Scheduling Algorithms in Map Reduce

Sahil Sangani
Department of Information and Technology,
Gandhinagar Institute of Technology,
Gandhinagar, Gujarat, India
sahilsangani2@gmail.com

Abstract: Data generated in the past few years cannot be efficiently manipulated with the traditional way of storing techniques as it is a large-scale dataset, and it can be structured, semi-structured, or unstructured. To deal with this kind of enormous dataset Hadoop framework is used, which supports the processing of large dataset in a distributed computing environment. Hadoop uses a technique named as MapReduce for processing and generating a large dataset with a parallel distributed algorithm on a cluster. It automatically handles failures and data loss due to its fault-tolerance property. The scheduler is a pluggable component of the MapReduce framework. Hadoop MapReduce framework uses various scheduler as per the requirements of the task. FIFO (First In First Out) is a default algorithm used by Hadoop, in which the jobs are executed in the order of their arrival. This paper will discuss myriad of schedulers such as FIFO, Capacity Scheduler, LATE Scheduler, Fair Scheduler, Delay Scheduler, Deadline Constraint Scheduler, and Resource Aware Scheduler. Besides these schedulers, we also conducted study of comparison of schedulers like Round Robin, Weighted Round Robin, Self-adaptive Reduce Scheduling (SARS), Self-adaptive MapReduce Scheduling (SAMR), Dynamic Priority Scheduling, Learning Scheduling, Classification & Optimization-based Scheduler (COSHH), Network-Aware, Match-matching, and Energy-Aware Scheduler. Hopefully, this study will enhance the understanding of the specific schedulers and stimulate other developers and consumers to make accurate decisions for their specific research interests.

Keywords: *Big Data, Hadoop, MapReduce, HDFS, Scheduler, Job Scheduling, Jib Tracker, Task Tracker*

*****

I. Introduction

Concept of the "Big Data" arise with the advent of the Internet. Due to some technological advancements, a huge amount of data generated in past several years from various sources such as social media, transactional data from enterprise applications, sensors, electronic devices, machine-generated data, electronic files, a huge number of high definition videos and many more. (Seethalakshmi et al, 2018). It is arduous to be maintained by conventional methods. These kinds of data can be processed by Parallel Processing and Distributed Computing. The most popular tool for the Big Data is based on the Apache open source Hadoop Map Reduce environment, which offered by the most commercial providers such as Microsoft, Oracle, IBM, Amazon, etcetera. Hadoop environment has a rich set of libraries such as Hbase, Hive, Pig, Mahouat, and Oozie. Processing of a large amount of data can be done by the Job Schedulers. Therefore, here we discuss a comprehensive study of all Hadoop Schedulers, which gives clear idea for the best-fitted scheduler for a particular task or job. This paper will depict the Hadoop Architecture, MapReduce working, and various Job Schedulers with their advantages and disadvantages.

Hadoop &Hadoop Architecture

Hadoop is a software library framework that allows for the distributed processing of large datasets across clusters of computers using a simple programming model. It provides facilities like distributed storage and computation across clusters of computers. Hadoop Distributed File System (HDFS) – storage part, and MapReduce **-** processing part is a core part of Hadoop Architecture. The Hadoop files system has master/slave architecture where 'name node' acts as a master and 'data nodes' acts as a slave. Figure 1 illustrates the Hadoop Architecture.

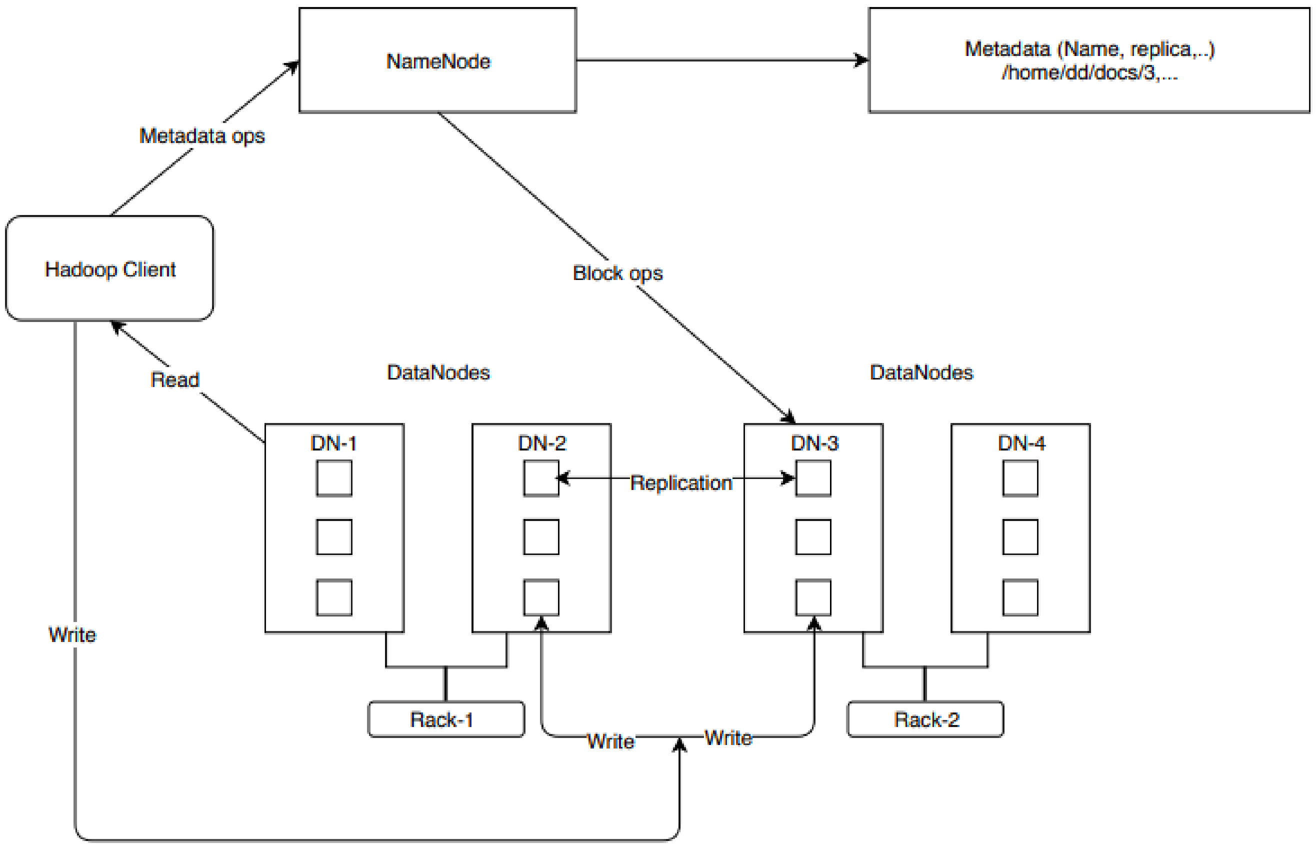


Figure 1 Hadoop Architecture

The NameNode(Master) manages metadata of HDFS – the directory tree of all files in the file system and tracks the files across the cluster.The DataNode(Slave) is responsible for storing actual files in the form of blocks. NameNode and DataNode are in constant communication. The DataNode sends a heartbeat to the NameNode, which refers to DataNode is active. If it fails, then the NameNode marks it fails and suspend it for further operations. The NameNode also manages the data replication over various slaves for fault-tolerance, performance, and reliability. The process starts with the client request, which is verified by the NameNode and allocating the DataNode. Then, the NameNode splits the files in the chunk of specific sized block usually 64MB, and this block is store by the allocated DataNode. If the NameNode fails, then the secondary NameNode will be active and avoid the cluster failure. After resolving the problem, the primary NameNode will join the cluster as a secondary NameNode.

MapReduce

MapReduce is a data processing technique and a robust programming model based on Java, designed for processing large scale dataset in a distributed and parallel, on large clusters of commodity hardware in a reliable manner. The MapReduce algorithm mainly two tasks, Map and Reduce. It usually divides the input datasets into the number of independent chucks and processing will be done on them in parallel by working nodes. The MapReduce algorithm takes a list as output and also produces output in the form of a list. The map phase takes a set of data to produce intermediate results (Key-Value Pair), which is further fetched by reduce phase as an input to produce the final output after applying some computations. The entire execution process is controlled by the Jobtracker(master) and Tasktrackers(slave). The Jobtracker is responsible for the complete execution of a submitted job by the client, whereas task trackers perform the given job.

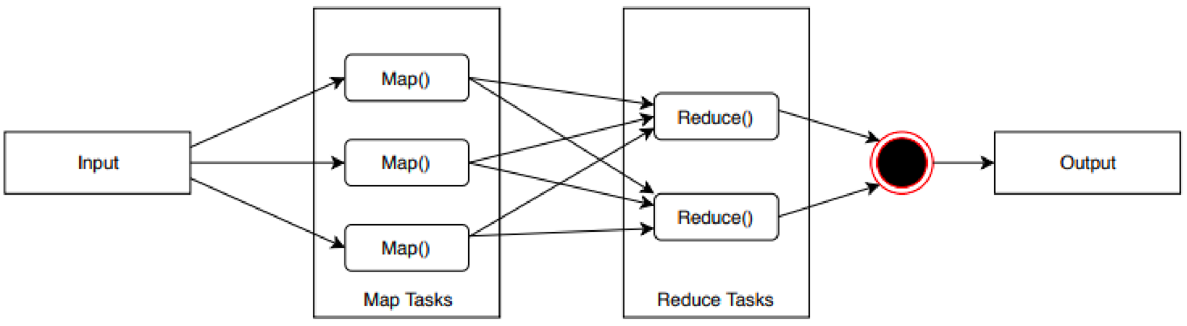


Figure 2 Map Reduce Process

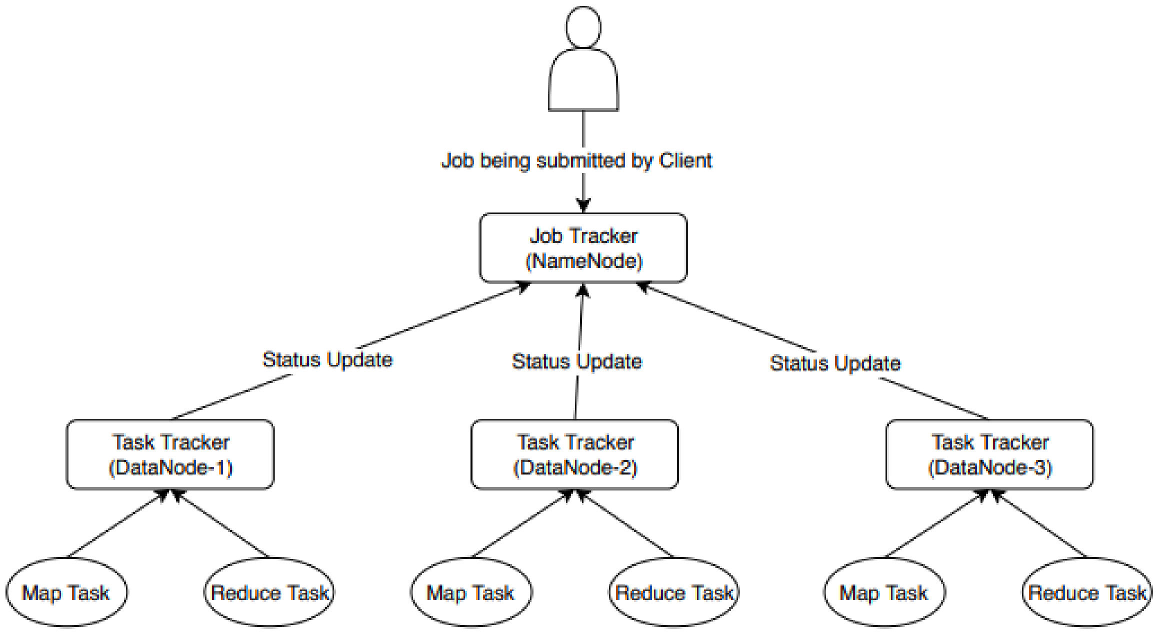


Figure 3 Map Reduce Architecture

Locality, Synchronization, and Fairness: These are the three major scheduling issues of MapReduce. Various scheduling algorithms have been proposed in past years in order to solve these issues; some of the algorithms are described in the following section.

First in First Serve (FIFO) Scheduler

(Patel and Sonaliya et al, 2015) FIFO is the traditional and default Hadoop scheduler. It schedules the job based on their priorities in first-come-first-serve (first-out) order. The main task is divided into sub-task which are assigned at the end of the job queue by a task tracker. Job tracker pulls the oldest job from the job-queue without considering any priority or size of the particular job. It works until the all task have been done from the job-queue. In FIFO, starvation of the job can be avoided, but it takes a longer response time for smaller jobs, which reduced resource utilization. It works fine only for a single type of job. This limitation overcome by the use of Fair and Capacity Schedulers.

Capacity Scheduler

The algorithm for both Capacity scheduler and Fair scheduler almost similar. The key difference is that fair scheduler utilizes pools, whereas capacity scheduler utilizes queues. (Seethalakshmi et al, 2018) It is specifically designed by Yahoo for a scenario where a number of users large significant, and to fairly allocate computational resources among them. An organization has assigned with the specific queue, which has dedicated resource, and it follows the FIFO strategy. If a queue is weighty, it seeks for the yet unallocated resource. To utilize the resources at its maximum capacity, it permits re-allocation of resources of the free queue. As soon as, jobs arrive from another organization of that queue, the running job is finished, and the resources are passed to the original queue.

Fair Scheduler

In order to fairly manage the Hadoop cluster, this mechanism originally developer at Facebook. (Sharma et al, 2015) Each user is assigned with the equal share of cluster resource over time. Users assign jobs to pools, where each pool is allocated with a dedicated least number of Map and Reduce slots. The new pool has been allocated free slots of ineffective pools. The scheduler will kill tasks whenever a higher priority task comes in the same queue to assign that slot to it. Different pools are also allocated priority criteria. Here, priority-based tasks are planned interleaved.

Longest Approximate Time to End (LATE) Scheduler

The principal objective of the LATE scheduler is to optimize the performance of jobs and minimize job response time. A large number of background processes, slow background processes, high CPU load, unavailability of resource, etc. slowdowns some long jobs. The LATE scheduler identifies processes running with the slow speed in the cluster and, creates an equivalent process in the background as a backup; such kind of processes are called speculative execution of tasks. The LATE scheduler is proposed by Zaharia to robustly enhance performance by reducing the overhead of speculation execution tasks. (Zaharia et al, 2012)It optimizes the performance of the job by minimizing job response time. It is highly robust to node heterogeneity. On the other side, it does not ensure reliability.

Delay Scheduler

Delay scheduler is developed to overcome the significant issues of Fair scheduling: Sticky Slot and Head-of-Line, which occurs at the time of fair sharing. It works efficiently in Hadoop cluster because tasks are short, and the task can access each data block from the multiple locations. According to improvised data locality policy of delay scheduler, if data is not available for a task, then task tacker will wait for a fixed amount of time.(Usama et al, 2017) When a node requests a particular task, it will skip too short jobs and start looking for the next job to run. However, it creates a non-local task to prevent starvation, if job skipping continuously long enough. Talking about pros, it is simple, no overhead for complex calculations and resolves the data locality problem, whereas it is not efficient when tasks are much more than an average job, and it has limited slots per node. (Usama et al, 2017)

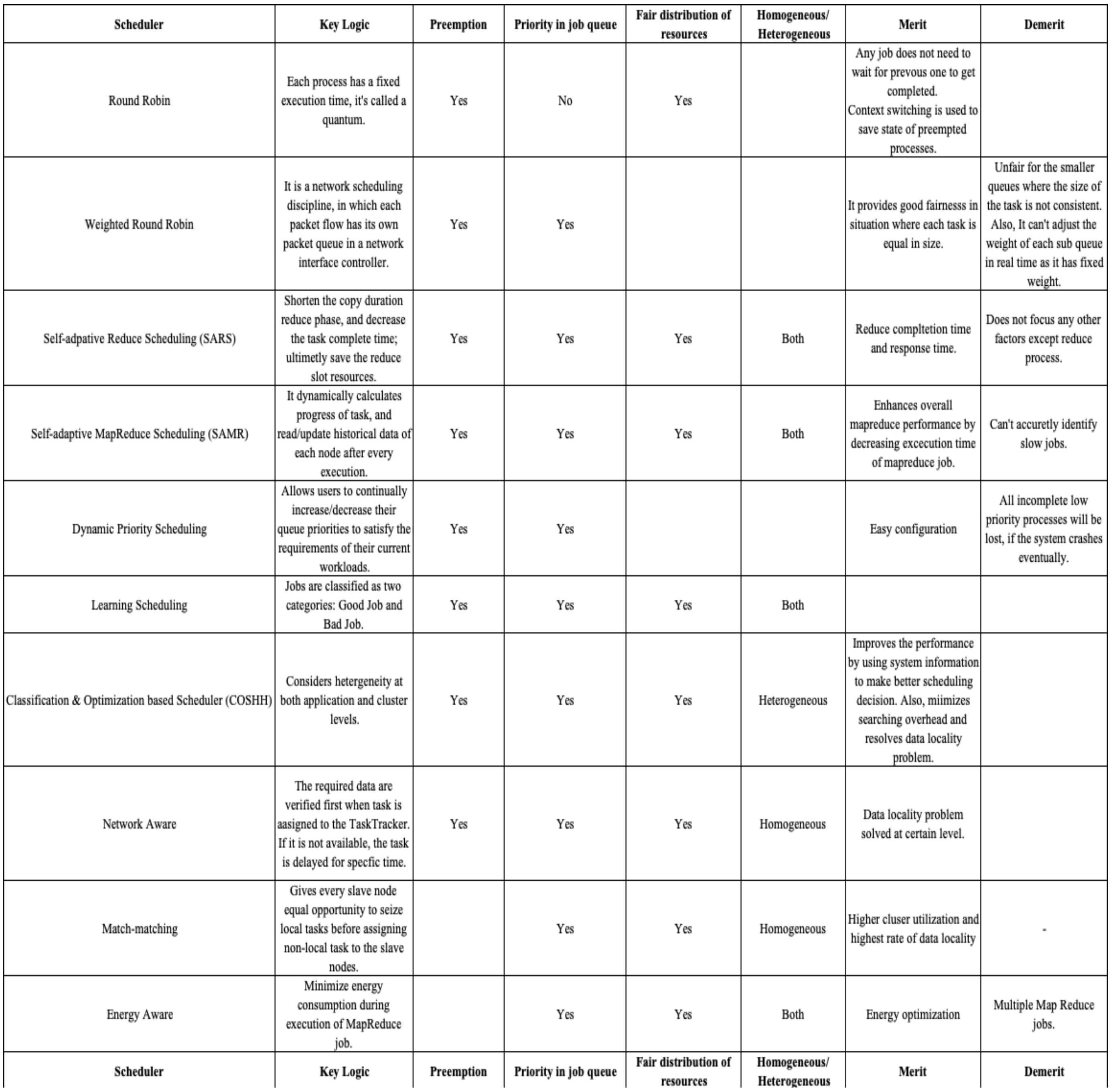
Deadline Constraint Scheduler

Deadline constraint scheduler addresses of issues of deadline and focuses more on improving utilization of the system. In this scheduling, the user specifies the deadline while submitting the jobs. Job tracker computes the minimum slots needed by the cost model of job execution, which considers parameters such as input size of data distribution, map and reduce run time, etc. (Usama et al, 2017)If a minimum number of slots satisfies the requirement, the jobs are allocated for processing, otherwise, the user is notified that slots are unavailable and the user can resubmit the job with modification in deadline. Here, presently running jobs doesn't affect as the jobs aren't killed or freed the slots; instead, the deadline is being modified. Whenever any job is scheduled, the scheduler examines whether or not it will be completed within the given deadline.

Resource Aware Scheduler

Today, Resource Aware Scheduling in Hadoop has become one of the research challenges in cloud computing. (Andrews et al, 2013) Unlike schedulers like Fair scheduler, FIFO, and Capacity scheduler the administrator does not have to ensure sharing of resources manually in this scheduler; instead, various resources such as CPU utilization, disk utilization, memory utilization, IO utilization, and network utilization are shared more efficiently. Two nodes named Master Node and Slave Node, also known as Job Tracker and Task Tracker, is responsible for scheduling. The Job Tracker maintains lists of tasks allocated to each Task Tracker, states of Task Trackers in the cluster, and the queue of the currently running jobs, while the Task Tracker is responsible for the execution of each task configured with the maximum number of available slots. (Usama et al, 2017)The jobs are allotted in the task tracker if resources are available, else either the jobs remained in the wait state until the resources are released or fragmented based on the required size, and assigned to each slot. ‘Free Slot Filtering' and ‘Dynamic Free Slot Advertisement' are two different approaches to Resource Aware Scheduling.

Table 1 Comparison of Schedulers



Scheduler	Key Logic	Preemption	Priority in job queue	Fair distribution of resources	Homogeneous/ Heterogeneous	Merit	Demerit
Round Robin	Each process has a fixed execution time, it's called a quantum.	Yes	No	Yes		Any job does not need to wait for prevous one to get completed. Context switching is used to save state of preempted processes.	
Weighted Round Robin	It is a network scheduling discipline, in which each packet flow has its own packet queue in a network interface controller.	Yes	Yes			It provides good fairnesss in situation where each task is equal in size.	Unfair for the smaller queues where the size of the task is not consistent. Also, It can't adjust the weight of each sub queue in real time as it has fixed weight.
Self-adpative Reduce Scheduling (SARS)	Shorten the copy duration reduce phase, and decrease the task complete time; ultimetly save the reduce slot resources.	Yes	Yes	Yes	Both	Reduce compltetion time and response time.	Does not focus any other factors except reduce process.
Self-adaptive MapReduce Scheduling (SAMR)	It dynamically calculates progress of task, and read/update historical data of each node after every execution.	Yes	Yes	Yes	Both	Enhances overall mapreduce performance by decreasing excecution time of mapreduce job.	Can't accuretly identify slow jobs.
Dynamic Priority Scheduling	Allows users to continually increase/decrease their queue priorities to satisfy the requirements of their current workloads.	Yes	Yes			Easy configuration	All incomplete low priority processes will be lost, if the system crashes eventually.
Learning Scheduling	Jobs are classified as two categories: Good Job and Bad Job.	Yes	Yes	Yes	Both		
Classification & Optimization based Scheduler (COSHH)	Considers hetergeneity at both application and cluster levels.	Yes	Yes	Yes	Heterogeneous	Improves the performance by using system information to make better scheduling decision. Also, miimizes searching overhead and resolves data locality problem.	
Network Aware	The required data are verified first when task is aasigned to the TaskTracker. If it is not available, the task is delayed for specfic time.	Yes	Yes	Yes	Homogeneous	Data locality problem solved at certain level.	
Match-matching	Gives every slave node equal opportunity to seize local tasks before assigning non-local task to the slave nodes.		Yes	Yes	Homogeneous	Higher cluser utilization and highest rate of data locality	-
Energy Aware	Minimize energy consumption during execution of MapReduce job.		Yes	Yes	Both	Energy optimization	Multiple Map Reduce jobs.
Scheduler	Key Logic	Preemption	Priority in job queue	Fair distribution of resources	Homogeneous/ Heterogeneous	Merit	Demerit

II. Conclusion

This paper summarizes taxonomy of various task schedulers considering different parameters such as priority, completion time, resource, etc. in Hadoop MapReduce framework. Moreover, it compares the properties of schedulers. Several issues of Big Data – Data Volume, Data Variety, Data Velocity, Data Veracity, Cost, Connectivity, Security and Privacy, etc. can be handled at a certain extent by task schedulers. We have discussed the features and drawbacks of most of the schedulers. User opts specific scheduler as per the task requirement to gain effective and high throughput output, which ultimately turn into the improved data retrieval and job processing. However, the comprehensive study of scheduling algorithm is an open area for research as it can be enhanced in the future by adding or removing few constraints and making it more generic.

Declaration

Availability of data and material

All relevant data and material are presented in the main paper.

Funding
Not Applicable

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
Authors acknowledge thesupport provided by Department of Information Technology, Gandhinagar Institute of Technology, Gandhinagar, Gujarat, India. Authors are also thankful to the college for giving permission topublish this research.

References

[1]. Mohd Usama, Mengchen Liu, Min Chen. Job schedulers for Big data processing in Hadoop environment: testing real-life schedulers using benchmark programs. (2017) 260-273
[2]. P. Nguyen, M. Halem, T. Simon, D. Chapman, Q. Le. A Hybrid Scheduling Algorithm for Data Intensive Workloads in a MapReduce Environment. Fifth International Conference on Utility and Cloud Computing. IEEE/ACM 2012.
[3]. P. Nimbalkar, D. Gadekar. Survey on Scheduling Algorithm in MapReduce Framework. International Journal of Science, Engineering and Technology Research (IJSETR), Volume 4, Issue 4, (April 2015) 1226-1230
[4]. SeethalakshmiV. ,Govindasamy V. and Akila V. Job Scheduling in Big Data – A Survey. Volume 8. Issue 2. 1000226
[5]. J. Manjaly and C. Edwin. A Relative Study on Task Schedulers in Hadoop MapReduce. International Journal of Advanced Research in Computer Science and Software Engineering. Volume 3 . Issue 5 . (May, 2013). 744-747
[6]. Jyoti V. Gautam, Harshadkumar B. Prajapati, Vipul K. Dabhi, Sanjay Chaudhary, A survey on job scheduling algorithms in big data processing, IEE Conf. Pap. (March 2015).
[7]. Norman Lim, ShikhareshMajumdar, Peter Ashwood-Smith, MRCP-RM: a technique for resource allocation and scheduling of MapReduce jobs with deadlines, IEEE Trans. Parallel Distributed Syst. PP (99) (2016).
[8]. Q. Chen, C. Liu, Z. Xiao, Improving MapReduce Performance Using Smart Speculative Execution Strategy. IEEE Transactions on Computer, 2013.
[9]. MateiZaharia ,DhrubaBorthakur, and Joydeep Sen Sarma, “Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling” , Yahoo! Research ,University of California, Berkeley , 2009.
[10]. B. Andrews and Binu. Survey on Job Schedulers in Hadoop Cluster. IOSR Journal of Computer Engineering (IOSR-JCE). Volume 15, Issue 1 (Sep. - Oct. 2013), PP 46-50
[11]. R.Thangaselvi, S.Ananthbabu, R.Aruna. An efficient Mapreduce scheduling algorithm in Hadoop. International Journal of Engineering Research & Science (IJOER). Vol-1, Issue-9, (December, 2015)
[12]. A. Sharma. Hadoop MapReduce Scheduling Algorithms – A Survey. International Journal of Computer Science and Mobile Computing. Vol. 4, Issue. 12, December 2015, pg.171 – 176