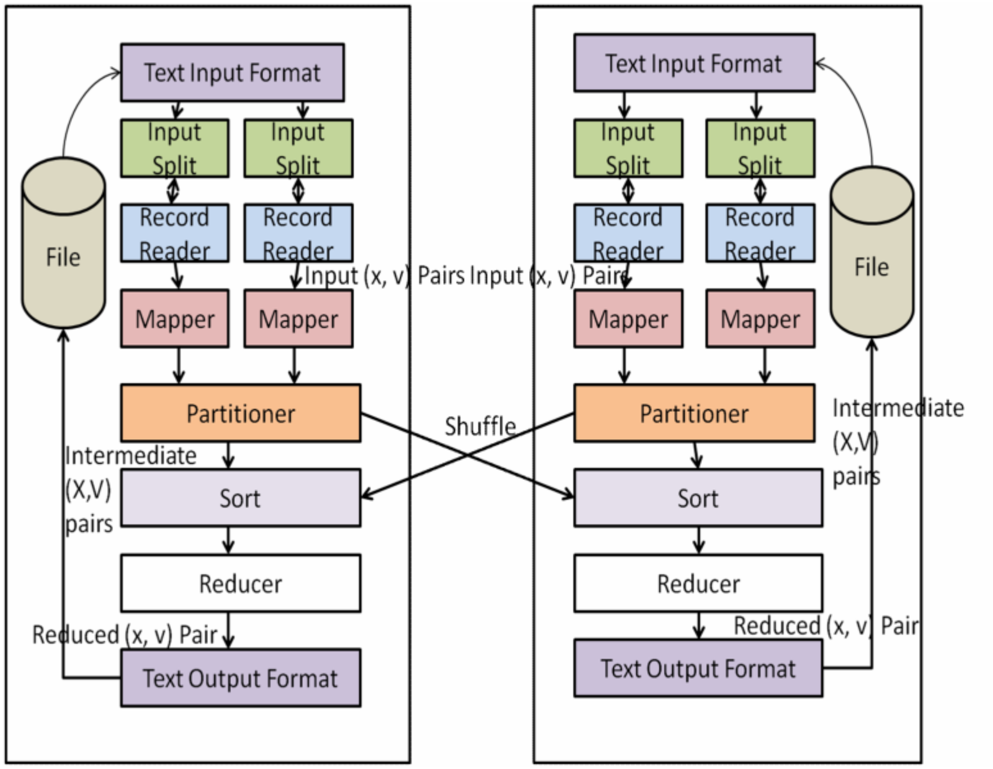

Figure 6. MapReduce WorkFlow

4.2. RecordReader

RecordReader is the class that defines how to access the logs in the log files and converts them into (Key, value) pairs readable by Mapper. TextInputFormat provides LineRecordReader which treats each line as a new value. RecordReader gets invoked until it does not deplete whole block. The Map() method of the mapper is called each time when RecordReader is invoked.

4.3. Mapper

Mapper does Map phase in the MapReduce job. Map() method emits intermediate output in the form of (key, value) pairs. For each Map task, a new instance of Mapper is instantiated in a separate java process. Map() method receives four parameters. Theses parameters are key, value, OutputCollector, Reporter. Collect() method from the OutputCollector object forwards intermediate (Key, value) pairs to Reducer as an input for Reduce phase. Map task provides information about its progress to Reporter object. Thus providing information about current task such as InputSplit, Map task, also it emits status back to the user.

4.4. Shuffle And Sort

After first Map tasks have completed, nodes starts exchanging intermediate output from Map tasks to destined Reduce tasks. This process of moving intermediate output from Map tasks to Reduce tasks as an input is called as shuffling. This is the only communication step in MapReduce. Before putting these (key, value) pairs as an input to Reducers; Hadoop groups all the values for the same key no matter where they come from. Such partitioned data is then assigned to Reduce tasks.

4.5. Reducer

Reducer instance calls Reduce() method for each key in the partition assigned to a Reducer. It receives a key and iterator over all the values associated with that key. Reduce() method also has parameters like key, iterator for all values, OutputCollector and Reporter which works in similar manner as for Map() method.

4.6. Output Data

MapReduce provides (key, value) pairs to the OutputCollector to write them in the output files. OutputFormat provides a way for how to write these pairs in output files. TextOutputFormat works Similar to TextInputFormat. TextInputFormat writes each (key, value) pair on a separate line in the output file. It writes a line in “Key Value” format. OutputFormat of Hadoop writes to the files mainly on HDFS.

5. EXPERIMENTAL SETUP

We present here sample results of our proposed work. For sample test we have taken log files of banking application server. These log files contain fields like IP address, URL, date, age, hit, city, state and country. We have installed Hadoop on two machines with three instances of Hadoop on each machine having Ubuntu 10.10 and above version. We have also installed Pig for query processing. Log files are distributed evenly on these nodes. We have created web application for user to interact with the system where he can distribute log files on the Hadoop cluster, run the MapReduce job on these files and get analysed results in the graphical formats like bar charts and pie charts. Analyzed results shows total hits for web application, hits per page, hits for city, hits for page from each city, hits for quarter of the year, hits for each page during whole year, etc. Figure 7. gives an idea about total hits from each city, Figure 8. shows total hits per page.

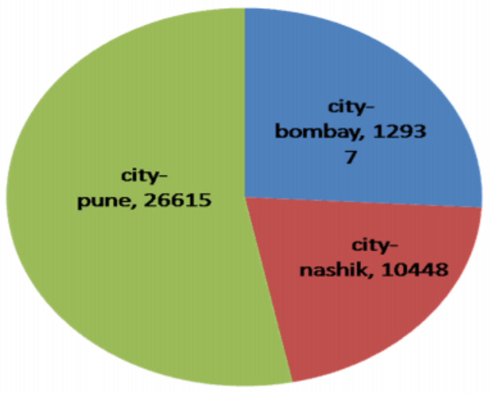

Figure 7. Total hits from each city

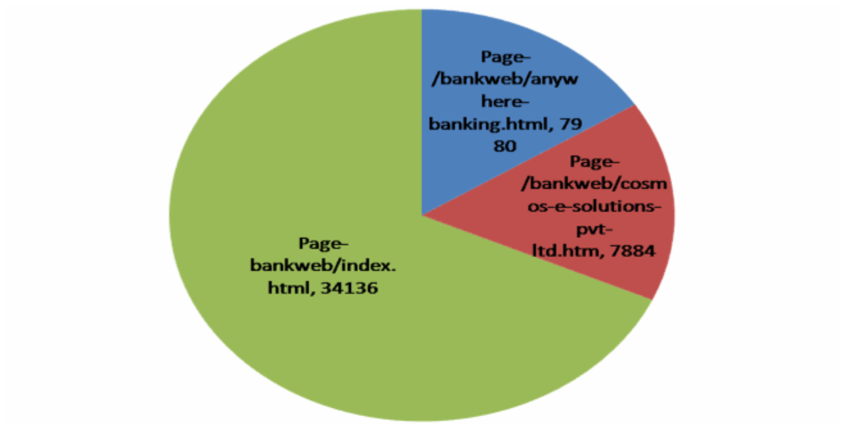

Figure 8. Total hits per page

6. CONCLUSIONS

In order to have a summarized data results for a particular web application, we need to do log analysis which will help to improve the business strategies as well as to generate statistical reports. Hadoop – MR log file analysis tool will provide us graphical reports showing hits for web pages, user's activity, in which part of website users are interested, traffic sources, etc. From these reports business communities can evaluate which parts of the website need to be improved, which are the potential customers, from which geographical region website is getting maximum hits, etc, which will help in designing future marketing plans. Log analysis can be done by various methods but what matters is response time. Hadoop MapReduce framework provides parallel distributed processing and reliable data storage for large volumes of log files. Firstly, data get stored in the hierarchy on several nodes in a cluster so that access time required can be reduced which saves much of the processing time. Here hadoop's characteristic of moving computation to the data rather moving data to computation helps to improve response time. Secondly, MapReduce successfully works for large datasets giving the efficient results.

REFERENCES

[1] S.Sathya Prof. M.Victor Jose, (2011) "Application of Hadoop MapReduce Technique to Virtual Database System Design", International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT), pp. 892-896.

[2] Yulai Yuan, Yongwei Wu_, Xiao Feng, Jing Li, Guangwen Yang, Weimin Zheng, (2010) "VDB-MR: MapReduce- based distributed data integration using virtual database", Future Generation Computer Systems, vol. 26, pp. 1418-1425.

[3] Jeffrey Dean and Sanjay Ghemawat., (2004) "MapReduce: Simplified Data Processing on Large Clusters", Google Research Publication.

[4] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, (2010) "The Hadoop Distributed File System", Mass Storage Systems and Technologies(MSST), Sunnyvale, California USA, vol. 10, pp. 1-10.

[5] C.Olston, B.Reed, U.Srivastava, R.Kumar, and A.Tomkins, (2008) "Pig latin: a not-so-foreign language for data processing", ACM SIGMOD International conference on Management of data, pp. 1099– 1110.

[6] Tom White, (2009) "Hadoop: The Definitive Guide. O'Reilly", Scbastopol, California.

[7] M.Zaharia, A.Konwinski, A.Joseph, Y.zatz, and I.Stoica, (2008) "Improving mapreduce performance in heterogeneous environments" OSDI'08: 8th USENIX Symposium on Operating Systems Design and Implementation.

[8] Mr. Yogesh Pingle, Vaibhav Kohli, Shruti Kamat, Nimesh Poladia, (2012)"Big Data Processing using Apache Hadoop in Cloud System", National Conference on Emerging Trends in Engineering & Technology.

[9] Cooley R., Srivastava J., Mobasher B., (1997) "Web mining: informationa and pattern discovery on world wide web", IEEE International conference on tools with artificial intelligence, pp. 558-567.

[10] Liu Zhijing, Wang Bin, (2003) "Web mining research", International conference on computational intelligence and multimedia applications, pp. 84-89.

[11] Yang, Q. and Zhang, H., (2003) "Web-Log Mining for predictive Caching", IEEE Trans. Knowledge and Data Eng., 15(4), pp. 1050-1053.

[12] P. Nithya, Dr. P. Sumathi, (2012) "A Survey on Web Usage Mining: Theory and Applications", International Journal Computer Technology and Applications, Vol. 3, pp. 1625-1629.

[13] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Michael Stonebraker, (2009) "A Comparison of Approaches to Large-Scale Data Analysis", ACM SIGMOD'09.

[14] Gates et al., (2009) "Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience", VLDB 2009, Section 4.

[15] LI Jing-min, HE Guo-hui, (2010) "Research of Distributed Database System Based on Hadoop", IEEE International conference on Information Science and Engineering (ICISE), pp. 1417-1420.

[16] T. Hoff, (2008) "How Rackspace Now Uses MapReduce and Hadoop To Query Terabytes of Data".

[17] Apache-Hadoop,http://Hadoop.apache.org

Authors

Sayalee Narkhede is a ME student of IT department in MIT-Pune under University of Pune. She has Received BE degree in Computer Engineering from AISSMS IOIT under University of Pune. Her research interest includes Distributed Systems, Cloud Computing.

Tripti Baraskar is an assistant professor in IT department of MIT- Pune. She has received her ME degree in Communication Controls and Networking from MIT Gwalior and BE degree from Oriental Institute of Science and Technology under RGTV. Her research interest includes Image Processing.