An efficient Mapreduce scheduling algorithm in hadoop

R.Thangaselvi[1], S.Ananthbabu[2], R.Aruna[3]

[1]M.E: Department of Computer Science, VV College of Engineering, Tirunelveli, India
[2]Assistant Professor, Department of Computer Science, VV College of Engineering, Tirunelveli, India
[3]M.E: Department of Computer Science, Jeyamatha College of Engineering, Kanyakumari, India

Abstract— *Hadoop is a free java based programming framework that supports the processing of large datasets in a distributed computing environment. Mapreduce technique is being used in hadoop for processing and generating large datasets with a parallel distributed algorithm on a cluster. A key benefit of mapreduce is that it automatically handles failures and hides the complexity of fault tolerance from the user. Hadoop uses FIFO (FIRST IN FIRST OUT) scheduling algorithm as default in which the jobs are executed in the order of their arrival. This method suits well for homogeneous cloud and results in poor performance on heterogeneous cloud. Later the LATE (Longest Approximate Time to End) algorithm has been developed which reduces the FIFO's response time by a factor of 2.It gives better performance in heterogenous environment. LATE algorithm is based on three principles i) prioritising tasks to speculate ii) selecting fast nodes to run on iii)capping speculative tasks to prevent thrashing. It takes action on appropriate slow tasks and it could not compute the remaining time for tasks correctly and can't find the real slow tasks. Finally a SAMR (Self Adaptive MapReduce) scheduling algorithm is being introduced which can find slow tasks dynamically by using the historical information recorded on each node to tune parameters. SAMR reduces the execution time by 25% when compared with FIFO and 14% when compared with LATE.*

Keywords— Hadoop, Mapreduce, Colud Computing, Scheduling, SAMR, Tuning.

I. INTRODUCTION

Hadoop is a software library framework that allows for the distributed processing of large datasets across clusters of computers using simple programming model [1]. Mapreduce is the data processing framework that automatically handles failures. It deals with the implementation for processing and generating large datasets with a parallel distributed algorithm on a cluster [2].Mapreduce is used in cloud computing because of hiding the complexity of fault tolerance from the programmer [3].Input data is splitted and fed to each node in the map phase. The results generated in this phase are shuffled and sorted then fed to the nodes in the reduce phase[4]. Hadoopdefaultly schedules the task using FIFO technique which is static [5]. Later several techniques are being developed which supports homogeneous tasks. LATE the dynamic scheduling technique is being introduced to schedule the jobs in heterogeneous environment[6]. Then the SAMR mapreduce scheduling technique is being developed which uses the historical information and find the slow nodes and launches backup tasks. The historical information is stored in each nodes in XML format. It adjusts time weight of each stage of map and reduce tasks according to the historical information respectively[7]. It decreases the execution time of mapreduce job and improve the overall mapreduce performance in the heterogeneous environment. In this paper we are tuning the parameters using k means clustering technique and then assigning tasks to each node thus improving the performance of hadoop in the heterogenous environment which is also known as Lloyd's algorithm[8]. In Hadoop 1, a single Namenode managed the entire namespace for a Hadoopcluster[9]. With HDFS federation, multiple Namenode servers manage namespaces and this allows for horizontal scaling, performance improvements, and multiple namespaces.YARN, the other major advance in Hadoop 2, brings significant performance improvements for some applications, supports additional processing models, and implements a more flexible execution engine. YARN is a resource manager that was created by separating the processing engine and resource management capabilities of MapReduce as it was implemented in Hadoop 1[18]. YARN is often called the operating system of Hadoop because it is responsible for managing and monitoring workloads, maintaining a multi-tenant environment, implementing security controls, and managing high availability features of Hadoop[10].

II. LITERATURE SURVEY

Hadoop defaultly uses FIFO technique in which the tasks are given priority in the order they arrived. This technique takes more response time for slower jobs when compared to faster jobs[5].Then in round robin technique each tasks are given equal priority[11]. In the fair scheduling technique all tasks get an average and equal share of resources over time[12].Then in capacity scheduling technique resources are allocated in a timely manner under constraints of allocated capacities[13].The

weighted Round Robin scheduling allocate weight to each queue then scheduling tasks of different sub queue according to weight[14]. Improved Weighted Round Robin scheduling uses weight update rules to reduce workload and to balance tasks allocation[15]. Hybrid scheduling is designed for data intensive workloads and tries to maintain data locality during execution[16]. SARS(Self-Adaptive Reduce Scheduling) can decide the start time points of each reduce tasks dynamically according to each job context, includes the job completion time[17].LATE (Longest Approximate Time to End) scheduling improves the execution in hadoop by finding real slow tasks[6]. SAMR improves the execution in hadoop by finding real slow tasks[7].

TABLE 1
LITERATURE SURVEY ON VARIOUS MAPREDUCE SCHEDULING TECHNIQUES

S.NO	ALGORITHM	ADVANTAGE	DISADVANTAGE
1	First In First Out(FIFO) scheduling	Reduces response time due to speculative execution. Works well in case of short jobs.	Uses fixed threshold for selecting tasks to reexecute. Can't identify which tasks to be reexecuted on fast nodes correctly.
2	Round Robin scheduling	No need to wait for the previous one to get completed. Job that can't be completed in its turn will be stored back to the queue waiting for the next turn.	Largest jobs take enough time for scheduling. Supports only internal scheduling of jobs.
3	Fair scheduling	Can work well in both small and large clusters.	Job weight is not considered for each node.
4	Capacity scheduling	Improve the utilization of resources through dynamic adjustment of resource allocation. Improve job efficiency.	User needs to know system information and make queue set and queue select group for the job.
5	Weighted Round Robin scheduling	Can provide good fairness when the size of each task is same.	Provides unfairness for the smaller queues if the size of the task is inconsistent. Due to fixed weight, can't adjust the weight of each sub queue in real time.
6	Improved Weighted Round Robin scheduling	Easy to implement. Low cost.	Defects occur when consider the external influence on the time that each task took when they switched scheduling. Does not maintain stability under high concurrency, large capacity and high workload.
7	Hybrid scheduling	Fastand flexible scheduler. Improves response time for multiuser hadoop environment.	The time taken for the creation of tasks and result retrieval is increased due to the increase in the number of tasks.
8	Self-adaptive Reduce scheduling(SARS)	Reduces completion time. Decrease the response time	Only focuses on reduce process.
9	Longest approximate time to end(LATE) scheduling	Robustness to heterogeneity. Address the problem of how to robustly perform speculative execution to maximize performance.	Does not compute remaining time for tasks correctly and can't find real slow tasks.Poor performance due to the static manner in computing the progress of the tasks.
10	Self-adaptive mapreduce(SAMR) scheduling	Decreases the execution time of mapreducejob. Improve the overall mapreduce performance in the heterogeneous environment.	Does not find the slow jobs accurately.

III. THEORETICAL FOUNDATION

The MapReduce framework first splits an input data file into G pieces of fixed size, typically being 16 megabytes to 64 megabytes (MB). These G pieces are then passed on to the participating machines in the cluster. Usually, 3 copies of each piece are generated for fault tolerance. It then starts up the user program on the nodes of the cluster. One of the nodes in the cluster is special the master. The rest are workers that are assigned work by the master. There are M map tasks and R reduces tasks to assign. M and R is either decided by the configuration specified by the user program, or by the cluster wide default configuration. The master picks idle workers and assigns them map tasks. Once map tasks have generated intermediate outputs, the master then assigns reduces tasks to idle workers. Note that all map tasks have to finish before any reduce task can begin. This is because a reduce task needs to take output from every map task of the job. A worker who is assigned a map task reads the content of the corresponding input split. It parses key/value pairs out of the input data chunk and passeseach pair to an instance of the user defined map function. The intermediate key/value pairs produced by the map function are buffered in memory at the corresponding machines that are executing them. The buffered pairs are periodically written to a local disk and partitioned into R regions by the partitioning function. The framework provides a default partitioning function but the user is allowed to override this function by a custom partitioning. The locations of these buffered pairs on the local disk are passed back to the master. The master then forwards these locations to the reduce workers. When a reduce worker is notified by the master about these locations, it uses remote procedure calls to read the buffered data from the local disks of mapworkers. When a reduce worker has read all intermediate data, it sorts it by the intermediate key so that all occurrences of the same key are grouped together.

The sorting is needed because typically many different keys are handled by a reduce task. If the amount of intermediate data is too large to fit in memory, an external sort is used. Once again, the user is allowed to override the default sorting and grouping behaviors of the framework. Next, the reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the reduce function. The output of the reduce function is appended to a final output file for this reduce partition. When all map tasks and reduce tasks have completed, the master wakes up the user program. At this point, the MapReduce call in the user program returns back to the user code.

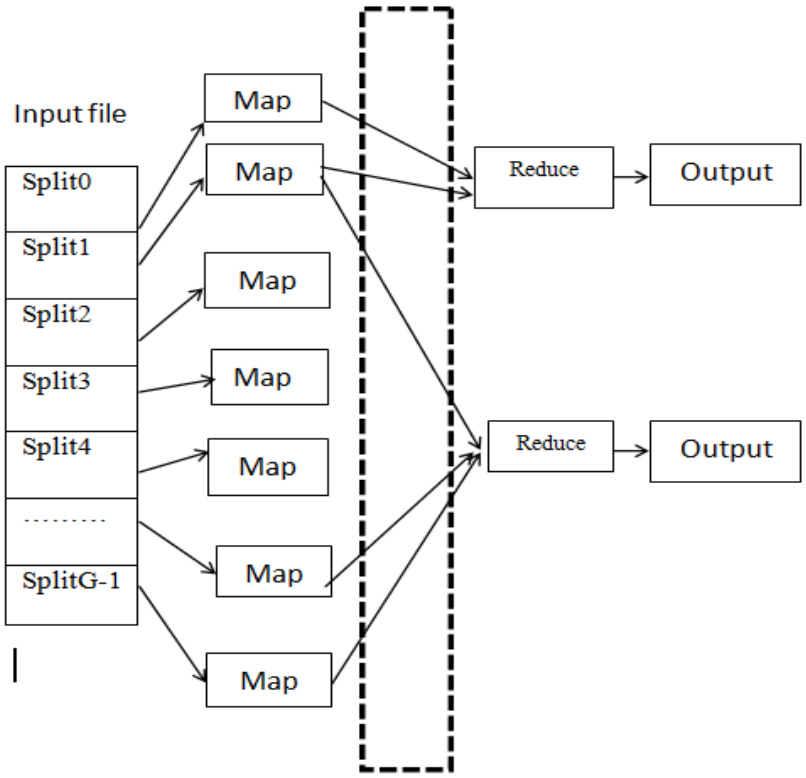

FIG 1: MAP REDUCE FRAMEWORK

IV. METHODOLOGY

4.1 Proposed methodology

The SAMR technique uses the historical information that is being stored in each node and using that information it finds the real slow tasks. Then it maps the slow tasks and reduces the slow tasks. In this paper we use the k-means clustering technique to tune the parameters in the historical information and finding the slow tasks very accurately. The proposed K-means algorithm can solve even the most difficult clustering issues. It requires the number of clusters that we are going to use in our process. The algorithm finds k centroids, one for each cluster. Depending on the location of the centroid the result will vary. During the map phase it finds the M1 temporary value and using this value it finds in the clusters which one is closest to the M1 value. Similarly inthe reduce phase it finds the R1 temporary value and using this value it finds in the clusters which one is closest to the R1 value. Based on the result the centroid location is changed and the values are recalculated again.

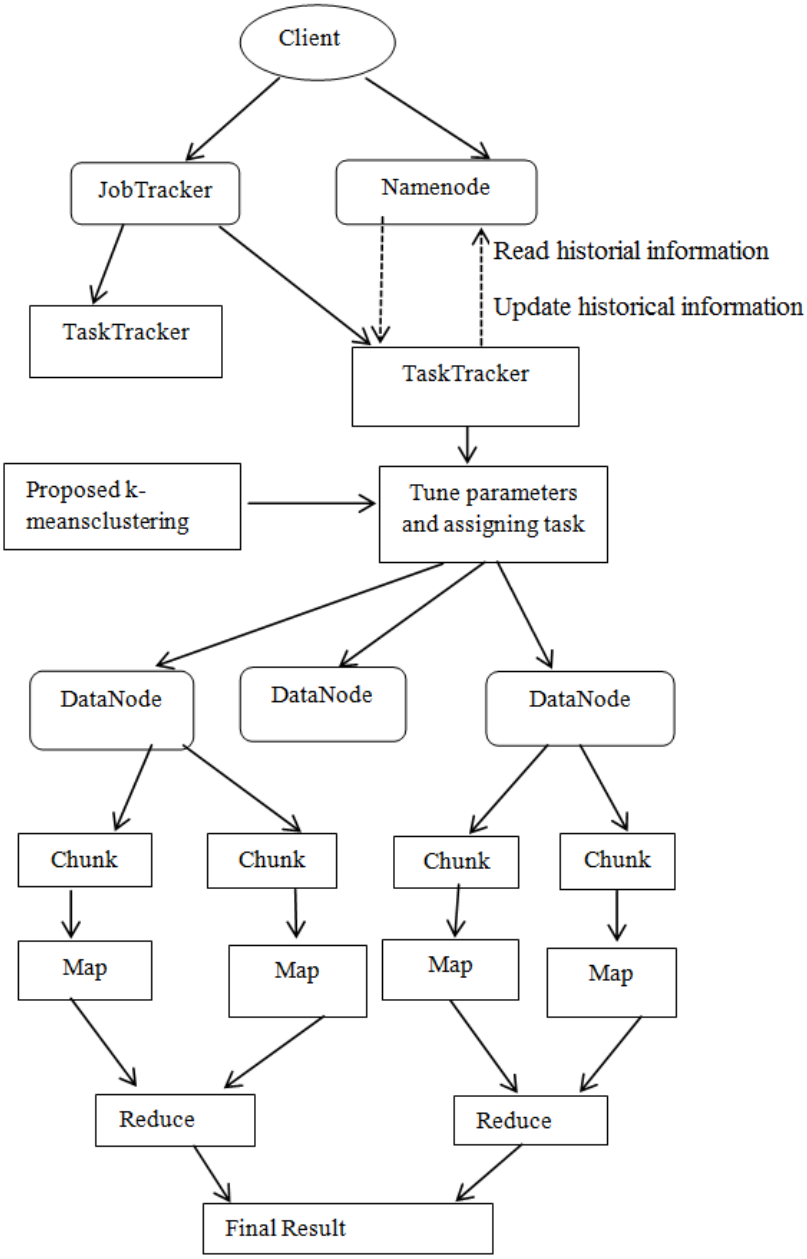

FIG 2: MAPREDUCE IMPLEMENTATION

4.2 Algorithm

Algorithm 1 Self-adaptive MapReduce

1: Start procedure

2: input: Key/Value pairs

3: output: Statistical results

4: read historical information

5: tune parameters using proposed k-means clustering

6: Find slow tasks

7. Find slow tasktrackers

8: Launch back up tasks

9: Using the results update the historical information

10. End procedure

Algorithm 2 Proposed k-means clustering

1: Start procedure

2: Input: D-set of n datanodes, n-number of datanodes, C-set of k centroids, k-number of clusters

3: Output: A-set of k clusters

4: Compute distance between each data nodes to all centroids

5: For each Di find the closest Ci

6: Add Di to A

7: Remove Di from D

8: Repeat for all Di……Dn andCi...Ck

9: End procedure

V. RESULTS AND DISCUSSION

We applied the proposed k-means clustering algorithm to improve the performance of the Self-adaptive MapReduce scheduling algorithm. The proposed k-means clustering algorithm works better than the k-means clustering algorithm. When the input file is given the job tracker manages the information and assigns tasks to its slave nodes also known as task trackers. The Namenode holds the metadata which is the master of all datanodes. The jobtracker and tasktrackers

communicate among themselves and the datanodes will communicate with the namenode.The proposed k-means clustering algorithm find the closest distance between each datanodes and each centroids . Using this result it update the historical information in the name node and find the accurate slow tasks , launch backup tasks and assign tasks to each task trackers. The proposed k-means clustering technique takes less amount of computation time than the basic k-means clustering algorithm. The following table shows the expected experimental results for the basic k-means clustering and the proposed k-means clustering algorithm.

TABLE 2
EXPECTED EXPERIMENTAL RESULTS

No of files	No of clusters	Algorithm	Computation time in seconds
100	3	k-mean	0.105
		Proposed k-mean	0.085
200	3	k-mean	0.130
		Proposed k-mean	0.100
300	4	k-mean	0.145
		Proposed k-mean	0.125
400	2	k-mean	0.165
		Proposed k-mean	0.145
500	2	k-mean	0.205
		Proposed k-mean	0.185

VI. CONCLUSION

In this paper we proposed a method to improve the efficiency of the map reduce scheduling algorithms. It works better than existing map reduce scheduling algorithms by taking less amount of computation and gives high accuracy. We used the proposed k-means clustering algorithm together with the Self-Adaptive MapReduce(SAMR) algorithm. However this technique works well it can assign only one task to each data node. In the future we have decided to improve its efficiency by allocating more number of tasks to the datanodes.

REFERENCES

[1] Hadoop, "Hadoop home page" http://hadoop.apache.org/ .

[2] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified data processing in large clusters", in OSDI 2004: Proceedings of 6th symposium on operating system design and implementation, (New York), pp. 137-150, ACM Press, 2004.

[3] JineshVaria,"Cloud architectures", White paper of Amazon, jineshvaria.s3.amazonaws.com/ public/ cloud-architecturesvaria.pdf,2008.

[4] Yaxiong Zhao*, Jie Wu, and Cong Liu , "Dache: A data aware caching for bigdata applications using the MapReduce framework" , in TSINGHUA science and technology, ISSN 1007-0214 05/10 pp. 39-50 Volume 19, Number 1.

[5] JaideepDhok and VasudevaVarma , "Using pattern classification for task assignment in mapreduce"

[6] MateiZaharia, Andy Konwinski, Anthony D.Joseph, Randy Katz, Ion Stoica , "Improving MapReduce performance in heterogeneous environment" ,8th USENIX symposium on operating systems design and implementation.

[7] Quan Chen, Daqiang Zhang, MinyiGuo, Qianni Deng and Song Guo, "SAMR: A self-adaptive MapReduce scheduling algorithm in heterogeneous environment.

[8] Andrew Moore: "K-means and hierarchical clustering Tutorial Slides" htttp://www-2.cs.cmu.edu/~awm/ tutorials/kmeans.html.

[9] Dan Sullivan ,"Getting started with Hadoop 2.0" , http://www.tomsitpro.com/articles/hadoop-2-vs-1,2-718.html.

[10] ThilinaGunarathne: " HadoopMapReduce v2 Cookbook" , second edition PACKT publishing, Birmingham-Mumbai.

[11] Gothali E and Balasubramanie P ,"A novel approach for partitioning in hadoop using Round Robin technique", ISSN: 1992-8645 JATIT , Vol. 63 No. 2.

[12] "Hadoop Fair Scheduler Design Document" , http://svn.apache.org/hadoop/fair_scheduler_design_doc.pdf .

[13] "Capacity Scheduler Guide" http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html .

[14] HaiyangLi , "PWBRR algorithm of hadoop platform".

[15] Yu Liping, Yang Lishen and Zhu Liang , "Multiuser scheduling strategy research based on mapreduce cluster"

[16] AysanRasooli and Douglas G. Down , "A Hybrid Scheduling approach for scalable heterogeneous hadoop systems.

[17] ZhuoTang ,Lingang Jiang , Junqing Zhou , Kenli Li and Keqin Li " A self-adaptive scheduling algorithm for reduce start time" .

[18] ArunC.Murthy, Vinod Kumar Vavilapalli, Dough Eadline, Joseph Niemiec and Jeff Markham, "Apache Hadoop Yarn: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2.

[19] MapReduce, http://www.mongodb.org/display/ DOCS/ MapReduce

[20] Junjie Wu "Advances in k-means clustering" , a data mining thinking

[21] K.Krishna and M.NarasimhaMurty , "Genetic k-means algorithm" IEEE Transactions, Vol. 29, No. 3, June,1999

[22] H. Spath, "Clustering Analysis Algorithms" New York: Van Nostrand Reinhold, 1991.

[23] C. Tian, H. Zhou, Y. He, and L. Zha, "A dynamic MapReduce Scheduler for heterogeneous workloads," in Proceedings of the 2009 eighth International Conference on Grid and Cooperative Computing-Volume 00, pp. 218-224, IEEE Computer Society,2009.

[24] L. Barroso, J.Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture", IEEE Micro, vol. 23, no. 2, pp. 22-28, 2003

[25] P. Nguyen, T. Simon, M. Halem and D. Chapman, "A hybrid algorithm for data intensive workloads in a mapreduce environment" , 2012 IEEE/ACM fifth International Conference on Utility and Cloud Computing

[26] H. Ozkose, E. Sertacand C. Gencer ,"Yesterday, Today and Tomorrow of Big Data" .