Improving Performance of Heterogeneous Hadoop Clusters using Map Reduce for Big Data

Dr. Mahesh Kumar Singh
Assistant Professor, Department of Computer Science and Engineering
Bansal Institute of Engineering and Technology, Lucknow, Uttar Pradesh, India

Alok Sachan
M.Tech Scholar, Department of Computer Science and Engineering
Bansal Institute of Engineering and Technology, Lucknow, Uttar Pradesh, India

Abstract- The key problem that arises due to enormous growth of connectivity between devices and systems is creating so much data at an exponential rate that a feasible solution for processing it is becoming difficult day by day. Therefore, developing a platform for such advanced level of data processing, hardware as well as software enhancements need to be conducted to come in level with such magnanimous data. To improve the efficiency of Hadoop clusters in storing and analyzing big data, we have proposed an algorithmic approach that will cater the needs of heterogeneous data stored over Hadoop clusters and improve the performance as well as efficiency. The proposed paper aims to find out the effectiveness of new algorithm, comparison, suggestions, and a competitive approach to find out the best solution for improving the big data scenario. The Map Reduce technique from Hadoop will help in maintaining a close watch over the unstructured or heterogeneous Hadoop clusters with insights on results as expected from the algorithm.

Keywords: Big data, Hadoop, Heterogeneous clusters, Map reduce, Throughput, Latency

I. INTRODUCTION

The studies conducted so far demonstrates that roughly 30% of data exhibited from digital world can be useful for various purposes if analyzed properly. But a mere 0.5% of data is utilized by the existing means which states clearly the inadequacy posed in big data field because of limited capabilities in existing systems. The current analysis of unstructured data residing on servers or clusters reveals the conflicting influences of data over systems. The Map Reduce model can provide high throughput, fairness among job distribution or low latency. But the speed at which it is done needs to be improved to cope up with increasing and unorganized unstructured data. Another issues to be taken care of are, turnaround times, effective clustering techniques, algorithms to sort data, structure it and retrieve data with high throughput and low latency.

Hadoop, the open source framework to store and analyze data over low-cost hardware has made a big round of buzz among developers and organizations. The Hadoop entertains the storage, analysis, and retrieval of data using clusters of nodes operating under it to manifest the possibility of screening large data sets that are quiet inefficient for relational databases. It is designed and aided in such a manner that a single framework is enough to scale thousands of servers complemented with fast local computing and storage. The feature that makes it useful in fast paced development scenario is the fact that it can equally monitor structured as well as unstructured data that dominates the Internet using Map Reduce. Big data comprises of data that is so large that and complex that traditional databases are incompetent to store, or analyze it. The big data can be understood by four V's, i.e. Variety, Velocity, Value and Volume.

Big data consists of predictions to derive values from very large databases. The analyzing and processing involved determines how sufficient value could a data or frame provide. It involves prying with data that could deliver some useful and meaningful content by storing, sharing, search, visualizing, or querying. Business Intelligence and statistics are deeply associated with it. Other relative fields like language processing, artificial intelligence, astronomy, genomics, weather conditions, or any field that holds a magnanimous amount of data are all related to it. Data is generated with exponential quantity today, to tackle this data; big data involves predictions, learning and involves complex calculations and algorithms to provide some value. The coming years will see a huge rise in need of highly calculative and processing applications to deal with big data. There are a number of applications and frameworks that use big data but are still at a loss at many points.

There are algorithms that classify the data based on their own set of principals which provide a smooth utility to clustering. The algorithms thus used provide a way to account all the data and associate them during Map Reduce.
The objective is to identify existing studies related to Hadoop, Big Data, and Map Reduce framework and work with the problems that are commonly encountered, to deal with large sets of statistical or unstructured data and review them for the dissertation work, to perform the comparative study of existing algorithms that are used for clustering data on heterogeneous Hadoop clusters and which one of them suits best for future trends, to design a new algorithm that removes the shortcomings of existing algorithms to provide a better functionality than the previous ones, and to use the algorithm to improve performance of heterogeneous Hadoop clusters using Map Reduce model for big data and display the results.

The proposed work identifies and improves such challenges to make the community better and useful versions than previous ones. With big data, one could improve sales, marketing, discovery of new medicines, devising strategies for lay out and a whole new world of tasks.
Most of the problem in handling big data arises from the unlabeled objects which are hard to tackle in Hadoop clusters.

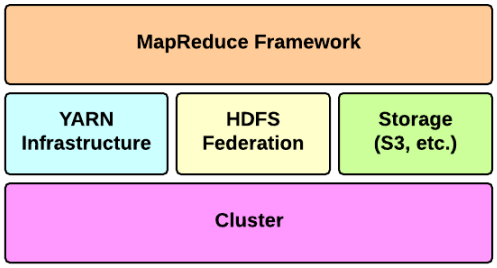

Figure1. Architecture of Hadoop

1.1 Background Work-

The internet is expanding at exponential rates; given the current scenario scientists evaluate that in a matter of few year times, the size of data will exceed thousands of petabytes. To cope with such information or specifically data burst, one needs a good system to not only store and retrieve the data but also be able to query and analyze it. The big data scenario presents challenges in itself as to store the data typically which is a much unorganized manners. Primarily the data that one need today is important to scientific, retail, or business based. So in order to manifest the data to turn up some profit or simply understand the patterns, one needs to dive deep and conclude to a meaningful interpretation. There are several platforms to store data. Traditionally, relational databases offered a wide variety of options to store and properly analyze data but as the data limits and types of data exceeded current trends, it became vitally important to store unlabeled or unstructured data. The unstructured data here represents any kind of data that is in raw form and is simply a collection of datasets where one chunk might differ from another very much.

As Zholiu points out, over 33% of the digital data can prove valuable if we have proper tools and methods to analyse it whereas the current uses scores up to mere 0.5%. These kinds of observations initiate the need of highly efficient, scalable and flexible approaches to help analyze gigantic size of big data. Map Reduce algorithms over a small time have proved their importance as efficient, scalable, and fault-tolerant data analytics programming model but somewhat lacks query-level semantics in task scheduling resulting in low utilization of system resources, prolonged execution time and low query throughput. Map Reduce offers task-level scheduling through Directed Acyclic Graphs (DAG) and set of query languages aided with data warehousing capabilities.

The scalability of larger clusters in Hadoop is generally heterogeneous. Consider the case where a website gathers content over continents. Typically, it will include different platform users with languages and format suitable to their own. The website has to have complacency with the heterogeneity and to store such data over clusters is a big task. The clustering involved will have to set parameters to associate those data with enticed values so that clusters could be easily identified and used for queries. As mentioned by [Zingben Xu, et al] cluster configuration highly effects the parallelization of data in Hadoop. The challenge among the heterogeneity of the clusters is to produce and handle infrastructures for computing that harnesses not only existing computing platforms but also with low-cost equipment provide better results at hand.

The computing is largely in continually in need of tools for dealing with structured and unstructured data. There exists lack of performance of Hadoop's heterogeneous clusters which implement further problems in distributed

processing. The performance bottlenecks of such systems as addressed by, [Krushnakanth Kametkar] states that the hardware configurations, logical connectivity and proper underlying framework can reduce the problems faced by heterogeneous Hadoop clusters. The clusters could be managed by following a minimum requirement over a specific number of nodes in a cluster to avoid data node failure, replication techniques, rack maintenance etc. As also noted by [Jiong Ji et al], there clearly exists a distinction where low performance clusters are surpassed by high-performance clusters in terms of local data transfers. By distributing the load between high and low performance heterogeneous clusters, there is a significant improvement in Map Reduce tasks.

II. PROPOSED ALGORITHM

The dissertation work focuses on improving the performance of heterogeneous clusters on Hadoop by following a set of phases that improves the data input/ output queries, improves the routing of algorithm to heterogeneous clusters, and then improving the performance of processing of queries in such a way that they are easily connected to the right part of execution in minimal time as possible without increasing the costs at computing level. The proposed work follows a series of steps to handle these scenarios which are described in subsequent sections. A series of steps or phases are added to improve the efficiency of heterogeneous clusters. Each step is explained below:

a) Query Improvisation:

When queries are fetched into the parser and semantic analyzer, they invariably compute the dependencies among the queries. But once this parsed query is sent to the Hadoop's Map Reduce to execute the dependencies that were calculated for the Hive query is lost between the transitions. Once the dependencies are calculated they can be used for semantics extractions in Hive QL processor. In the second step we can use these dependencies such as logical predicates and input tables to process the dependencies among different queries to be closely attached to each other during transition. When the sematic gap between Hive and Hadoop is bridged by these intermediary steps, they can be easily used for clustering similar queries at query level, and hence by minute steps improving the query job execution.

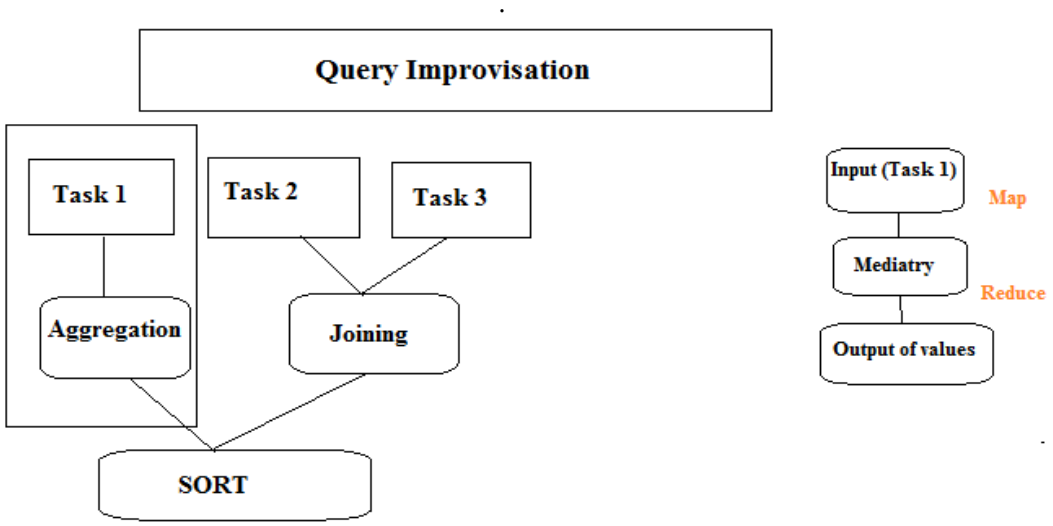

Figure 2. Query Improvisation process

b) Query Scheduling Algorithm:

Hadoop uses First in First out (FIFO) scheduler for job distribution by default. But there are other schedulers as well like, Hadoop Fair Scheduler (HFS) and Hadoop Capacity Scheduler (HCS) that are customizations for Hadoop ecosystem. To have a good fairness among different jobs, the HCS and HFS are sufficient. In the proposed methodology, both the schedulers are used one by one to find the outcome which can help in maintaining order among different capacity clusters.

c) Clustering Algorithm:

The algorithm proposed under this dissertation work is to simplify the need to categorize big data that arrives in dynamic fashion. If the clusters of data could be improved effectively.

2.1 Algorithm Used-

Input: Datasets cleansed of garbled values.

Tq = Read_Queries from Query.DB();
Fq = Frequency of item dataset
Ci = item in Cluster i ds = Similarity matrix from
Clustering through K- Means Algorithm.
Compute fq

Output: Clustered data of importance.

1. for Q=1…n, Then n= Number of log transactions
2. Compute, IQ = I. getQuery();
3. Loop in, for i = 1…x
4. then if (IQ is in (Ci))
5. then, compute fq = 1;
6. Compute ds =Compute distance (Similarity matrix);
7. Else, compute IQ = New(Ci);
8. End If;
9. End for i;
10. End for Q;

The algorithm finds the frequency of an item that is of importance and is associated to a similar Matrix

2.2 Implementation of Map Reduce-

Map:

Step 1: Input unstructured data file: (log_file.txt)
Step 2: Function that divides data sets: (log_file_intermediate)

Split(log_file.txt): (lf1)+(lf2)+(lf3)….+(lfn)

Step 3: Collect Output: (log_file_output; lfId, 1)

Reduce:

K: Count of the individual terms.
Step 1: Input file which was output from map function: (log_file_output; lfId, [k])
Step 2: Function to summarize the intermediate terms and give a final value to all the detected valid terms in the module:

p=Sum (k)

Step 3: Provide output to sum up the file values. (log_part, countId, p)

2.3 *Applying TF-IDF-*

As described in the earlier parts, TF-IDF (Term Frequency- Inverse Document Frequency) provides an equivalent measure of the weight of information retrieval system. The queries on which a TF-IDF weight is to be calculated, implores a good result on the filtered words that are more important than the others in a classification system. For this particular requirement of categorizing big data sets.

III. Experiment and Result

The following results have been established by the conductance of the proposed work, the graphs are depicted on the formula,

$$\text{Throughput (Nq)} = \sum_{q=0}^{Q} filesize(x) / \sum_{q=0}^{Q} time(x) \quad (1)$$

The estimation of time taken by a query is calculated through this throughput. The following graph depicts the efficiency as demonstrated by the different schedulers based on their throughput.

The rise of amount of data digitally poses challenges that need solutions to maintain them as well as to identify the correct sets of information to maximize the utilization with cost benefits. The two level query processing technique can be used to improve the efficiency of such problems and classifying data with the help of Map Reduce on Big Data.

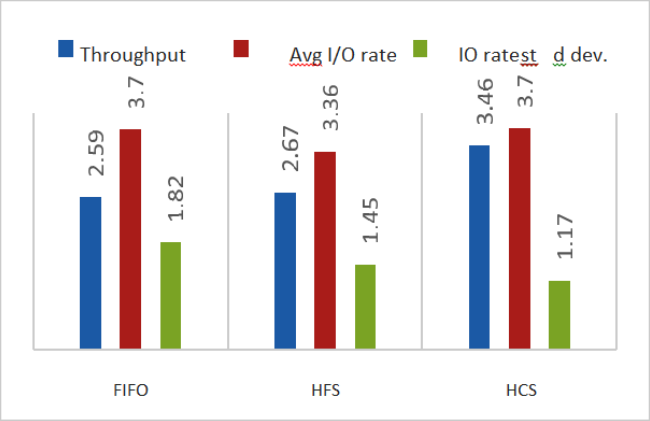

Figure 3: Comparison of different Hadoop Schedulers

The test execution time taken by all the different schedulers is as below:

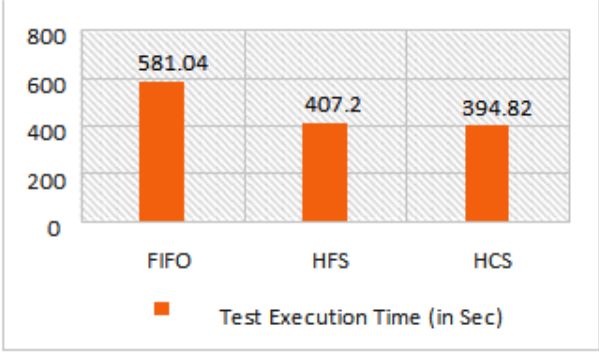

Figure 4: Test Execution time for different Schedulers

Generating a new algorithm to solve these problems for the commercial as well as non-commercial uses can help the betterment of community. The proposed algorithm can help improve the placement of data classifying algorithm Map Reduce in heterogeneous Hadoop clusters.

IV.CONCLUSION

The dissertation work and experiments conducted under this work have emulated quite surprising results, some of them being the choice of schedulers to schedule jobs, placement of data in similarity matrix, clustering before scheduling queries and moreover, iterative, mapping and reducing and binding the internal dependencies together to avoid query stalling and execution times. The major points of thought that were resulted as conduction of the dissertation work are as following:

Calculating query dependencies: The queries that are analyzed in the parsers, [2] evaluate a certain dependencies among each other. These dependencies are lost once the queries are sent to the query processor in Hadoop as single jobs. Then once when these queries are again computed or processed as a job the internal job dependency arises. This can be reduced by maintaining an index or query dependency table which can not only reduce time but effect the overall computing in a good term. The processor could be utilized to process more and more data over the distributed nodes of Hadoop. The heterogeneous clusters used in the Hadoop can benefit by only in taking the relevant and discarding the irrelevant jobs to process.

Job scheduling mechanisms: The default scheduler of Hadoop and MapReduce is the FIFO Scheduler (First in First Out) which seems relevant as and when Hadoop job arrives. But in a dynamic environment of Big Data when the humongous size of data is to be processed, taking in everything that comes can become unfruitful and waste of resource, because in the end to profit from such big chunks of data, it has to be sorted at the last. Therefore scheduling the jobs that process such data so that the clusters with high-performance can deal with big data sets and clusters with slow-ends can help in other processing. The scheduling mechanism can help clear out the vision of processing data into meaningful loads to differently captive ends of Hadoop nodes cluster.

Iterative processing: The combination of TF-IDF and iterative mapping and reducing itself proves to be fruitful applicant of processing and finding out meaningful insights about data. The best feature where one doesn't have to invest in larger components to sort things out. The iterative processing can define the input of similarity matrix and hence simplify the processing and could complete jobs in relevantly shorter time spans.

REFERENCES

[1] Zhuo Liu, Efficient Storage Design and Query Scheduling for Improving Big Data Retrieval and Analytics, 2015

[2] Zongben Xu, Yong Shi, Exploring Big Data Analysis: Fundamental Scientific Problems, Springer, Dec. 2015

[3] Fernando G. Tinetti1, Ignacio Real, Rodrigo Jaramillo, and Damián Barry, Hadoop Scalability and Performance Testing in Heterogeneous Clusters, PDPTA 2015

[4] Krushnarah Kamtekar, Performance Modeling of Big Data, May 2015

[5] Fong-Hao Liu, Ya-Ruei Liou, Hsiang-Fu Lo, Ko- Chin Chang, and Wei-Tsong Lee, The omprehensive Performance Rating for Hadoop Clusters on Cloud Computing Platform, November 2014

[6] T.K.Das, P.Mohan Kumar, BIG Data Analytics: A Framework for Unstructured Data Analysis, IJET

[7] ISSN: 0975-4024, Vol 5 No 1 Feb-Mar 2013

[8] Florica Novăcescu, Big Data in High Performance Scientific Computing, 2013

[9] B. Thirumala Rao, N.V. Sridevi, V. Krishna Reddy, L.S.S. Reddy, Performance Issues of Heterogeneous Hadoop Clusters in Cloud Computing, GJSCT, Vol. XI Issue VII, May 2011

[10] Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares, and Xiao Qin, Improving MapReduce Performance