Task Scheduling Algorithm for Map Reduce to Control Load Balancing in Big Data

[1]Ms. N. Saranya M.E., [2]Mr. T. Yoganandh,
[1] PG Scholar, [2]Assistant Professor/CSE,
Jay Shriram Group of Institutions,
Tirupur

***Abstract*- Load balancing is biggest concern in today's world where this needs to be handled effectively. To improve the load balancing for distributed applications, in existing work XTrie and Etrie extended partitioning techniques are used. By improving load balancing, Map Reduce programs can become more efficient at handling tasks by reducing the overall computation time spent processing data on each node. However skews and stragglers problem cannot be resolved in existing work. This problem is overcome in this work by introducing the method called micro partitioning. In addition to that, Map Reduce Task Scheduling algorithm for Deadline constraints for the efficient scheduling is also proposed. For that node classification method is used which distribute the workload among the nodes according to the node capacity. After that a micro partitioning method is used for applications using different input samples. This approach is only effective in systems with high-throughput, low-latency task schedulers and efficient data materialization. The experimental results prove that the proposed methodology provides the better result than the existing methodology.**

Index Terms—Big Data, Map Reduce, Task scheduling, Micro-partitioning, Deadline Constraints.

I.INTRODUCTION

Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on- hand database management tools or traditional data processing applications. The challenges include capture, cu ration, storage, search, sharing, transfer, analysis, and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions. The limitations also affect Internet search, finance and business informatics. Data sets grow in size in part because they are increasingly being gathered by ubiquitous information-sensing mobile aerial sensor technologies (remote sensing), software logs, cameras, microphones, identification readers, and wireless sensor networks. Big data is difficult to work with using most relational database management systems and desktop statistics and visualization packages, requiring instead "massively parallel software running on tens, hundreds, or even thousands of servers". What is considered "big data" varies depending on the capabilities of the organization managing the set, and on the capabilities of the applications that are traditionally used to process and analyze the data set in its domain. "Forsome organizations, facing hundreds of gigabytes of data for the first time may trigger a need to reconsider data management options. For others, it may take tens or hundreds of terabytes before data size becomes a significant consideration.

A. Hadoop File System

Apache Hadoop is an open-source software framework for storage and large scale processing of datasets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users. It is licensed under the Apache License 2.0.The Apache Hadoop framework is composed of the following modules:

1. Hadoop Common - contains libraries and utilities needed by other Hadoop modules
2. Hadoop Distributed File System (HDFS) - a distributed file-system that stores data on the commodity machines, providing very high aggregate bandwidth across the cluster.
3. Hadoop YARN - a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications.
4. Hadoop MapReduce - a programming model for

large scale data processing.

All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines or racks of machines) are common and thus should be automatically handled in software by the framework. Apache Hadoop's MapReduce and HDFS components originally derived respectively from Google MapReduce and Google File System (GFS) papers.

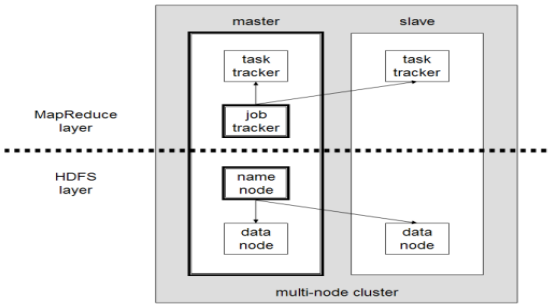

Fig.1 Architecture of Multi-node Hadoop Cluster.

B. Applications

The HDFS file system is not restricted to MapReduce jobs. It can be used for other applications, many of which are under development at Apache. The list includes the Hbase data base, the Apache Mahout machine learning system and the Apache Hive Data Warehouse System. Hadoop can in theory be used for any sort of work that is batch-oriented rather than real-time, that is very data- Intensive and able to work on pieces of the data in parallel. As of October 2009, commercial applications of Hadoop included:

1. Log and/or click stream analysis of various kinds.
2. Marketing analytics.
3. Machine learning and/or sophisticated data mining.
4. Image processing.
5. Processing of XML messages.
6. Web crawling and/or text processing.
7. General archiving, including of relational/tabular data, e.g. for compliance.

C. Background and preliminaries

MapReduce is a programming model developed as a way for programs to manage with large amounts of data. It achieves this goal by distributing the workload among multiple computers and then working on the data in parallel. From the programmers point of view compared to traditional methods MapReduce is a moderately easy way to create distributed applications. Programs that execute on a MapReduce framework need to divide the work into two phases:

1. Map
2. Reduce

Each phase has key-value pairs for both input and output.

To implement these phases, a programmer needs to specify two function:

1. A map function called a Mapper.
2. Corresponding reduce function called a Reducer.

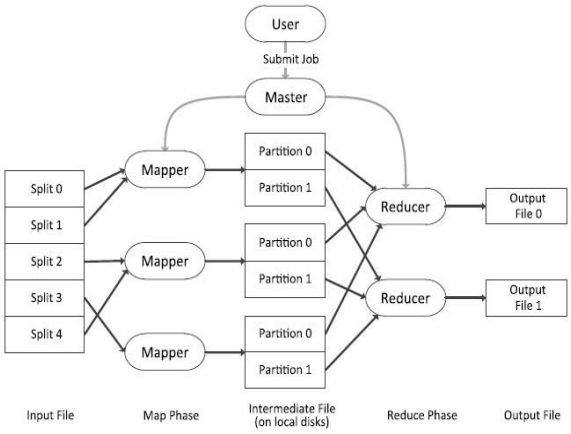

Fig 2: Dataflow Diagram of MapReduce.

When a MapReduce program is executed on Hadoop(open source software), it is expected to be run on multiple computers or nodes. Therefore, a master node is required to run all the required services needed to coordinate the communication between Mappers and Reducers. An input file is then split up into fixed sized pieces called input splits. These splits are then passed to the Mappers who then work in parallel to process the data contained within each split. As the Mappers process the data, they partition the output. Each Reducer then gathers the data partition designated for them by each Mapper, merges them, processes them, and produces the output file. It is the partitioning of the data that determines the workload for each reducer. In the MapReduce framework, the workload must be balanced in order for resources to be used efficiently .An imbalanced workload means that some reducers have more work to do than others. This means that there can be reducers standing idle while other reducers are still processing the workload designated to them.

II.OBJECTIVE

1. To distribute the workload among the nodes in the map reduce framework.
2. In the map reduce frame work the workload is distributed and working on the data in parallel manner.
3. To effectively distribute the workloads among the nodes and to reduce the storage space.

III. EXISTING SYSTEM

In the existing system, in order to reduce the time it takes to process the data, and the storage space to store the data, introduce an innovative approach called MapReduce programming model. The main contribution of this method is,

1. A method for improving the work load distribution amongst nodes in the MapReduce framework.
2. A method to reduce the required memory footprint.
3. Improved computation time for MapReduce when these methods are executed on small or medium sized cluster of computers.

A. Xtrie method

The Xtrie algorithm presented here provides a way to improve the cut point algorithm inherited from TeraSort. One of the problems with the TeraSort algorithm is that it uses the quicksort algorithm to handle cut points. By using quicksort, TeraSort needs to store all the keys it samples in memory and that reduces the possible sample size, which reduces the accuracy of the selected cut points and this affects load balancing. Another problem TeraSort has is that it only considers the first two characters of a string during partitioning. This also reduces the effectiveness of the TeraSort load balancing algorithm.In this method, there are two advantages over the quick sort algorithm. Firstly, the time complexity for insertion and search of the Trie algorithm is O(k) where *k* is the length of the key. Meanwhile, the quick sort algorithm best and average case is O(n log n) and in the worst case O( ) where *n* is the number of keys in its sample. Secondly, a Trie has a fixed memory footprint. This means the number of samples entered into the Trie can be extremely large if so desired. A Trie code is similar to a hash code, but the codes it produces occur in sequential ASCII alphabetical order. The equation for the Trie code is as follows:

Trie Code =X + X +……..+ X

1. User constraints such as deadlines are important requirements which are not considered.
2. Less accuracy in Scheduling.

IV. PROPOSED SYSTEM

In the proposed system, the MapReduce Task Scheduler for Deadline (MTSD) algorithm is used to meet the users' job deadline requirement in the cloud environment. The MTSD algorithm takes the data locality and cluster

$$= \sum X$$

Etrie method

heterogeneity into account. The data locality is the key factor that affects the efficiency of MapReduce jobs' running. The data locality means that the task's operation code and the task's input data are on the same computing One of the problems inherited by TeraSort and Xtrie is that they use an array to represent the trie. The main problem with this technique is that it tends to contain a lot of wasted space. For example, plain text files and log files often contain only rudimentary alphanumerical characters, whitespace, line breaks, and punctuation marks. Furthermore, when processing text, the whitespace, line breaks, and punctuation marks are often filtered out by the Mapper. Moreover, many kinds of applications only use a small range of keys, resulting in a lot of wasted space by the Trie. This work therefore presents the ReMap algorithm, which reduces the memory requirements of the original Trie by reducing the number of elements it considers. The algorithm does this by placing each ASCII character into one of the three categories: null, alphanumeric, or *other*. Alphanumeric characters are upper and lower case letters or numbers. Other is used for all characters that are not alphanumeric except for null. Null is used to represent strings that have fewer characters than the number of levels in the trie. The ReMap chart maps each of the 256 characters on an ASCII chart to the reduced set of elements expected by the Etrie. Since the purpose of Etrie is to reflect words found in English text ReMap reassigns the ASCII characters to the 64 elements. In the Etrie Partitioning method, the remap algorithm is used which reduces the memory requirements of the original trie by reducing the number of elements it considers. The algorithm does this by placing each ASCII character into one of the three categories: null, alphanumeric, or other. Alphanumeric characters are upper and lower case letters or numbers. Other is used for all characters that are not alphanumeric except for null. Null is used to represent strings that have fewer characters than the number of levels in the trie. The Re Map chart maps each of the 256 characters on an ASCII chart to the reduced set of elements expected by the Etrie. By reducing the number of elements to consider from 256 to 64 elements per level, the total memory required is reduced to 1/16th of its original footprint for a two-level trie.

Etrie Code = X + X + …….+ X

node or the same rack. Of course, the efficiency when the code and data are on the same node is higher than on the same rack. If the code and data are on the same node, it would avoid the data transmission in the network and greatly reduce the delay. Therefore, in the large scale data processing applications, shifting the code would be "cheaper" than moving data. In this paper, in order to meet the time constraints of the job and further improve the data locality, the MapReduce Task Scheduler for Deadline (MTSD) algorithm is proposed, which based on the computing power of meet time constraints in the heterogeneous environments. The contributions of this paper are as follows:

1. We introduce A node classification algorithm to improve the Map task's data locality.
2. We present a novel remaining task execution time model which is based on node classification algorithm.

A. Micro-Partitioning method

In addition to that we use micro-partitioning methods to divide the workload into smaller tasks. We introduce a new approach for avoiding skew and stragglers during the reduce phase. The key technique is to run a large number of reduce tasks, splitting the map output into many more

partitions than reduce machines in order to produce smaller tasks. These tasks are assigned to reduce machines in a "just-in-time" fashion as workers become idle, allowing the task scheduler to dynamically mitigate skew and stragglers. Running many small tasks lessens the impact of stragglers, since work that would have been scheduled on slow nodes when using coarser-grained tasks can now be performed by other idle workers. With large tasks, it can be more efficient to exclude slow nodes rather than assigning them any work. By assigning smaller units of work, jobs can derive benefit from slower nodes.

B.Task Scheduling Method based on Deadline constraints

In MapReduce, the job's execution progress includes $= \sum^{X}$.

C. Disadvantages

The disadvantages of existing system are Map and Reduce stage. So, the job's completion time contains Map execution time and Reduce execution time. In view of the differences between Map and Reduces code, we divide the scheduling progress into two stages, namely Map stage and Reduce stage. In the aspect of the
task's scheduling time prediction, the execution time of Map and Reduce is not correlative; their execution time depends on the input data and function of their own. Therefore, in this work the scheduling algorithm sets two deadlines: map-deadline and reduce-deadline. And reduce-deadline is just the users' job deadline. In order to get map-deadline, we need to know the Map task's time proportion on the task's execution time. In a cluster with limited resources, Map slot and Reduce slot number is decided. For an arbitrary submitted job with deadline constraints, the scheduler has to schedule reasonable with the remaining resources in order to assure that all jobs can be finished before the deadline constraints. According to map-deadline, we can acquire the current map task's slot number it needs; and with reduce- deadline, we can get the current reduce task's slot number it needs.

C. Advantages

The advantages of proposed system are

1. More Accuracy.
2. Reduce the delay.
3. Micro-partitioning is used.
4. This approach is only effective in systems with high-throughput, low-latency.

D. System Architecture

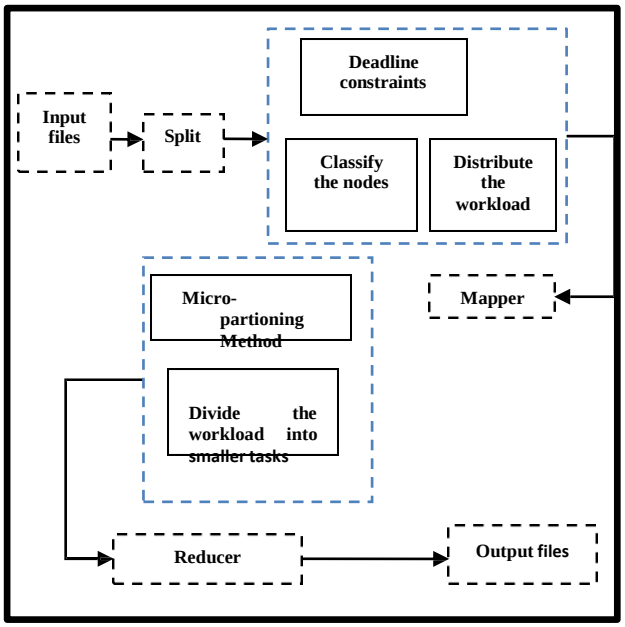

Fig 4.Arichitecture diagram of Micro-Partitioning

V. MODULE DESCRIPTION

For developing one hadoop platform, have to analyze the requirements for the resources and to allocate the job tasks. In this module, the resource assumption takes place which means need to mention the Terasorts used by map reduce, node classification method, tire and xtrie method and the resources configuration (Processor, RAM, Storage, Capacity, etc.). To analyze the task, the task from the user is analyzed and forward for the allocation process.

VI.SCREENSHOTS

A .To Get the Configuration Path

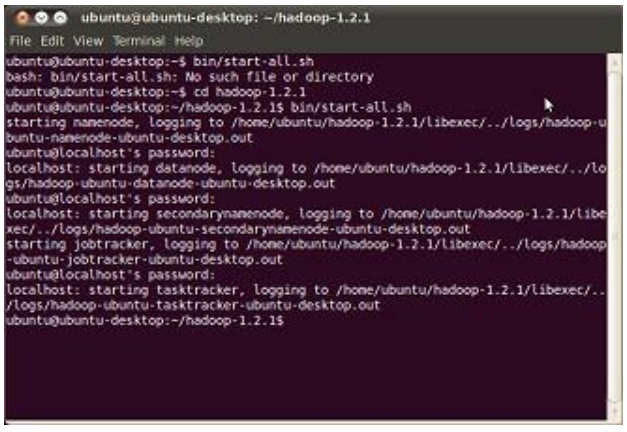

Fig 5.To Get the Configuration Path

B. Job Tracker allocated by Terasort in MapReduce

Fig.6 Job Tracker allocated by Terasort in Map reduce

C. Task Input

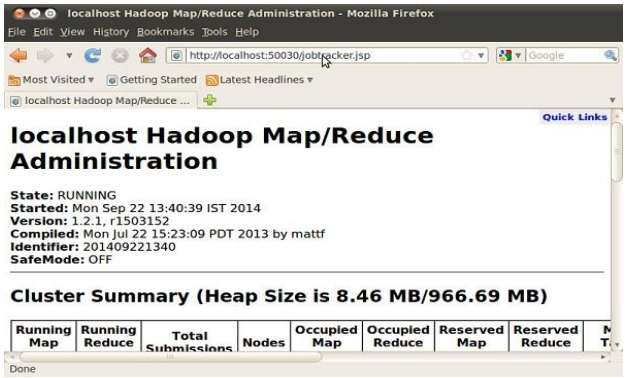

Fig.7 Task Input

D. Calculating the MapReduce

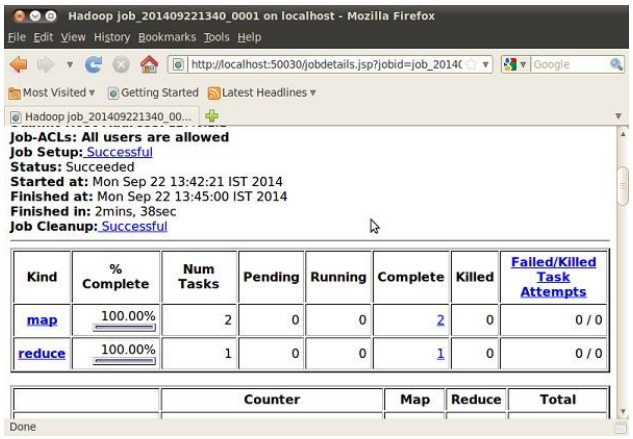

Fig.8 Calculating the MapReduce

E. Final Output of MapReduce

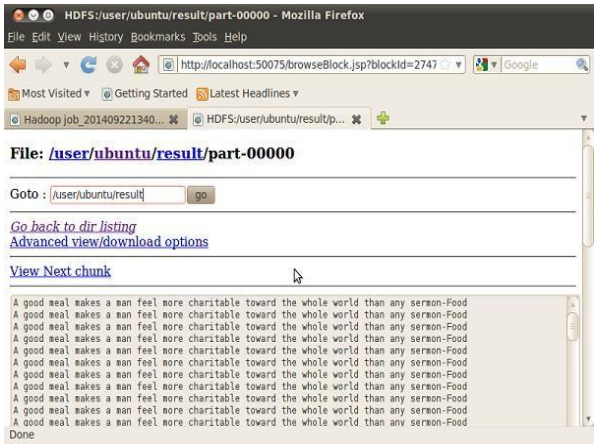

Fig 9. Final Output of MapReduce

VII. CONCLUSION AND FUTURE WORK

To improve the load balancing for distributed applications, we use XTrie and Etrie extended partitioning techniques. By improving load balancing, MapReduce programs can become more efficient at handling tasks by reducing the overall computation time spent processing data on each node. In that particular computing environment and for that data configuration, each partition generated by MapReduce appeared on only one or two nodes. In contrast, our study looks at small- to medium-sized clusters. This study modifies their design and optimizes it for a smaller environment. A series of experiments have shown that given a skewed data sample, the ETrie architecture was able to conserve more memory, was able to allocate more computing resources on average, and do so with less time complexity. In addition to that we use MapReduce Task Scheduling algorithm for Deadline constraints for the efficient scheduling. For that we use node classification method and distribute the workload among the nodes according to the node capacity. After that a micro partitioning method is used for applications using different input samples. This approach is only effective in systems with high- throughput, low-latency task schedulers and efficient data materialization. In the future, we would like to implement the proposed task scheduler architecture and perform additional experiments to measure performance using straggling or heterogeneous nodes. We also plan to investigate other benefits of micro-tasks, including the use of micro-tasks as an alternative to preemption when scheduling mixtures of batch and latency-sensitive jobs.

VIII. REFERENCES

1. Kenn Slagter, Ching-Hsien Hsu ,Yeh-Ching Chung ,Daqiang Zhang “An improved partitioning mechanism for optimizing massive data analysis using MapReduce” Springer Science, Business Media New York 2013.
2. Candan KS, Kim JW, Nagarkar P, Nagendra M, and Yu R (2010) RanKloud: scalable multimedia data processing in server clusters. IEEE MultiMed 18(1):64– 77

3. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrws M, Chandra T, Fikes A, Gruber RE (2006) Big table: a distributed storage system for structured data. In: 7th UENIX symposium on operating systems design and implementation, pp 205–218.
4. Dean J, Ghemawat Dean S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51:107–113.
5. Ghemawat S, Gobi off H, Leung S-T (2003) The Google file system. In: 19th ACM symposium on operating systems principles (SOSP).
6. Jiang W, Agrawal G (2011) Ex-MATE data intensive computing with large reduction objects and its application to graph mining. In: IEEE/ACM international symposium on cluster, cloud and grid computing, pp 475–484.
7. Jin C, Vecchiola C, Buyya R (2008) MRPGA: an extension of MapReduce for parallelizing genetic algorithms. In: IEEE fourth international conference on escience, pp 214–220.
8. Kavulya S, Tany J, Gandhi R, Narasimhan P (2010) an analysis of traces from a production MapReduce cluster. In: IEEE/ACM international conference on cluster, cloud and grid computing, pp 94–95.
9. Krishnan A (2005) Grid BLAST: a globus-based high- throughput implementation of BLAST in a grid computing framework. Concurr Comput 17(13):1607– 1623.
10. Liu H, Orban D (2011) Cloud MapReduce: a MapReduce implementation on top of a cloud operating system. In: IEEE/ACM international symposium on cluster, cloud and grid computing, pp 464–474.
11. Hsu C-H, Chen S-C (2012) Efficient selection strategies towards processor reordering techniques

AUTHORS BIOGRAPHY

N.SARANYA received her B.E degree in Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, India and currently pursuing M.E degree in Jay Shriram Group of Institutions, Tirupur, India. Her research interests include Big data, Cloud Computing, Advanced Database, Computer networks, Operating System .

Mr.T.YOGANANDTH received his B.E. degree in Coimbatore Institute of Technology, Coimbatore, India and M.E. degree in Hindustan, university, Chennai, India. Currently he is working as an Assistant Professor in Jay Shriram Group of Institutions, Tirupur, India. His research interests include Data mining, Cloud Computing and Big Data.