Performance Tuning and Scheduling of Large Data Set Analysis in Map Reduce Paradigm by Optimal Configuration using Hadoop

Sasiniveda. G
PG Scholar, Sri Venkateswara
College of Engineering,
Pennalur,
Chennai-602105,Tamil Nadu

Revathi N
Asst.Prof, Sri Venkateswara
College of Engineering,
Pennalur,
Chennai-602105,Tamil Nadu

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 Mapper Reducer System. The proposed Mapper Reducer function using the mean shift clustering based algorithm 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 tuning 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 Tuning, Mean shift Clustering, Amazon web services.

1. INTRODUCTION

Cloud computing provides massive clusters for efficient large scale computation and data analysis. MapReduce [3] is a well known programming model which designed for improving the performance of large batch jobs on cloud computing systems. However, there is growing interest in employing MapReduce and its open-source implementation, called Hadoop, for various types of jobs. This leads to sharing a single Hadoop cluster between various users, which run a merge of lengthy

group jobs and short interactive queries on a shared dataset. 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. When hadoop clusters shared with multiple users it provides several advantages such as fairness, Data Locality, increases the utilization of the resources.

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 and also the performance of the mapreduce system greatly increased by executing the process in a distributed or in parallel manner by implementing Fair scheduling in Map Reduce paradigm. In this paper we parallelize a data mining algorithm , and scheduling of multiple jobs , performance tuning of Map Reduce and evaluate all the parameters for the optimizing the Map Reduce paradigm across multiple nodes of a Cloud Environment.

The Hadoop scheduler is the centrepiece of a Hadoop system. Desired performance levels can be achieved by proper submission of jobs to resources. Primary Hadoop scheduling algorithms, like the FIFO algorithm and the Fair-sharing algorithm, are simple algorithms which use small amounts of system information to make quick scheduling decisions.

Here mean shift clustering based algorithm is used. It is non-parametric mode of clustering procedure. It does not require prior knowledge of the [2] number of clusters, and does not constrict the figure of the clusters. It will produce arbitrarily shaped clusters that depend upon the topology of the data. In Mean Shift Mapper Reducer function random points is generated based on the probability density function. Mode value is used to calculate density function value and it runs multiple iterations. Centroid points change frequently and on one particular point it will attain the threshold value. By this mode value we can able to find the location of the data i.e. in the data node or particular cluster. Mean shift are automatically generate the random value based on the density function. Here mean shift is used as a multithread. So this will increases resource utilization,

The remainder of this paper is structured as follows. In Section 2 we give a brief summary of a Hadoop system. Current Hadoop scheduling algorithms are given in Section 3. Our Hadoop system model is described and formally introduced in Section 3. Then, in Section 4, we formally present the performance metrics of interest. Our proposed

Hadoop mean shift clustering scheduling algorithm is introduced in Section 5. In Section 6, details of the environment in which we study our algorithm are provided,and some advantages of the work is explained and we study the performance of our algorithm in various Hadoop systems. Finally, we provide some concluding remarks and discuss possible future work in the last section.

2. 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.

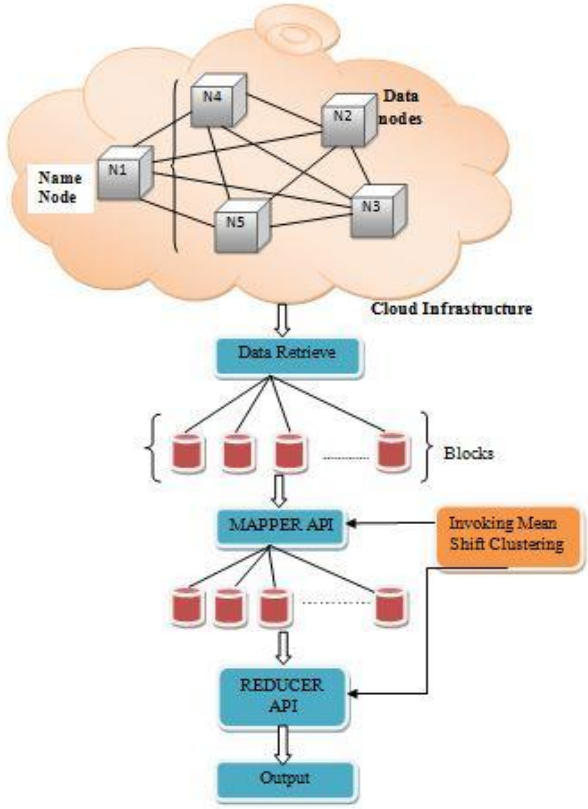

Fig 1: Map Reduce System Model

The above figure1 show the mapreduce system model which allow us to create 5 datanodes in the cloud infrastructure. This AWS cloud is created in amazon web server this can be accessed by any where across the world. It consists on 1 Namenode and 4 Datanodes and together forms a cluster like infrastructure and all the nodes are interconnected. Thus the client send a query to namenode and searches the required datas with the namenode,this will perform the mapreduce operation using hadoop. All the files are stored in the hadoop distributed file system by Client. Thus the Hadoop system consists of HDFS and MapReduce. HDFS is highly fault

2.4 Execution Overview

The Map invocations are distributed across multiple machines by automatically partitioning the input data into a set of M splits. The input splits can be processed in corresponding by dissimilar machines. Reduce invocations are disseminated by partition the intermediary key space into R pieces using a partitioning function (e.g.,hash (key) mod R). The number of partitions (R) and the partitioning function are specified by the user. When the user program calls the MapReduce function, the following series of actions occurs

1. The MapReduce documentation in the customer program first splits the input files into M pieces of typically 16 megabytes to 64 megabytes (MB) per piece (controllable by the user via an optional parameter). [4] It then starts up many copies of the program on a cluster of machines.

Thus mean shift clustering states that in the map phase the algorithm "select a point as the center and compute distances from the center to all the data points, and assign a label to the points within the bandwidth" and in reducer phase "collect data points of the same label and compute the center of mass of them".

3.IMPLEMENTATION OF MAPREDUCE PARADIGM

This topic consists of detailed description of each and every module with its advantages and data and execution flow of each module with algorithm. It helps to understand each and every module of the project more deeply and clearly. Each description consists of the basic concept of the module, input and also the excepted output.

2.1 Modules

The project has been divided into various modules and each module has been completed within a scheduled time line. The following are the modules of the project are cluster Hadoop Ecosystem ,Cloud Computing - IAAS (Amazom web services), set up a cluster of 3 nodes, using Mean shift Based clustering the map and reduce operations are performed.

Machines are typically dual-processor x86 processors running Linux, with 2-4 GB of memory per machine. Commodity networking hardware is used typically either 100 megabits/second or 1 gigabit/second at the machine level, but averaging considerably less in overall bisection bandwidth. A cluster consists of hundreds or thousands of machines, and therefore machine failures are common .Here we have considered 5 node homogeneous cluster in the amazon web services. Storage is provided by inexpensive IDE disks attached directly to individual machines. A distributed file system [3] developed in-house is used to manage the data stored on these disks. The file system uses duplication to afford availability and reliability on top of unreliable hardware.Users submit jobs to a setting up system. Each work consists of a set of everyday job, [4] and is mapped by the scheduler to a set of available machines within a cluster.

2.2 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.

2.3 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.

2. One of the copy of the program is individual to the master. The rest are workers that are [4] assigned work by the master. [4] There are M map tasks and R reduce tasks to assign. The master picks [12, 4] idle workers and assigns each one a map task or a reduce task.

	Block Size	Execution Time
Number of Input Block size	64MB	3Hrs 27 Min
	128 MB	3Hrs 5 Min
	256 MB	2Hrs 42 Min
	512MB	2Hrs 29 Min
DFS Block size	64 MB	
	Replication factor=1	4 Hrs 13Min
	Replication factor=3	3Hrs 27 Min
Compression	64MB	
	With	3 Hrs 6 Min
	Without	3 Hrs 27 Min
JVM	64MB	
	With Reuse= -1	-3 Hrs 8 Min
	Without Reuse=0	3 Hrs 27 Min
Reducers	64MB	
	One	3 Hrs 27 Min
	Three	3 Hrs 27 Min (Because of using three unique keys only)
Combiner	64MB	
	With	3 Hrs
	Without	3 Hrs 27 Min
Scalability (default configuration)	1 node	10 Hrs 8 Min
	2 node	7 Hrs 21 Min
	3 node	5 Hrs 34 Min
	4 node	3 Hrs 27 Min
io.sort.mb	64MB	
	100 MB	3 Hrs 27 Min
	200 MB	3 Hrs 27 Min
	128 MB	
	100 MB	3 Hrs 27 Min
	200 MB	3 Hrs 27 Min
	256 MB	
	100 MB	3 Hrs 27 Min
	200 MB	3 Hrs 27 Min
	512MB	
	100 MB	3 Hrs 10 Min
	200 MB	3 Hrs 5 Min

3. A member of staff who is assigned a map task reads the stuffing of the corresponding input split. [4] It parses key/value pairs out of the input data and passes each pair to the user defined Map function.[4] The intermediate key/value pairs produced by the Map function are buffered in memory.

4. Occasionally, the buffered pairs are written to local disk, partitioned into R regions by the partitioning function. [4] The locations of these buffered pairs on the local disk are passed back to the master, who is in charge for forwarding these locations to the reduce workers.

5. When a reduce worker is notice by the master about these location, it uses isolated method calls to read the buffered data from the limited disks of the map workers. When a reduce worker has read all in-between data, it sorts it by the intermediate keys so that all occurrence of the same key are grouped collectively. The sorting is needed because typically many dissimilar keys map to the same reduce task6. The reduce hand iterates over the sorted in-between data and for each distinctive in-between[4] key encounter, it pass the key and the parallel set of intermediary values to the user's Reduce function. The productivity of the Reduce function is append to a final output file for this reduce partition.

6. When all map tasks and reduce tasks have been completed, the master wakes up the user program. [4] At this point, the MapReduce call in the user program returns back to the user code. After [4] successful completion, the output of the mapreduce execution is available in the R output files [4] (one per reduce task, with file names as specified by the user).

2.5 Node Configuration

- EC2 small instances
- Need 1.75 memory
- One EC2 complete unit (1 virtual core)
- 64 bit Amazon Linux
- 160 GB instance storage

Table I Performance of MapReduce jobs

3. MEAN SHIFT ALGORITHM

MS(Mean shift) is a nonparametric, iterative method for unsupervised clustering and global/local optimization. It has a wide range of applications in clustering and data analysis. For this the following steps has been followed

- Choose a search window size
- Choose the initial location of the search window
- Compute the mean location(centroid of the data)in the search window
- Center the search window at the mean location computed in step 3
- Repeat step 3 and 4 until convergence

4. TUNNING MAPREDUCE JOBS

Each map or reduce task finishes in less than 30-40 seconds. A large job does not utilize all available slots in the cluster. After most mappers or reducers are scheduled, single or multiple leftovers awaiting and then run all alone.

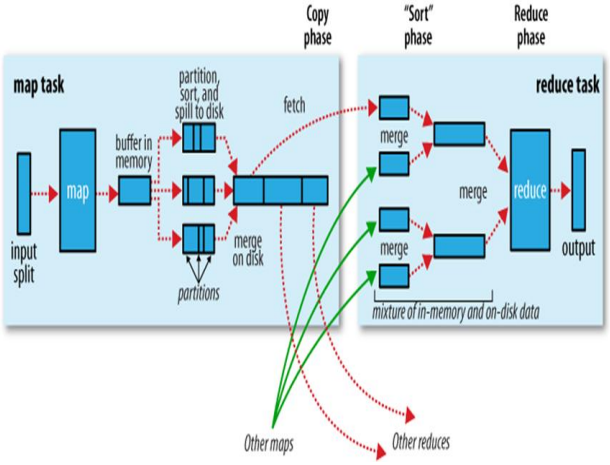

Fig 2: Overview of MapReduce Jobs

In the above figure 2 overview of mapreduce jobs are described. It has two phases. Map phase takes the input and flow to the buffer for storage and it will split the tasks and partition to the disk. All the partitions on the disk fetches the result and send to the reduce phase. This will mix the memory content and disk on the data to produce the reduce output. The cluster specification includes

- Datanodes+tasktrackers = 4 machines
- Namenode+jobtracker = 1 machine

Each machine has 4GB main memory,8 core CPU, 5 TB Hard drive Cloudera Hadoop 0.18.3.

5. MAPREDUCE IMPLEMENTATION

To simplify deployment on the cluster, the JAR-file was built to include any libraries it uses in it's build path. This provides a relatively large file compared to without these libraries. This does however remove the need to include libraries on launch from the MapReduce framework itself. The largeness of the files fits the HDFS, but with XML files there is a slight problem of tags existing outside of file splits. When splitting files, HDFS and MapReduce does not account to any specific split locations by default, so This utilizes the MapReduce Record Reader's ability to read outside of the split given, so the InputFormat can read from a tag start to a tag end. To process the data following steps has to be done

- Start a specific amount of Hadoop VMs.
- Configue them accordingly.
- Upload the data and the JARfile to the master VM.
- Put the data onto HDFS.
- Run the job 5 times, making sure to remove the output folder between runs.
- Go to the JobTracker Web Service and analyse the time.

Table II Parameter received by MapReduce Jobs

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 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.

6. ADVANTAGES

The performance tuning 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. Mapper Reducer function allows us to analyse 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.

7. CONCLUSION

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. This will increases the synchronization and parallel computing process.

8. FUTURE WORK

This MapReduce model can be used efficiently in the hadoop version 1 and the proper autoscaling for a Hadoop cluster can be done more efficiently using the later version of hadoop.

MapReduce system will be deployed in cloud as a service which can be used by the user across the world.

9. 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] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. In 19th Symposium on Operating Systems Principles, pages 29–43, Lake George, New York, 2003.

[4] Jeffrey Dean and Sanjay Ghemawat "Map Reduce: Simplified Data Processing on Large Clusters" International Journal of Engineering Research and Applications ISSN: 1 – 13, July 2004.

[5] Matei Zaharias, Andy Konwinski, et al "Improving Map Reduce Performance in Heterogeneous Environments" IEEE Transactions on Parallel and distributed processing, Vol. 23, No. 19, April 2010.

[6] Quan Chen, Daqiang Zhang, et al. "SAMR: A Self-adaptive Map Reduce Scheduling Algorithm In Heterogeneous Environment" International Journal of Engineering Research and Applications ISSN: 2736-2743, July 2010.

[7] Mohammad Farhan Husain, James Mc Glothlin, et.al "Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing" IEEE Transactions on knowledge and data Engineering, Vol. 23, No. 9, September 2011.

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

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

[10] Kyong -Ha Lee, Hyunsik Choi "Parallel Data Processing with MapReduce: A Survey" International Journal of Engineering Research and Applications Vol. 40, No. 4 December 2011.

[11] Nikzad Babaii Rizvandi1,Albert Y. Zomaya , et.al " On Modeling Dependency between Map Reduce Configuration Parameters and Total Execution Time " IEEE Transactions on Distributed, Parallel, and Cluster Computing , Vol. 23, No. 9, March 2012.

[12] Gabriel G. Casta, Alberto Nunez, et al. "Dimensioning Scientific Computing systems to improve performance of Map-Reduce based applications" International Journal of Engineering Research and Applications ISSN: 226 – 235, July 2012.

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.