RESEARCH ARTICLE OPEN ACCESS

Review of Map Reduce Scheduling Algorithms

Anusha Itnal[1], Sujata Umarani[2]
[1,2](Computer Science, GIT, VTU, Belagavi)

----------**********----------

Abstract:

Big data is a technology that refers to massive, intensive data that is changing rapidly. Such diverse and huge volume data is very difficult to handle by present technologies and infrastructures. Hadoop overcomes this drawback with powerful programming model called MapReduce. The implementation of Map Reduce needs storage and parallel processing. HDFS is commonly used for storing and processing of the data. In this paper we suggest various scheduling algorithms for MapReduce that help increase the resource utility and speedup the response time of the system. Each of the algorithms are compared, nature and drawbacks of algorithms are mentioned.

Keywords — **MapReduce, Hadoop, BigData.**

----------**********----------

I. INTRODUCTION

In this age of digital data, speed with which data is generated is commendable. In 2005 digital data on global scale was 150 ExaBytes, which has grown to be 1200 ExaBytes in 2010. This growth is expected to increase by 44% in 2020 [1]. This data is not necessarily in structured format. It contains mixed data example tweets, status updates, blogs, videos etc; they're unstructured. Latest and advanced technologies like Hadoop are necessary to process such massive data. Hadoop is not just massive data storage engine, it is more than that. Data processing and Data storage can be combined. It is robust , scalable, reliable and inexpensive [2]. Hadoop uses HDFS for storing the data and Map Reduce to allocate the tasks.

HDFS: Hadoop Distributed File System is fashioned to store a large amount of data upto tera and peta bytes. HDFS also provides very easy access to all the data stored in it. HDFS has interface similar to unix file system [3]. the meta data and application data are stored separately to increase the performance. HDFS is a master/slave architecture where namespace contains files and directories, these are presented by NameNode where permissions, access time records, space etc noted. the file content is stored in blocks (128 mega bytes). these blocks are replicated in DataNodes, which handle read and write requests received from the clients. Along with Read and Write it is responsible for block creation and deletion but only if the instruction is received from NameNode. NameNode stores the mapping of blocks to DataNodes.

HDFS is master/slave architecture where NameNode acts like master where each cluster has single master and DataNodes are slaves.

Mapreduce is a framework for processing large amount of data. Apache adopted MapReduce from Google. It is a neat programming model which processes large amount of data in parallel. MapReduce is also master/slave architecture [19]. MapReduce technique has map function that generates intermediate key/value pair by processing a key/value pair and reduce function takes all intermediate values with same key and concatenates them.

Map function: The main node i.e master node is given the input and the input is divided into several smaller parts. All smaller sub-problems are distributed to slave nodes. Slave node sends back the result after processing the data to master node.

ReduceFunction: Map function sends this data to Reduce function. The master node in this collects

all the intermediate files and combines and processes it in a way which results in answer to original problem.
Mapreduce is run on multiple nodes when executed on Hadoop.

Step by step guide:

1. The data is divided into M parts (16Mb to 64 Mb). Many copies of the data are started in the cluster.
2. One copy is master and others are slave. Master assigns work for slaves. M map tasks and R reduce tasks should be assigned. This is done by picking up idle workers and given either map or reduce tasks.
3. Worker assigned a map task reads the copy, processes it and passes the key/value pair to master node. All the key/value pairs generated by slave nodes are taken and buffered in a memory as intermediate key/value output.
4. The buffered output is distributed to regions in local disk and location of regions are sent back to master. The master forwards them to reduce workers.
5. RPCs are used to fetch the data from local disk. Reduce workers sort all the intermediate data. This is done because there might be multiple copies of the same key.
6. The sorted intermediate data is taken and values corresponding to each key is forwarded to reduce function. Reduce function forwards the final output file.
7. When Map and Reduce tasks are done the masters wakes up and call is returned to client program.

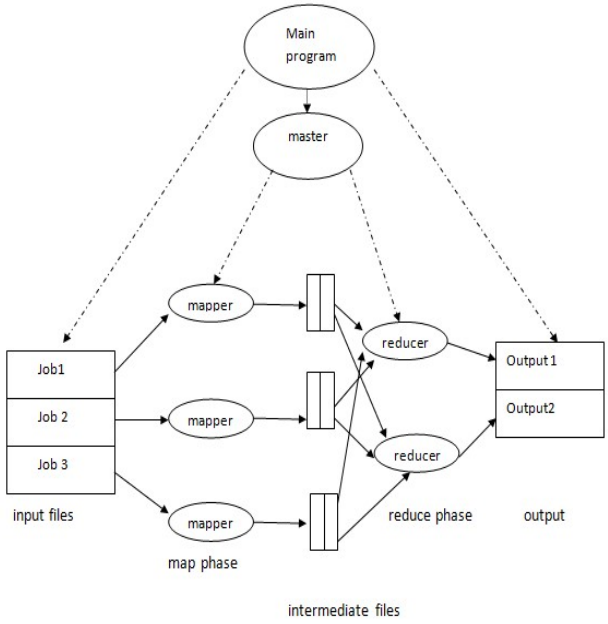

Fig. 1 Map Reduce Job

II. SCHEDULING ALGORITHMS

MapReduce programming model processes the tasks in parallel fashion therefore scheduling is highly necessary. If scheduling is inappropriate the whole system could break down and performance of entire system is degraded. Depending upon the type of the task, nature of the environment etc factors the scheduling algorithm is decided. The main goal of any algorithm chosen is to minimize the time taken to complete the job. These scheduling algorithm chosen must cater such that performance is increased. MapReduce tasks are categorized as NP-hard.
If the algorithm focuses on priority it may have to compromise data locality and so on. This will have affect on jobs' performance. Different jobs may have different priorities and before selecting an algorithm all these factors must be considered and analysed.

MapReduce has two types of run time behavior – adaptive and non adaptive. Adaptive algorithms are those which use parameter values and make decisions . Non-Adaptive algorithms are those which do not take any values into consideration before making scheduling decisions and jobs are processed in the order predefined.

A. FIFO

FIFO is default scheduler in Hadoop. The work of FIFO scheduler is to align the jobs based on their arrival; first they arrive, first they are served [4]. HDFS sends heartbeat each time there is empty slot in map or reduce. FIFO checks for the job that had arrived prior and has high waiting time. That job is selected if it is suitable and is data local, otherwise job next in line is selected. This goes on until matching job is found. Drawback for FIFO is it gives better performance when all the jobs to be scheduled are similar, if the jobs vary in their nature FIFO doesn't discriminate between short and longer jobs are thus performance is affected.

B. Fair scheduling

As the name suggests, the scheduling is fair to all the jobs. All the resources are shared among all the jobs fairly [5]. Each job gets resources for equal amount of time. This scheduler was first used by Facebook to handle the access in Hadoop [6]. Unlike FIFO this scheduler doesn't let jobs starve, each job is given equal time and resource enabling smaller jobs to finish in reasonable time. There are pools where jobs are assigned and resources are equally distributed among them. Each user is assigned their own pool by default. There can be limit set to number of jobs each pool can handle or number of jobs a user can run at a time. This is advantageous because it limits the number of map or reduce tasks running which avoids overloading of jobs.

C. Capacity scheduler:

Each organization have their own set of resources, enough to meet the organization needs. When each organization has cluster it leads to under utilization and also leads to overloading of jobs. Sharing cluster between organizations is beneficial because it allows to scale across both organizations which makes it cost effective and managing the cluster becomes easier than having private cluster per organization. This increases the utilization of resources. Pools are replaced by queues. Resources are divided among queues and one queue is given to each organization [7]. Measures are taken such that organization can access only its queue. This guarantees security. Like fair scheduling algorithm, each queue has job and/or task limit minimizing overloading and maximizing resource utilization. [6] [8]. If a queue has free resources, it gives it to other queues which have overloaded jobs/tasks. After the job is completed the resources are returned back to lender queue. Jobs can be assigned some priority level and executed based on the priority [9].

D. Hybrid Scheduler

Hybrid scheduler is based on priority of jobs. The priority is assigned dynamically [10]. This helps reduce the delay and also helps maintain the data locality. It is observed that response time over Hadoop clusters was 2 times faster. This is achieved by relaxing the tight fairness by exponential policy model, Advantages of this is it's fast and flexible.

E. Delay Scheduling

The main focus of any algorithm is to provide resources to all the clusters and to not have negative impact on the fairness. One way to do that is kill the old tasks provide space for new ones or wait until the tasks are finished and allot the space to new jobs. When the task is killed , all the work done by that task will be wasted. New scheduling called delay scheduling is introduced which relaxes the fairness temporarily. When any node requests a task and the next job in the queue fails to launch the task that is local, the job is skipped and another job that can launch the task is selected. If the skipped job is skipped for longer time, to avoid starvation it is chosen to launch non-local task [11].

F. LATE

Longest Approximate Time to End [12] was proposed by Zaharia et al. This scheduler focused on increasing the robustness and reducing the overhead of tasks on cluster. LATE scheduler does this by finding the slow tasks in Hadoop clusters by calculating remaining time of all other tasks. The remaining time is used to assign ranks for tasks and the starts the task that has progress rate lower than average process rate [13] [12]. Three principles of LATE are prioritizing tasks, choosing fast nodes,

capping tasks. This algorithm works well with synchronization of map and reduce phases.

G. SAMR

Self-Adaptive MapReduce is better than LATE scheduling algorithm. LATE cannot find slow tasks efficiently therefore Chen et al [14] proposed SAMR for heterogeneous environment. It identifies fast nodes and slow nodes by observing all the nodes and their computation speeds. MapReduce is parallel environment, one slow node can degrade the performance of entire system but in heterogeneous cluster nodes do not have same capacity, power, memory etc and therefore do not behave in same way. Three factors that make SAMR better are : it finds data locality for the nodes. Data locality can increase the computation capacity. It is evaluated in multiple platforms after it is evaluated on rented Cloud computing platform.

H. Maestro

Maestro is developed by Ibrahim et al. This technique avoids the problem of non-local tasks that depend on replica of map tasks. This is overcome by keeping track of all replica locations addressed by each node. This will help maestro to launch the map tasks without affecting other nodes' tasks. Depending on number of map tasks that are hosted and on their input data replication scheme it fills the empty slots of the node. Probability of queuing a map task based on replica of the tasks on the given machine, runtime scheduling is calculated. These two main waves that maestro takes into account will help execution of map tasks in local data location. And helps distribute intermediate data evenly. In experiment conducted by [15]. A cluster of 100 nodes, Maestro achieves local map executions to up to 95%, reducing speculative map tasks by 80%. Thus resulting in improved execution time by 34%.

I. CREST

Combination Re-Execution Scheduling Technology was proposed by Lei et al [16] helps in increased running time of map tasks. It is estimated that CREST reduces the running time of map tasks (speculated) by 70% and worst case 50%. The idea behind this is executing repeatedly the combination of tasks on couple of nodes may result in faster progress compared to subjecting a tasks directly on target node.

J. COSSH

One algorithm that considers heterogeneity at both application level and cluster level is COSSH. Proposed by Rasooli and Down [17]. This scheduler is specially designed for Hadoop. COSSH uses system information and takes scheduling decisions which increases the performance of overall system. Two basic steps in COSSH are, allotting the job in appropriate queue when the new job arrives and assign job to free resource upon receiving a heartbeat.

K. SARS

Tang et al [18] proposed Self-Adaptive Reduce Scheduling. SARS algorithm analyses the map and reduce phases in detail and it is able to determine dynamically the start time of the reduce task depending on the size of the output of map tasks. SARS increases the resource utilization rate by minimizing reduce tasks waiting time. First reduce tasks read the each map output from data blocks. This is copy phase. Second these outputs from map phase are ordered and merged. This data is divided into two types, one is data in memory and other is data in circular buffer. The memory that is assigned for reduce tasks is limited, which is why the buffer data must be written to disk frequently. In external sorting, the new data must be merged with data on the disks regularly. If there are large number of map tasks or large sized map tasks, external sorting must be done multiple times. Copy phase and sort phase together is called shuffle phase.

III. CONCLUSION

In Big Data, the amount of data stored and processed is huge. Hadoop technology is used to handle this data. It requires mechanism that can speed up the process and improve the performance and reduce the computation time. Different map reduce algorithms are mentioned in the paper, each

with advantages and drawbacks. Adaptive algorithm are ones that take future or current parameter into account for making decision and non adaptive algorithms in contrast doesn't make any environment parameters into account and schedules the tasks based on previously defined values. All these algorithms are suitable for different jobs/tasks , different environment. All these algorithms speed up Hadoop's data retrieval and processing, and also improve the performance and resource utilization. The comparison of each algorithm discussed in the paper is shown in the table 1.

TABLE I
COMPARISONO OF ALGORITHMS

Scheduling Algorithm	Taxonomy	Idea to Implementation	Advantages	Dis-advantages
FIFO	Non-adaptive	schedule jobs based on their priorities in first-come first-out.	1. cost of entire cluster scheduling process is less. 2. simple to implement and efficient.	1. designed only for single type of job. 2. Low performance when run multiple types of jobs. 3. poor response times for short jobs compared to large jobs.
Fair Scheduling	adaptive	do a equal distribution of compute resources among the users/jobs in the system.	1. less complex. 2. works well when both small and large clusters. 3. it can provide fast response times for small jobs mixed with larger jobs.	1. does not consider the job weight of each node.
Capacity	adaptive	Maximization the resource utilization and throughput in multi-tenant cluster environment.	1. ensure guaranteed access with the potential to reuse unused capacity and prioritize jobs within queues over large cluster .	1. The most complex among three schedulers.
hybrid scheduler based on dynamic priority	adaptive	designed for data intensive workloads and tries to maintain data locality during job execution	1. is a fast and flexible scheduler. 2. improves response time for multi-user Hadoop environments.	-
LATE	adaptive	Fault Tolerance	1. robustness to node heterogeneity. 2. address the problem of how to robustly perform speculative execution to maximize performance.	1. only takes action on appropriate slow tasks. 2. However it does not compute the remaining time for tasks correctly, and cannot find real slow tasks in the end. 3. poor performance due to the static manner in computing the progress of the tasks.
SAMR	adaptive	To improve MapReduce in terms of saving the time of the execution and the system's resources.	1. decreases the execution time of map reduce job. 2. improve the overall MapReduce performance in the heterogeneous environments.	1. do not consider the data locality management for launching backup tasks.
delay scheduling	adaptive	To address the conflict between locality and fairness.	1. Simplicity of scheduling	1. No particular
Maestro	adaptive	Proposed for map tasks, to improve the overall performance of the MapReduce computation.	1. provide a higher locality in the execution of map tasks. 2. provide more balanced intermediate data distribution for the shuffling phase.	-
CREST	adaptive	re-executing a combination of tasks on a group of computing nodes.	1. decrease the response time of MapReduce jobs. 2. optimal running time for speculative map tasks.	-
COSHH	adaptive	proposed to improve the mean completion time of jobs	1. improve the overall system performance. 2. addresses the fairness and the minimum share requirements.	-
SARS	adaptive	shorten the copy duration of the reduce process, decrease the task complete time, and save the reduce slots resources	1. reduce completion time. 2. decreased the response time.	1. only focus on reduce process.

REFERENCES

[1] K. Michael and K. W. Miller, "Big Data: New opportunities and new challenges [guest editors' introduction]," in *Comput*, vol. 46, Jun. 2013, pp. 22-24.

[2] Albert Bifet, "Mining Big Data in Real Time," *Informatica*, vol. 37, pp. 15-20, 2013.

[3] Soumya C L, Rashmi K, Rahul M Dr. Siddaraju, "Efficient Analysis of Big Data Using Map Reduce Framework," *IJRDET*, vol. 2, no. 6, June 2014.

[4] Devendra P. Gadekar Harshawardhan S. Bhosle, "Big Data Processing Using Hadoop: Survey on Scheduling," *IJRS*, vol. 3, no. 10, 2014.

[5] Andy Konwinski, Anthony D. Joseph, Randy Katz, Ion Stoica MateiZaharia, "Improving MapReduce performance in heterogeneous environment," in *In proceedings of 8th USINEX conference on Operating systems design and implementation*, december-2008, pp. 29-42.

[6] Hadoop homepage. [Online]. http://hadoop.apache.org/

[7] B. Thirumala Rao, LSS Reddy V. Krishna Reddy, "Research issues in Cloud Computing ," *Global Journal Computer Science & Technology*, vol. 11, no. 11, pp. 70-76, June 2011.

[8] D. Wang and W. Zhao J. Chen, "A Task Scheduling Algorithm for Hadoop Platform," *JOURNAL OF COMPUTERS*, vol. 8, no. 4, pp. 929-936, April 2013.

[9] N. Tiwari, "Scheduling and Energy Efficiency Improvement Techniques for Hadoop Mapreduce: State of Art and Directions for Future Research (Doctoral dissertation, Indian Institute of Technology, Bombay Mumbai)".

[10] T. Simon, M. Halem, D. Chapman, Q. Le P. Nguyen, "A hybrid scheduling algorithm for data intensive workloads in a MapReduce environment," in *In Proceedings of the 2012 IEEE/ACM fifth international conference on utility and cloud computing,* Washington DC, USA, 2012, pp. 161-168.

`[11] Berkeley, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenkar, Ion Stacia Matei Zaharia University of California, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling ,".

[12] A. Konwinski, A.D. Joseph, R. Katz and I. Stoica M. Zaharia, "Improving MapReduce performance in heterogeneous environments ," in *OSDI 2008: 8th USENIX Symposium on Operating Systems Design and Implementation 2008*.

[13] B.P Andrews and A. Binu, "Survey on Job Schedulers in Hadoop Cluster," *IOSR Journal of Computer Engineering*, vol. 15, no. 1, pp. 46-50, Sept-Oct 2013.

[14] D. Zhang, M. Guo, Q. Deng Q and S. Guo Q. Chen, "SAMR: a self-adaptive MapReduce scheduling algorithm in heterogeneous environment," in *The 10th international conference on computer and information technology. IEEE*, 2010, pp. 2736-43.

[15] H. Jin, L. Lu, B. He, G. Antoniu and S. Wu S. Ibrahim, "Maestro: replica-aware map scheduling for MapReduce ," in *The 12th international symposium on cluster, cloud and grid computing. IEEE/ACM*, 2012, pp. 435-477.

[16] T. Wo and C. Hu L. Lei, "CREST: Towards fast speculation of straggler tasks in MapReduce," in *The eighth international conference on e-business engineering. IEEE*, 2011, pp. 311-316.

[17] A. Rasooli and D.G. Down, "COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems," *Future Generation Computer Systems*, vol. 36, pp. 1-15, 2014.

[18] L. Jiang, J. Zhou, K. Li, and K. Li Z. Tang, "A self-adaptive scheduling algorithm for reduce start time.," *Future Generation Computer Systems, 2014*.

[19] Syama R Liya Thomas, "Survey on MapReduce Scheduling Algorithms," *IJCA*, vol. 95, no. 23, june 2014.