Analysis of Task Scheduling in Hadoop MapReduce Framework

Kamalakant Bawankule[1]
Computer Science and Engineering,
MNNIT,Allahabad

Anil Kumar Singh[2]
Computer Science and Engineering,
MNNIT,Allahabad

Rupesh Kumar Dewaang[3]
Computer Science and Engineering,
MNNIT,Allahabad

Abstract— **There are many open source platform available for storage and computation of big data, hadoop is one of them. Hadoop can be used for implementation of programming model like MapReduce which is very efficient for processing the shorter jobs with low response time. MapReduce framework, which is popular for computation of big data in parallel, distributed across the cluster. In our experiment we are analyzing scheduling of each task in MapReduce[11] framework with the help of two applications Word count and grep on FIFO (first in first out) scheduler, Fair scheduler and capacity scheduler. The jobs are submitted simultaneously for execution to analyze the task scheduling. We tried to varied the workload as well as Map Tasks on each slaves to observe the effect on tasks scheduling. Experiment has been carried out on text files of 1GB, 2GB and 5GB with variations in Map Tasks as 1, 2, 3, 4 and 5 on each slave nodes, before executing the jobs. In the execution, first we submit Grep followed by Word Count in all the above cases for different workloads with different Map Tasks. We observed that in FIFO scheduler jobs are submitted as per the policy first in first out,the jobs that are submitted first will be executed first. In Fair scheduler and Capacity scheduler all the jobs are given the equal share of resources, means both the job executes simultaneously. Observation of results can help us to conclude that in all the three schedulers i.e FIFO schedeuler, Fair schedeuler, and Capacity schedulers, FIFO scheduler takes more turnaround time for bigger data size where as it outperforms for the shorter jobs. However, now a days much more big data applications are developed with MapReduce model which requires low turnaround time for the larger jobs as well as for shorter jobs. As a result, it becomes necessary to verify the performance of MapReduce, especially for larger jobs which is more popular now a days and which has attracted more and more attentions from research, industry and academia.**

Keywords: Hadoop, MapReduce Performance, HDFS, Scheduler, scheduling, FIFO, FAIR and CAPACITY.

I. INTRODUCTION

Big data can help to make the efficient and strategic decisions. There are many source available today from which data is generated. Every day google [14] generates 2.5 quintillion bytes of data but still the growth in data is increasing. Facebook[5] process the data using Hadoop, where it users generates 500+ terabytes of data each day. Facebook[5] has a single largest hadoop system which can store and process 100 petabytes of data which is one of the biggest cluster in the world. Various sources like social media, news channels, scientific labs, meteorological department generates huge amount of data every day. This big data requires most effective storage and efficient analysis techniques which can bring some knowledge out of it which is a great challenge today for the data world. Analysis of these cases brings up a challenge in the improvement of Big Data platforms. MapReduce is such a programming model which is most popular, where parallel and distributed computing can be carried out which is adaptive and it can be further scalable as per the user requirement[9].

Big Data varying in the size of petabytes can be processed by an open source platform Apache Hadoop[1]. Hadoop an open source project developed by Doug Cutting owner of Apache Lucene. It can be used for big data storage and further for batch processing across the distributed clusters of commodity hardware. It is very much more adaptive and scalable. Hadoop can be scaled as per the user requirement[12]. Data are stored in blocks which are further replicated at different locations with The high availability of data around the cluster nodes.It has two very important components HDFS(Hadoop distributed file system) [1][10] and MapReduce[9]. Hadoop distributed file system as the first component provide the storage of big data in blocks replicated across the cluster where as MapReduce as a second component provide efficient, parallel and distributed computation of Big Data.

Implementation of MapReduce on Hadoop followed by the master-slave architecture. There are many number of slaves connected with a single master node. Monitoring of slaves and fulfilling the resources requirement is carried out by the master node. Aliveness of slave nodes in the cluster are monitored periodically after some interval time. Further MapReduce divide the job into various tasks for effective and efficient processing.Two types of tasks i.e 'Map Task' and 'Reduce Task' helps to process the big data. Scheduling of tasks on the cluster nodes plays a very vital role for the improvement in turnaround time, and resource utilization [23] [7][21]. In previous studies several issues has been communicated related to MapReduce scheduling. Some issues were still not optimized by most proposed promising MapReduce schedulers which further leads to poor performance. So a significant research can be carried out to improve performance of MapReduce by improving task scheduling techniques.[9][23][20][7].

Today's growing trend in processing Big Data using MapReduce motivated us to analyze different scheduling techniques with various workloads. In the following sec-

tions paper provide the detailed overview of related work in section II. In the section III we explained the existing MapReduce Task scheduling mechanism, followed by section IV which gives an overview of existing scheduling of tasks in MapReduce. Section V provides the detailed experimental setup. In Section VI evaluates the performance of existing schedulers such as FIFO, Fair, and Capacity based on experimental results. In Section VII, results for the evaluation are shown in tabular format with the respective graphs. In the last section VIII we conclude the paper with some of research directions.

II. Related Work

Data sets size are increasing with increase in use of internet, where the data is collected from various sources like digital photos, social media, online transactions,etc. Massive amount of data collected may be structured and unstructured formats. Computation of big data can be carried out with the help of open source platform Hadoop MapReduce. To process jobs in MapReduce it is divided into number of task in series. Scheduling of task in MapReduce model is key factor and important aspect need be considered. Better techniques can be used to schedule job and improve the response time of the cluster.

There are many techniques for scheduling task in Hadoop Mapreduce. FIFO scheduling technique is a default where job coming first for execution in queue has the highest priority. FIFO scheduling has disadvantage were longer jobs are scheduled before shorter jobs which leads to starvation. Fair Scheduling technique share all the resources across the cluster nodes equally for all jobs. Capacity scheduling technique minimizes the resource utilization across the cluster node, which was introduced by Yahoo.

Many authors had presented the review papers related to job scheduling techniques in Hadoop MapReduce for big data processing. Authors[20] reviewed and tested the existing scheduling techniques in Hadoop MapReduce with their advantage and disadvantage. [18] presented a survey on scheduling algorithms in Hadoop MapReduce. Authors[23] from university of California presented a techniques called as LATE Longest approximate time to end which improve performance of MapReduce computation model in heterogeneous environment. Authors[6] [19] proposed a techniques to improve the MapReduce performance in heterogeneous environment by improving the speculative task execution. Facebook[22] provide a scheduling technique, to improve performance and response time by delaying some task, called the technique as Delay scheduling. In deadline scheduler [16] user imposes the deadline constraints before job scheduling to increase system utilization. To improve resource utilization many authors [17][15] proposed different techniques to complete scheduling of jobs. Matchmaking scheduling author[13], proposed a technique to which mark every node in the cluster by the locality marker, further ensuring that every node got an equal chance to hold a task which is local. These above scheduling techniques improved scheduling,

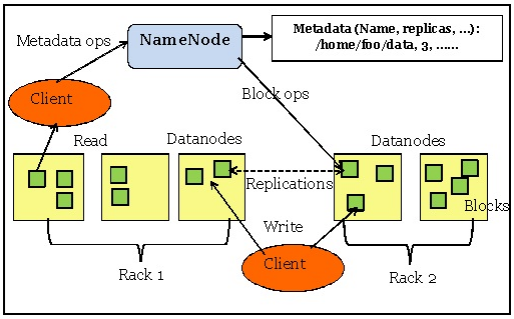

Fig. 1: Architecture of Hadoop Distributed Filesystem [2]

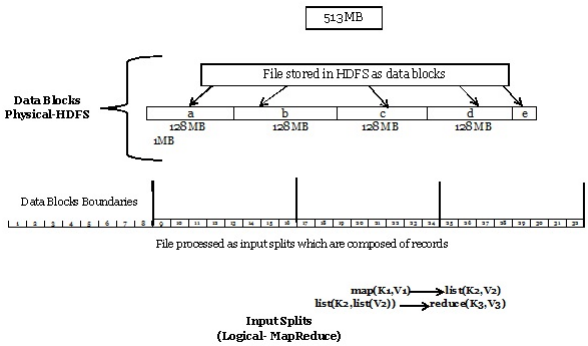

Fig. 2: Blocks in HDFS[4]

with improved data locality, resource allocation and cluster resource utilization.

III. MapReduce Task scheduling mechanism

In this section we explain in details the mechanism used by Hadoop to schedule the work across a cluster. MapReduce implementation in Hadoop closely resembles Googles MapReduce implementation[11]. Hadoop open source platform follows master-slave architecture. A single master controls number of slaves. Input files in hadoop resides in HDFS (Hadoop distributed file system) throughout the cluster. The input is further divided into the even sized chunks of size 64MB, 128MB and 256MB[1].

The size of chunks can be modified by the user as per their requirement. Each Mapreduce job are further divided into series of tasks. There are two type of task 'Map Task' and 'Reduce Task'. Each input chunks are processed by map task, where output of map task is a key-value pairs which act as an input to the reducer. Map outputs splits into a groups based on key. These keys and value are intermediate data copied by the reducers in copy phase. Data further sorted by key in sort phase where as a reducer is applied on data to reduce it and produce the output in key-value pairs.

Hadoop executes several maps and reduces simultaneously on each slave. By default two map task and one reduce task is executed to overlap computation and I/O. Each slave node communicate with master node to update the empty task slots available with them. Scheduler assigns the task if the slave node update regarding the empty slots. Response time

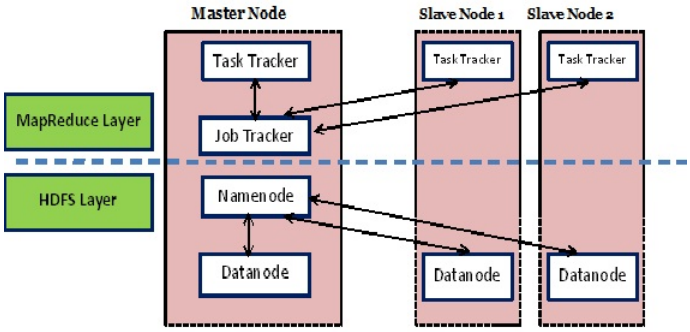

Fig. 3: High Level Architecture of Hadoop[3]

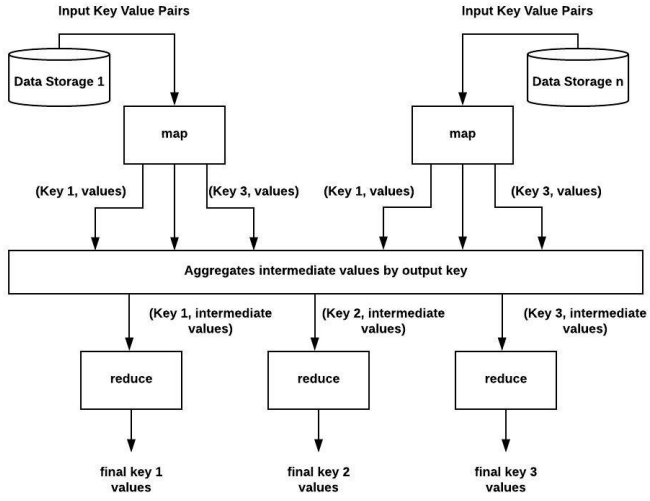

Fig. 4: MapReduce Computation[23]

play a very vital role for short jobs where a user waits for instant answer for example queries on debugging of log data, counting, monitoring and for business intelligence. Major use MapReduce computation framework is carried out for Short jobs.

IV. Existing Scheduling Techniques

Schedulers are used to schedule the task and acquire resources from the cluster to execute the task as per the requirement. Hadoop has three default schedulers FIFO, Capacity, and Fair Schedulers. Different schedulers has their own advantage and disadvantage as per the scheduling policy they adopts. We will further explain the default schedulers in details with their advantage and disadvantage. In the past plenty of research has been carried out to improve the performance of hadoop schedulers[23] [7][22][16] and they are further classified as homogeneous schedulers and heterogeneous schedulers, out of which some of them are static and some of them are dynamic. There are very few schedulers which work well in homogeneous environment and in heterogeneous environment both. In homogeneous environment hadoop assumes that computing nodes in the cluster are same computing capacity, where as in heterogeneous[23] environment the computing nodes in the cluster have different computing capacity. These scheduler are also classified based on their scheduling strategies such as static scheduling and dynamic scheduling. In static scheduling job is allocated to the processor before the execution of the program, where as in dynamic scheduling it allocates the job to the processor at the time of execution of the program.

A. First In First Out Scheduler

Application are submitted to the queue in particular order, FIFO[20][12] scheduler run the application as per the submission (First in first out) manner. Request for resources are made for first application, once its completed, the request of next application are completed available in the queue, which is continued further. FIFO scheduler is simple to understand where as it does not require any configuration, it is not feasible to use in shared cluster as it has many disadvantages. Smaller applications need to wait for resources when resources are been used by larger applications. When a new application or job arrives, the Job Tracker takes the job which is submitted first in the queue which is irrespective of size of the job and the priority.

1) Advantages: 1. FIFO scheduling technique is the most efficient and simplest among all default the schedulers. 2. The jobs or applications are executed as per the order of their submission.

2) Disadvantages: 1. FIFO scheduling is non pre-emptive scheduling technique. This techniques is not feasible for the interactive jobs. 2. A major drawback is bigger applications or jobs will acquire all the system resources delaying the execution of short jobs running behind it. 3. FIFO Scheduler does not balance the resource allocation across the cluster between long running jobs and short running jobs. 4. Its another drawback is data locality is not considered and reduced, it creates starvation of jobs.

B. Cacpcity Scheduler

A Scheduler which allows multiple-tenants to share a large cluster securely between organization with a minimum capacity guarantee. Applications allocates resources in a time interval manner as per the allocated capacities[18][6]. It maximizes throughput of the cluster with better utilization of the resources among the multi-tenant in the cluster. The cluster is divided into different organization, where each organization has their own separate private set of computing resources which are sufficient to meet the organization's SLA in peak condition. It has poor utilization with difficultly in managing independent cluster overhead one per organization. Organization shares the cluster in cost effective manner without creating their own private clusters which is economically beneficial to all organization participating in it. It provides the benefit that any organization can use the excess resources available which not being used by remaining organizations. It provides elasticity to the organizations economically in a cost-effective manner.

1) Advantages: 1. Capacity scheduler maximizes throughput of the cluster with better utilization of the resources among the multi-tenant. 2. Use of unused resources among the jobs in the cluster within the queues. 3. It provide support the hierarchical queues feature, with extended elasticity.

2) Disadvantages: 1. It is one of the complex schedulers among all the three schedulers. 2. It has complexity in selecting the proper queue. 3. Some limitations in regards of pending jobs, maintaining fairness and stability among the user in the cluster.

C. Fair Scheduler

Fair[8] scheduling policy takes care of assigning resources to the jobs, where all the jobs in the cluster gets an average equal share of resources to complete the job. Single job running in the cluster use entire resources of the cluster. When multiple jobs are submitted, they are provided the free task slots, so each can get roughly same amount of CPU time. It share the resources equally which helps short job to finish it in reasonable time without starving the long jobs. Fair scheduler provides the best policy to share the cluster among the multiple users[15]. This policy can work with job priorities where it use priorities as the weight to determine the fraction of total compute time of each job. The scheduling policy maintains the jobs into the pools, where resources are divided equally among the pools. Each user has separate pool, as each one can share the equal share of resources in the cluster. In every pool, jobs can be scheduled by on the cluster using two scheduling policies Fair and FIFO as per the user recommendation. Like capacity scheduling policy fair scheduler assigns guaranteed minimum shares among the pools. If a pool has any jobs, it gets the minimum shares, but if the pool does not require the amount of resources available, then excess is shared among other pools . If user do not get the minimum share which is required for some period of time, it will preempt the jobs of other pools. Fair Scheduler provides the facility to limits concurrent running jobs per user and each pool. This is very helpful when user submits many jobs simultaneously[20]. When multiple concurrent jobs are running in the cluster it may be possible that intermediate data can occupy the complete disk space on the cluster which needs to be monitored. Limiting job submission in fair scheduler help the load balancing in the cluster where new submitted job will wait for earlier job to get complete. Execution of jobs for each user or pool is selected as per the priority assigned and based on their submission. Fair scheduler is optimized in terms of load balancing where job waits for some of the dependency on the external service such as IO, web service and database which can overload the cluster by too many maps or reduce task running simultaneously.

1) Advantages: 1. Scheduler provides fair and dynamic share of resources among the pools. 2. Very helpful in improving starvation, where shorter jobs finishes first compared o larger jobs. 3. It has a dynamic control over limiting multiple jobs running across the cluster for each user and each pool.

2) Disadvantages: 1. Fair scheduler configuration much more complicated and complex compared to others. 2. The policy does not consider the weight of each job, which further create a unbalance performance for each pool ans users 3. Limitation on concurrent running jobs across the user and the pool, will be based on the availability of resources.

V. Experimental Setup

Hadoop is a distributed open source platform deployed for commodity machines. The framework is divided into two part one for storage and other for parallel and distributed computation. A open source platform hadoop can be used by various operating system like linux, Mac-OS and many others. But it is best supported in the linux platform. In our experiment we installed hadoop on Ubuntu 18.04.1 LTS operating system. To evaluate the performance of the existing scheduler, we created a cluster of four machines with cluster node configuration given below in the tables. Our cluster consist of total four machine. Three of them work as worker node i.e slaves and another machine act as a master node. These machine are interconnected with each other through the switch, all of them lay in the same network. Two node in the cluster master node and slaves node has to manage the complete cluster, by running Job Tracker and Name Node on master node and Task Tracker and Data Node on slave node. Nodes has Ubuntu 18.04.1 LTS as an operating system and Open JDK 10.1 to run hadoop applications. Word count and Grep application require java as necessary software to run on the hadoop cluster. Hadoop 1.2.1 execution environment was installed on the nodes for creating the cluster. Below Table provide the detailed configuration parameters used for hadoop. Detailed Configuration of each machine is shown below in the tables :

TABLE I: Master Node

MasterNode	Details
Operating system	Ubuntu 18.04.1 LTS
RAM	12 GB
ROM	1TB GB
Processor	Intel® Core™ i7-4790

TABLE II: Slave Node 1

Slave1	Details
Operating system	Ubuntu 18.04.1 LTS
RAM	8 GB
ROM	1TB GB
Processor	Intel® Core™ i7-4790

TABLE III: Slave Node 2

Slave2	Details
Operating system	Ubuntu 18.04.1 LTS
RAM	8 GB
ROM	1TB GB
Processor	Intel® Core™ i7-4790

TABLE IV: Slave Node 2

Slave3	Details
Operating system	Ubuntu 18.04.1 LTS
RAM	4 GB
ROM	1TB GB
Processor	Intel® Core™ i7-4790

VI. Performance Evaluation

In the previous studies authors made a comparative study of job scheduling techniques along the experimental results[20]. Experiment is carried out for analyzing scheduling of each task in MapReduce framework with the help of two application Word count and grep. Analysis of task scheduling is carried out on FIFO (first in First out) scheduler, Fair scheduler and capacity scheduler by executing the jobs, Word count and Grep. The jobs are submitted simultaneously for execution to analyze the task scheduling. We varied the workload and Map Task on each slave for the every job i.e 1GB, 2GB and 5GB text files and 1,2,3,4 and 5 Map Task. While executing the jobs, we submit Grep followed by Word count in all the above case for different workload and Map Task. When execution started we tried to observe that the slave node executes the Map Task as per the configuration. Also we tried to find out the effect on the performance of the scheduler by varying the Map Task and input sizes. For running default schedulers in hadoop cluster need to do some configuration changes on master node. By default execution of jobs can be carried out with default scheduler FIFO. So to use another scheduler for the execution of jobs we need to configure the scheduler. To use Fair scheduler and capacity scheduler need to modify configuration of schedulers in the contrib/fairscheduler and contrib/capacityscheduler directory these are the jar file.

These jar file need to be copied in the classpath to use the particular scheduler. For the performance evaluation of the schedulers, we used two applications Word count and Grep. These application in Hadoop MapReduce are taken benchmark applications by many researchers to perform the evaluations. Input to these application is a text files, where word count application, count the number of occurrence of words in the files and grep search application count the occurrence of particular word in the files provided by the user, which is mostly used for data mining.

Turnaround Time : Sum of total time required when job submitted until the complete execution of the job .

TABLE V: Configuration parameters for hadoop

Parameters	Details [20]
Block Size	64 MB
Interval of Heartbeat	3 s
Reduce tasks per node	1
Replication factor	2

VII. Tabular Results

We tried to analyze performance of following schedulers, FIFO scheduler, Fair scheduler, and Capacity schedulers by running the Word count and Grep applications. We varied different data sizes with variation in Map Task on each slave. Result of experiment are given below in the table which shows the Turnaround time/ Response Time in seconds for different data size and different Map Task. We varied the data size as well as Map Task for our experiment by configuring the parameter "mapred.tasktracker.map.tasks.maximum" of "mapred-site.xml" in all the clients (slaves). We observed that in FIFO scheduler jobs are submitted as per the policy first-in-first-out, the jobs that are submitted first will be executed first. In Fair scheduler all the jobs are given the equal share of resources, means both the job executes simultaneously. Where as in capacity scheduler share are divided but still one job needs to wait for other job to complete.

We clarify our results by presenting the below three observations.Our first observation for word count application, where FIFO scheduler outperforms for shorter jobs of data size 1GB and 2GB where as FAIR scheduler outperforms for larger jobs of data size 5GB followed by capacity scheduler. Secondly in grep application fair scheduler outperforms in all types of workload as it executes both the jobs providing the equal share of resources across the cluster. Third and overall observation regarding improving response time of MapReduce, increase in Map Task on slaves, decrease overall execution time of jobs. Variation in Map Task can be carried out as per the system configuration. For higher configuration more Map Task can be assigned, where as to lower configuration minimum Map Task can be assigned. But it has a disadvantage of overloading the cluster with maximum task failure and increase speculative executions. These above experimental results help us to find the load which can be berried by each slave node.

TABLE VI: Turnaround time in second for 1GB data

MapTask	1	2	3	4	5
FIFO	92	82	76	70	63
FAIR	93	84	79	73	66
CAPACITY	93	86	82	75	68

TABLE VII: Turnaround time in second for 2GB data

MapTask	1	2	3	4	5
FIFO	158	127	103	91	80
FAIR	159	132	106	65	87
CAPACITY	162	135	109	97	91

TABLE VIII: Turnaround time in second for 5GB data

MapTask	1	2	3	4	5
FIFO	382	360	318	278	183
FAIR	374	359	290	238	177
CAPACITY	378	364	294	242	179

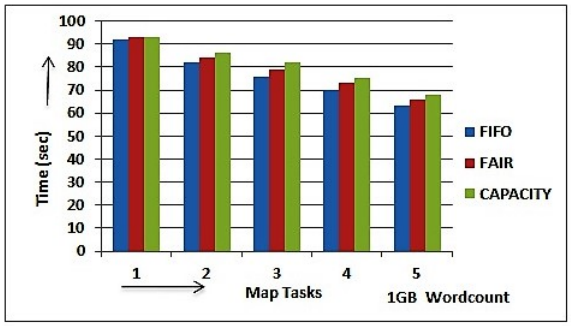

Fig. 5: Turnaround Time for 1GB Word-Count Application

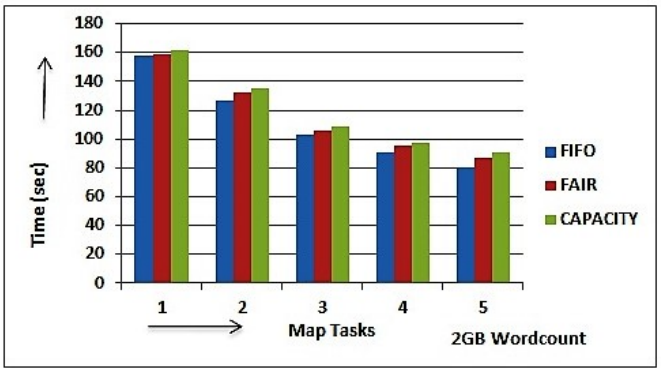

Fig. 6: Turnaround Time for 2GB Word-Count Application

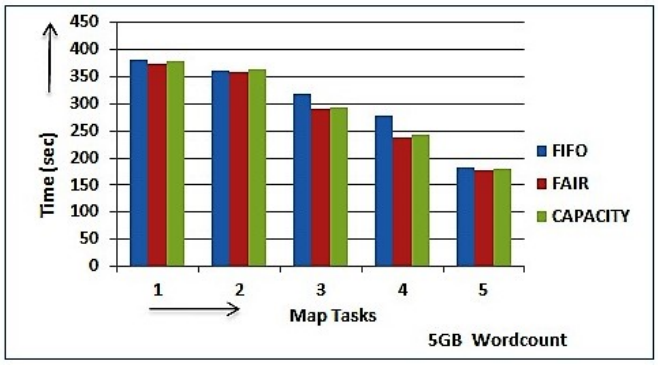

Fig. 7: Turnaround Time for 5GB Word-Count Application

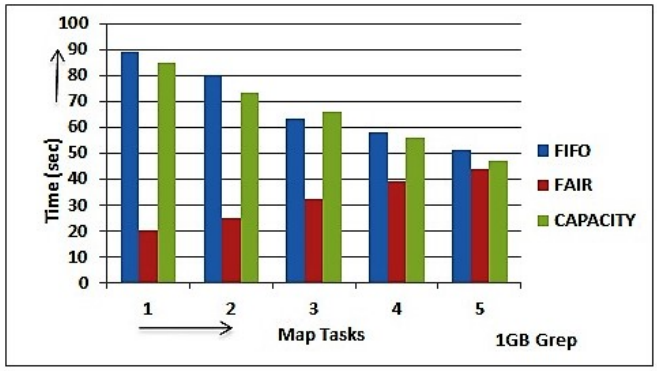

Fig. 8: Turnaround Time for 1GB Grep Application

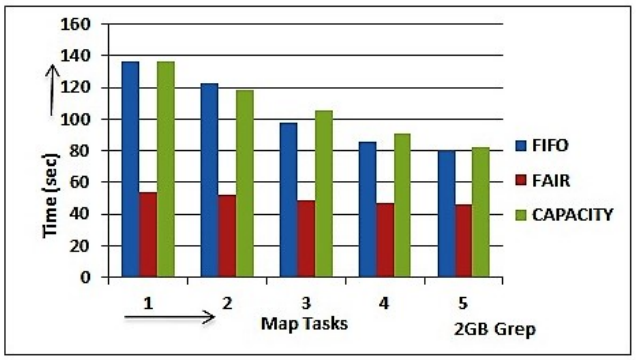

Fig. 9: Turnaround Time for 2GB Grep Application

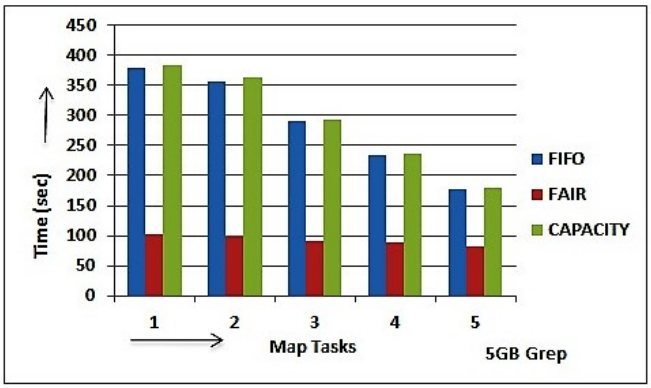

Fig. 10: Turnaround Time for 5GB Grep Application

VIII. Conclusion

Scheduling jobs and managing resources in the cluster is carried out with the help of schedulers. In big data, processing of data is very important aspect which needs to be considered to improve the performance of MapReduce, which can be achieved by improving job scheduling techniques. There are still many issues in processing big data which need to be improved. Some of the issues to handle big data has been communicated by designing variety of schedulers. In this paper we present the comparative study of schedulers like FIFO, FAIR and CAPACITY schedulers with their drawbacks and advantages. Execution of jobs in the cluster is managed by the schedulers by reserving the resources like Memory, CPU, IO and user demands etc. By observing the experiment results we can conclude the Fair scheduler and Capacity schedulers can be used for larger jobs where fair and equal utilization of resources is required. Where as FIFO schedulers can be used for shorter jobs. Our Overall observation regarding improving response time of MapReduce is increase in Map Task on slaves, decrease overall execution time of jobs. Variation in Map Task can be carried out as per the system configuration. For higher configuration more Map Task can be assigned, where as to lower configuration less Map Task can be assigned. But this has a disadvantage of overloading the cluster with maximum task failure and increase in speculative executions. Finally

we can conclude that scheduler must be selected to schedule the job as per the size of workload and based on system configurations working in the cluster, to obtain early results from the cluster nodes.

REFERENCES

[1] Apache hadoop. *https://hadoop.apache.org/*.

[2] Hadoop file system archtecture. *https://bigdatapath.wordpress.com/2018/02/27/hdfs-overview/*.

[3] Hadoop high level archtecture. *https://www.dezyre.com/article/hadoop-architecture-explained-what-it-is-and-why-it-matters/317*.

[4] Hdfs blocks. *https://www.dummies.com/programming/big-data/hadoop/input-splits-in-hadoops-mapreduce/*.

[5] How big is facebook's data? *https://techcrunch.com/*.

[6] Quan Chen, Daqiang Zhang, Minyi Guo, Qianni Deng, and Song Guo. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 2736–2743. IEEE, 2010.

[7] Dazhao Cheng, Jia Rao, Yanfei Guo, Changjun Jiang, and Xiaobo Zhou. Improving performance of heterogeneous mapreduce clusters with adaptive task tuning. *IEEE Transactions on Parallel and Distributed Systems*, 28(3):774–786, 2017.

[8] Ya-Wen Cheng and Shou-Chih Lo. Improving fair scheduling performance on hadoop. In *Platform Technology and Service (PlatCon), 2017 International Conference on*, pages 1–6. IEEE, 2017.

[9] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[10] Mohd Rehan Ghazi and Durgaprasad Gangodkar. Hadoop, mapreduce and hdfs: a developers perspective. *Procedia Computer Science*, 48:45–50, 2015.

[11] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. *The Google file system*, volume 37. ACM, 2003.

[12] Rong Gu, Xiaoliang Yang, Jinshuang Yan, Yuanhao Sun, Bing Wang, Chunfeng Yuan, and Yihua Huang. Shadoop: Improving mapreduce performance by optimizing job execution mechanism in hadoop clusters. *Journal of parallel and distributed computing*, 74(3):2166–2179, 2014.

[13] Chen He, Ying Lu, and David Swanson. Matchmaking: A new mapreduce scheduling technique. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 40–47. IEEE, 2011.

[14] Bernard Marr. How big is google's data? *https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/*, 2018.

[15] Xiaoqiao Meng, Jian Tan, and Li Zhang. Resource aware scheduling in a distributed computing environment, December 1 2015. US Patent 9,201,690.

[16] Quentin Perret, Gabriel Charlemagne, Stelios Sotiriadis, and Nik Bessis. A deadline scheduler for jobs in distributed systems. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 757–764. IEEE, 2013.

[17] Jorda Polo, Claris Castillo, David Carrera, Yolanda Becerra, Ian Whalley, Malgorzata Steinder, Jordi Torres, and Eduard Ayguadé. Resource-aware adaptive scheduling for mapreduce clusters. In *Proceedings of the 12th International Middleware Conference*, pages 180–199. International Federation for Information Processing, 2011.

[18] B Thirumala Rao and LSS Reddy. Survey on improved scheduling in hadoop mapreduce in cloud environments. *arXiv preprint arXiv:1207.0780*, 2012.

[19] Xiaoyu Sun, Chen He, and Ying Lu. Esamr: An enhanced self-adaptive mapreduce scheduling algorithm. In *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*, pages 148–155. IEEE, 2012.

[20] Mohd Usama, Mengchen Liu, and Min Chen. Job schedulers for big data processing in hadoop environment: testing real-life schedulers using benchmark programs. *Digital Communications and Networks*, 3(4):260–273, 2017.

[21] Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares, and Xiao Qin. Improving mapreduce performance through data placement in heterogeneous hadoop clusters. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–9. IEEE, 2010.

[22] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European conference on Computer systems*, pages 265–278. ACM, 2010.

[23] Matei Zaharia, Andy Konwinski, Anthony D Joseph, Randy H Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *Osdi*, volume 8, page 7, 2008.