Data Analysis using Mapper and Reducer with Optimal Configuration in Hadoop

Sasiniveda.G[#1], Revathi.N[*2]

[1] *PG Scholar,* [2]*Associate Professor*
Department of Information technology,
Sri Venkateswara College of Engineering,
Sriperumbudur – 602105, Chennai, INDIA.

***Abstract*— Data analysis is an important functionality in cloud computing which allows a huge amount of data to be processed over very large clusters. Hadoop is a software framework for large data analysis. It provide a Hadoop distributed file system for the analysis and transformation of very large data sets is performed using the MapReduce paradigm. MapReduce is known as a popular way to hold data in the cloud environment due to its excellent scalability and good fault tolerance. Map Reduce is a programming model widely used for processing large data sets. Hadoop Distributed File System is designed to stream those data sets. The Hadoop MapReduce system was often unfair in its allocation and a dramatic improvement is achieved through the Elastic Mapper Reducer System. The proposed Mapper Reducer function allows us to analyze the data set and achieve better performance in executing the job by using optimal configuration of mappers and reducers based on the size of the data sets and also helps the users to view the status of the job and to find the error localization of scheduled jobs. This will efficiently utilize the performance properties of optimized scheduled jobs. So, the efficiency of the system will result in substantially lowered system cost, energy usage, management complexity and increases the performance of the system.**

***General Terms*- Data analysis, Hadoop, HDFS, MapReduce Paradigm**

Keywords— **Cloud Computing, Hadoop Distributed file System, Performance Paradigm.**

I. INTRODUCTION

Cloud computing provides on-demand access to computational resources which together with pay-per make use of Production models, facilitate appliance provider faultlessly scale their services. [11] To unleash the full power of cloud computing, it is well established that a cloud data processing system should provide a high degree of elasticity, scalability, and fault tolerance. Data analysis is an important functionality in cloud computing which allows a huge amount of data to be processed over very large clusters.Cloud computing provides massive clusters for well-organized data analysis and the huge amount of computation.

MapReduce [5] is a well-known programming model which was first designed for improving the performance of large batch jobs on cloud computing systems. This MapReduce is used for analysing large data sets and its open-source implementation, called Hadoop, for various types of jobs.

This leads to sharing a single Hadoop cluster between [19] multiple users, which run a merge of lengthy batch jobs and short interactive queries on a shared dataset. MapReduce is recognized as a possible means to perform elastic data processing in the cloud [4],[11].There are three main reasons for this. [11] First, the programming model of Map Reduce is simple yet expressive. Second, A large number of data analytical tasks can be expressed as a set of MapReduce jobs, together with SQL uncertainty, data withdrawal device learning, and chart processing. MapReduce achieves the desired elastic scalability through block-level scheduling and is proven to be highly scalable. Yahoo! has deploy MapReduce on a 4,000-node crowd together. Third, MapReduce provides fine-grained fault tolerance whereby only tasks on failed nodes have to be restarted.

Map-Reduce is a programming model that is used to analysis the big data in cloud environment and used to retrieve the data from the hadoop cluster. In this model, processing of large data is efficient, easy to use, it splits the tasks and executes on the various nodes in parallel. Thus it will speed up the computation and retrieve the required data from a huge data set in a faster manner. We introduce the map-reduce programming model [13] for analysis the big data in efficient manner using hadoop. It provides an well-organized data analysis, performance analysis and executes process in parallel distributed manner. By using this programming model, Performance of the system is increased, highly fault tolerant and scalable. Hardware cost is very low. High throughput access to application data. It is highly appropriate for the applications that have large data sets. Easily portable from one platform to another. The Hadoop Distributed File System

(HDFS), [8] is a specialized file system to store large amounts of data across adistributed system of computers with very high throughput and multiple replications on a cluster. [9] It provides reliability between the different physical machines to support a base for very fast computations on a large dataset. This Mapreduce Originally conceived by Google as a way of handling the enormous amount of data produced by their search bots, it has been adapted in a way that it can run on a cluster of normal commodity machines. This is open source and distributed by Apache hadoop.

The rest of the paper is structured as follows: In Section 2 describes the hadoop system model and Section 3 describes about the modelling of mapreduce paradigm Section 4 describes the programming model of the MapReduce Section 5 and 6 presents the design of the mapreduce application programming interfacel used in this work. Section 7 presents the set of experiments and their corresponding results. Finally, Section 8 and 9 presents conclusion and suggests some future work.

II. Hadoop System Model

Hadoop is a free ,Java based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of apache project sponsored by the Apache Software Foundation. The Hadoop framework is used by major players including [7] Google, Yahoo and IBM, largely for applications involving search engines and advertising. Thus the input files that are sending by the client are stored in the Hadoop distributed file system. HDFS is a high level architecture which is the storage system used by Hadoop. This mapreduce programming model[5] is used for processing large set of data and used to execute the data from Hadoop. This splits the tasks and executes on the various nodes simultaneously, thus it speeds up the computation and retrieves required data from a huge dataset in a fast manner.

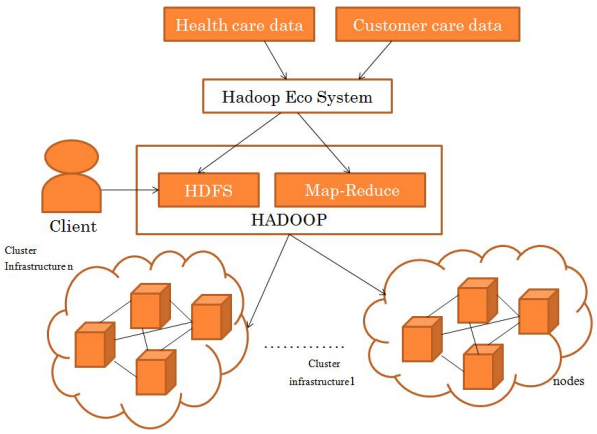

Fig. 1 MapReduce System Model

The above figure1 show the mapreduce system model which allow us to send a query to the hadoop distributed file system by Client. Thus the Hadoop system consists of HDFS and MapReduce. The multiple DataNodes that consider as a Cluster nodes that are in the Amazon EC2 will process the large data set. In general Hadoop consists of one NameNode and Multiple DataNodes which indicates data blocks. When a file is moved to HDFS, it stores them in blocks on the various nodes in the hadoop cluster. HDFS creates several replications of the data blocks and distributed them. When execute a query from a client it will reach out to the NameNode to get the file metadata information and then it will reach out to the DataNodes to get the real datablocks. Here we have considered the 5 nodes cluster in the Amazon cloud. HDFS is highly fault tolerant and is will run on the low cost hardware.

III. Modelling of Mapreduce Paradigm

In existing system while performing MapReduce operations it leads to inconvenience for joining the multiple data sets are tricky and slow, and oftenly the entire data sets gets copied into the process, so there is no indices. Optimal configuration of nodes are not obvious. This system does not schedule the jobs with optimal number of mappers and reducers and does not perform any tracking or monitoring of the jobs progress, and this responsibility is left to the individual node. So Performance of MapReduce programming model is unpredictable.

A. Data Pipelining

Client writes the input files as a block to the first data node. The first DataNode forwards the data to the next DataNode in the Pipeline and the next DataNode forward to all other DataNode in the cluster, and so on. When all [9] replicas are written, the Client moves on to write the next block in file.

B. Description of Data Sets

Here we have considered the health care data set and customer care detail data set. Also we selected wide range of data sets with varying sizes from kilobytes to gigabytes and that are used in our experiments. The health care data set comprises of health details of the customer such as name of the pills and so on. The customer care detail data set comprises of network details.

C. Data Segmentation

Before the MapReduce tasks can be debuted, data sets need to be segmented into chunks. Each chunks are collected for processing the data sets. The number of splited chunks will depends on the size of the dataset and the number of nodes available in the EC2. Users specify a map function that processes a (key, value) pair to generate a set of intermediate (key, value) pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. There are separate Map and Reduce steps. Each step done in parallel on sets of (key, value) pairs [5], [10].Thus, program execution is divided into a Map and a Reduce stage, separated by data transfer between nodes in the cluster.[16] The Map stage takes in a function and a section of data values as input

applies the function to each value in the input set and generates an output set. The Map output is a set of records in the form of (key, value) pairs stored on that node. The records for any given key could be spread across many nodes. The framework, then, sorts the outputs from the Map functions and inputs them into a Reducer [9] , [16] ,[17]. If any of the nodes fails in the hadoop environment, it will still come again the dataset accurately, as hadoop takes care of replicating and distributing the data efficiently across the multiple nodes.

IV. HADOOP DISRIBUTED FILE SYSTEM

It is a high level architecture which is the storage system used by Hadoop. When you dump a file (or data) into the HDFS, it supplies them in block on the various nodes in the hadoop cluster. HDFS creates several replications of the data blocks and distributes them accordingly in the cluster in way that will be reliable and can be retrieved faster. [10] A usual HDFS block range is 128MB. Each and all data block is simulated to multiple nodes across the cluster. Hadoop will internally make sure that any node failure will never results in a data loss. When you implement a query from a customer, it will enter at to the NameNode to get the file information, and then it will reach out to the DataNodes to get the real data blocks Hadoop provides a command line interface for administrators to work on HDFS.

A. MapReduce Programming Model

MapReduce is a parallel programming model for processing large set of data and used to retrieve the data from the Hadoop. This splits the tasks and executes on the various nodes parallel, thus it speed up the computation and retrieve required data from a huge dataset in a fast manner. This provides a clear abstraction for programmers. It has two functions Map and Reduce. The data are fed into the map function as key value pairs to produce intermediate key/value pairs .Once the mapping is done, all the intermediate results from various nodes are reduced to create the final output. Job Tracker keeps track of all the MapReduces jobs that are running on nodes. This schedules the jobs, keeps track of all the map and reduce jobs running across the nodes. If any one of those jobs fail, it reallocates the job to a different node. Task Tracker performs the map and reduce tasks that are assigned by the Job Tracker.

"Map" step: The master node takes the input, divide it into lesser sub-problems, and distributes them to worker nodes. [1] A worker node may do this yet again in turn, chief to a multi-level tree structure. The worker node process the less important problem, and passes the answer back to its master node.

"Reduce" step: The master node then [1] collects the answers to all the sub-problems and combines them in some way to form the output.

B. Elastic MapReduce system

A typical MapReduce computation processes many terabytes of data on thousands of machines. MapReduce typically split the input dataset into self-determining chunks. The number of splits depends on the size of the dataset and the number of nodes available. Users specify a map function that processes a (key, value) pair to generate a set of intermediate (key, value) pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. There are separate Map and Reduce steps. Each step done in parallel on sets of (key, value) pairs. Thus, program execution is divided into a Map and a Reduce stage, separated by data transfer between nodes in the cluster.

The Map stage takes in a function and a section of data values as input, applies the function to each value in the input set and generates an output set. The Map output is a set of records in the form of (key, value) pairs stored on that node.[17] The records for any given key could be spread across many nodes. The framework, then, sorts the outputs from the Map functions and inputs them into a Reducer. This involves data transfer between the Mappers and the Reducer. The values are aggregated at the node running the Reducer for that key. The Reduce stage produces another set of (key, value) pairs as final output. The Reduce stage can only start when all the data from the Map stage is transferred to the appropriate machine. MapReduce requires the input as a (key, value) pair that can be serialized and therefore, restricted to tasks and algorithms that use (key,value) pairs.

The MapReduce framework has a single master Job Tracker and multiple Task Trackers. Potentially, each node in the cluster can be a slave Task Tracker. The master manages the partitioning of input data, scheduling of tasks, machine failures, reassignment of failed tasks, inter-machine communications and monitoring the task status. The slaves execute the tasks assigned by the master.[12] Both input and output are stored in the file-system. The single JobTracker can be a single point failure in this framework. MapReduce is best suited to deal with large datasets and therefore ideal for mining large datasets of petabytes size that do not t into a physical memory. Most common use of MapReduce is for tasks of additive nature. However, we can tweak it to suit other tasks.

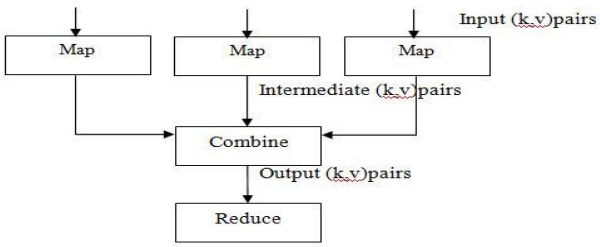

Fig. 2 Elastic Mapreduce Model

MapReduce is a simple and powerful programming model that

enables easy development of scalable parallel applications to process the vast amount of data on the commodity machines. In this model The computation takes a set of input key/value pairs and produces a set of output key/value pairs. For optimal configuration, the elastic Mapper Reducer System is used and improves the performance of the scheduled jobs. The Map phase takes a key-value pair generated through the Input Split. Each node runs one map task and is run parallel with each other. One Map task takes a key-value pair, process it and generates another key-value pair.

V. MAPPER API

For optimal configuration, the Elastic Mapper Reducer is used to schedule the jobs efficiently.HDFS creates a several replication of the data blocks and distributes them. In Map function the Jobtracker takes the input, divides it into smaller sub-problems and distributes them to Tasktracker [9]. The MapReduce framework operates exclusively on <key, value> pairs. Map function takes input as Key/Value pairs where Key is reference to the input values and Value is data set on which to operate the particular node. Internally this file is split into one or more block. Hadoop provides a command line interface to work on HDFS.

- Map (Key1,Value1) ---> emit (Key2,Value2)
- Reduce (Key2,Value2_list) ---> emit (Key2,aggregated_Value2)
- Combine (Key2,Value2_list) ---> emit (Key2,combined_Value2)
- Partition (key2) return reducer No

Each MapReduce job is split into tasks that comprise sequences of key-value pairs. For each task, we create a mapper object, which calls the map method for each key-value pair and this function is applied in parallel to every pair in the input data set [10]. The below graph shows the performance of mapreduce. Thus, figure 3 Graph shows for the various optimal configuration of Mapper and Reducer. This is plotted between the Number of Tasks and Time.

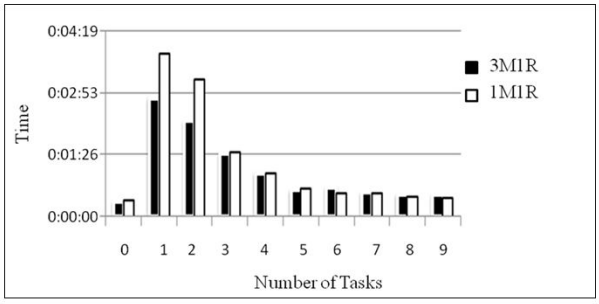

Fig. 3 Map/Redeuce Task Completion Graph

This produces a list of call for pair of each call. After that mapreduce framework collect all pairs with the sane key from the entire list and group them together. Thus it will create a one group with different keys. Then the reduce function is also applied parallely in each group which in turn produces a collection of values in the same domain. Each reduce call typically produce either one value or empty return. By using this We provide an efficient data analysis, and it improves the performance by executing the process in a distributed or in parallel manner.

VI. REDUCER API

The Tasktracker then collects the answers to all the sub-problems and combines them to form the output. The reduce function is then applied in parallel to each group for each one of the different generated keys. Reduce function starts with intermediate Key/Value pairs and ends with finalized Key/Value pairs.

TABLE I
PARAMETER RECEIVED BY MAPREDUCE

Parameter Received	Map (Bytes)	Reduce (Bytes)
File Bytes Read	0	543
HDFS Bytes Read	433	0
File Bytes Written	22,265	22,234
HDFS Bytes Read	0	365
CPU Time(ms)	200	980

Each of the jobs scheduled and produces a job id. This shows the status and the information parameters that are received from the TaskTracker. Thus, optimal configuration of Elastic Mapper Reducer gives some parameter such as Bytes read, Bytes written, CPU time, Memory limit, Dead nodes, Live nodes, Data size, Communication delay etc..,This will also show the status of the Namenode ang gives the information about the Heap size, Capacity of the node and also details of the Live nodes and Dead nodes.

VII. ADVANTAGE

The performance of the Elastic Mapper Reducer is achieved by using the optimal configuration by scheduled jobs. It provides the fine grain fault tolerance, so only the tasks on the failed nodes have to be restarted. Elastic Mapper Reducer function allows us to analyze the metadata set and achieve better performance in executing the job by using optimal number of mappers and reducer based on the size of the data sets and also helps the users to view the status of the job and to find the error localization of scheduled jobs. This will efficiently utilize the performance properties of optimized scheduled jobs. So, the efficiency of the system will result in substantially lowered system cost, energy usage, and management complexity it increases the performance of the system.

VIII. CONCLUSION

Elastic MapReduce programming model is performed efficiently for processing complex data sets. Thus optimal configuration of Mappers and Reducers are Performed based on the size of the data sets. This provides the information about the parameters that are received from the scheduled jobs. So, the efficiency of the system will result in substantially lowered system cost, energy usage, and management complexity it increases the performance of the system. This improves the overall system performance, efficiency and scalability. If any one of those jobs fails, it reallocates the job to another node and process the data in efficient Manner.

IX. FUTURE WORK

This design is enhanced by considering the Mean shift clustering Model for the desired data set and is used to show the Elastic MapReduce programming model efficiently and its better performance will be evaluated and enhanced MapReduce system will be deployed in cloud as a service which can be used by the user across the world.

REFERENCES

[1] Apache,"Hadoop," http://hadoop.apache.org/docs/r0.20.2/hdfs_design.html

[2] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Machine Intell., 24:603–619, 2002.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," Proc. 19th ACM Symp. Operating Systems Principles, 2003.

[4] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D. Stott Parker. Map-reduce-merge: simplified relational data processing on large clusters.In SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 1029–1040, New York, NY, USA, 2007. ACM.

[5] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, I. Stoica. Improving MapReduce Performance in Heterogeneous Environments. In OSDI, USENIX Symposium on Operating System design and Implementation pp.1-16August

[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified Data Processing on Large Clusters," Comm. ACM, vol. 51, no. 1, pp. 107-113,December 2008.

[7] Hadoop, http://lucene.apache.org/hadoop

[8] Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2

[9] Konstantin Shvachko," The Hadoop Distributed File System", Yahoo-Inc.com.

[10] T. Sun, C. Shu, F. Li, H. Yu, L. Ma, Y. Fang, An efficient hierarchical clustering method for large datasets with map-reduce, in: PDCAT'09: International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE Computer Society, Washington, DC, USA, 2009, pp. 494-499.

[11] Matei Zaharia, Dhruba Borthakur, Job Scheduling for Multi-User MapReduce Clusters Electrical Engineering and Computer SciencesUniversity of California at Berkeley April 30, 2009.

[12] D. Jiang et al. Map-join-reduce: Towards scalable and efficient data analysis on large clusters. IEEE Transactions on Knowledge and Data Engineering, 2010.

[13] D. Jiang et al . The performance of mapreduce: An in-depth study. Proceedings of the VLDB Endowment,3(1-2):pp 472–483, 2010

[14] M. Elteir, H. Lin, W. chun Feng, Enhancing mapreduce via asynchronous data processing, in: ICPADS'10: IEEE 16th International Conference on Parallel and Distributed Systems, 2010, pp. 397-405.

[15] Mr. Yogesh Pingle, Vaibhav Kohli, Shruti Kamat, Nimesh Poladia Big Data Processing using Apache Hadoop in Cloud System International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622.

[16] F.N. Afrati and J.D. Ullman, Optimizing Joins in a Map-Reduce Environment, Proc. 13th Int'l Conf. Extending Database Technology (EDBT '10), 2010.

[17] Y. Bu, B. Howe, M. Balazinska, and M. Ernst, "Hadoop: Efficient Iterative Data Processing on Large Clusters," Proc. VLDB Endowment, vol. 3, no. 1/2, pp. 285-296, 2010.

[18] Foto N. Afrati and Jeffrey D. Ullman, Optimizing Multiway Joins in a Map-Reduce Environment IEEE Transactions on knowledge and data Engineering, VOL. 23, NO. 9, September 2011.

[19] Indranil Palit and Chandan K. Reddy, Scalable and Parallel Boosting with MapReduce IEEE Transactions on knowledge and data Engineering, VOL. 24, NO. 10, October 2012.