Modern Applied Science; Vol. 13, No. 7; 2019
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

Hadoop MapReduce Job Scheduling Algorithms Survey and Use Cases

Alaa A. Abdallat[1], Arwa I. Alahmad[1], Duaa A. AlSahebAlT amimi[1] & Jaber A. AlWidian[1]

[1] Computer Science Department, Princess Sumaya University for Technology (PSUT), Jordan

Correspondence: Alaa A. Abdallat, Computer Science Department, Princess Sumaya University for Technology (PSUT), Amman, Jordan. E-mail: alaa-abdallat@live.com

Received: May 1, 2019 Accepted: May 30, 2019 Online Published: June 25, 2019

doi:10.5539/mas.v13n7p38 URL: https://doi.org/10.5539/mas.v13n7p38

Abstract

Data is the fastest growing asset in the 21st century, extracting insights is becoming of the essence as the traditional ecosystems are incapable to process the resulting amounts, complying with different structural levels, and is rapidly produced. Along this paradigm, the need for processing mostly real time data among other factors highlights the need for optimized Job Scheduling Algorithms, which is the interest of this paper. It is one of the most important aspects to guarantee an efficient processing ecosystem with minimal execution time, while exploiting the available resources taking into consideration granting all the users a fair share of the dedicated resources. Through this work, we lay some needed background on the Hadoop MapReduce framework. We run a comparative analysis on different algorithms that are classified on different criteria. The light is shed on different classifications: Cluster Environment, Job Allocation Strategy, Optimization Strategy, and Metrics of Quality. We, also, construct use cases to showcase the characteristics of selected Job Scheduling Algorithms, then we present a comparative display featuring the details for the use cases.

Keywords: big data, Hadoop, job scheduling, mapreduce

1. Introduction

The new digital world is growing, new day-to-day habits are adopted and every aspect of the world as we previously know now has a digital equivalent.

Connectivity changing from a luxury to a necessity changed the role the internet is playing, the massive increase in data generating devices and end users, the emergence of the modern terms like the internet of things(IoT) and the new digital life through Social Media are all affecting the amount of data being generated, stored and in need to be processed (Amir & Murtaza, 2015), as a result big data and its frameworks or ecosystems are now the keywords used to indicate the need for distributed, parallel computing. Big Data is dominating the world, all sorts of uses and applications are being explored, developed and enhanced (Bhadani Jothimani, 2016). While the data is only getting bigger the big data ecosystem is becoming more and more complex to meet the changing modern needs and technical requirements, even the original big data ecosystem is becoming obsolete and enhancements ought to be considered. Job Scheduling is one aspect of this evolving ecosystem, more specifically MapReduce Job Scheduling.

MapReduce is the programming model that was intentionally developed to handle very large, scalable datasets. In dis- tributed, parallel fashion (Hashem et al., 2016). It had its default job scheduling mechanism that was based on FIFO (Senthilkumar Ilango, 2016), which later was removed from MapReduce as a component and is now considered a plug- gable component allowing MapReduce Job Scheduling algorithm and technique to be customized per project.

The concept of Job Scheduling is one of oldest concepts in the world of IT, it has been around before the emergence of Big Data and its ecosystem. It has a lot of applications, and implementations, To name the most popular, Operating Systems (OS), Grid Computing, and it can be found in Load Balancers (LB)(Gautam, Prajapati, Dabhi, Chaudhary, 2015). In isolation of any implementation, Job Scheduling is the process of assigning resources to tasks or user-initiated requests. In this work, Job Scheduling is used to represent Hadoop Job Scheduling unless stated otherwise.

Job Schedulers are being constantly perfected to serve within the big data ecosystem. As a general, very high

perspective, job schedulers are aiming to tackle a few issues resulting from the MapReduce paradigm, resources managers and nego- tiators. And untimely, those schedulers are sometimes used along with optimizers to achieve a certain objective or set of objectives given a constraint acting as heuristic (Hashem et al., 2018).

Users expectations, whether they are end users or business owner, are only drastically getting higher. The massive data flow, the enormous data size and the immediate need to get an answer, feedback or an insight make it critical to keep constantly working on doing things better, faster and more accurately. Hence our motive for this work, we highlight the current algorithms for job scheduling along with their drawbacks and strength points, to help find a new algorithm or a hybrid of existing algorithms, acting as a unified ecosystem addressing the issues that are considered to be a major drawback for other algorithms.

This survey is organized into five chapters. This chapter provides an introduction along with the motivation, problem definition and the contribution. The remaining of the document is organized as follow, Method, Background along with essential basics are presented in the second chapter. In the third chapter we thoroughly discuss the most popular Job Scheduling algorithms, and a crucial comparison for the context. The fourth chapter we present use cases for different algorithms to emphasize the distinguishing points. Finally, in chapter Five we conclude this survey paper.

2. Hadoop MapReduce Breakdown

Hadoop MapReduce can be considered as a complex ecosystem that needs to be analyzed in order to draw a clear picture for the job scheduling algorithms.

2.1 Hadoop and MapReduce

2.1.1 MapReduce

MapReduce framework was originally proposed by Google is a programming model, aims to provide a powerful com- putation model for large input data, by parallelize and distribute large- scale computations cross multiple machines on clusters, or cores of the same machine in a reliable, fault-tolerant manner (Dean & Ghemawat, 2008). Map reduce model based on two main user defined functions Map and Reduce, as in Figure 1, (Bakshi, 2018) Map phase is responsible for processing the input data to produce intermediate set of data represented as pairs, key-value pairs. The data is then sorted, and using the shuffling technique the resulting dataset is fed to the Reduce function, where data with same intermediate keys is merged, producing key-values pairs, that is the Reduce phase (Dean & Ghemawat, 2010).

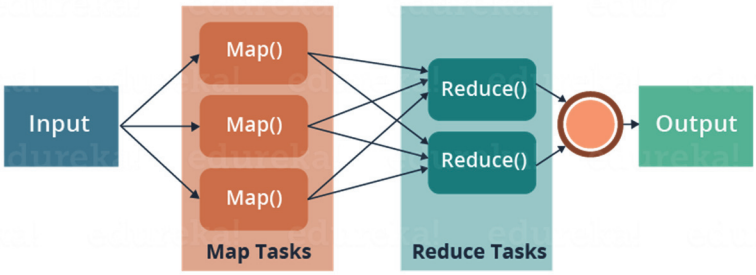

Figure 1. Map Reduce

2.2 Hadoop

Apache Hadoop, is an open source platform that gives the ability to easily implement distributed processing that is highly reliable and scale-able, while working with massive datasets, interpreted using programming models that are relatively simpler than the data itself. Hadoop is primarily based on clusters composed of commodity/low-cost computers, pro- viding feasible solutions for storing and processing large amounts of data with no constraints on the format or the struc- ture (”Apache Hadoop” , n.d.) Hadoop has four main components: Hadoop Common, Hadoop Distributed File System (HDFS), Hadoop YARN and Hadoop MapReduce. Hadoop Common is a collection of utilities supporting other Hadoop modules running on top of HDFS that is a highly efficient, fault-tolerant and low cost distributed file system design running on hardware best described as commodity, to store huge data across multiple machines. Hadoop YARN is the resource management layer on Hadoop, responsible for job scheduling and cluster resources management. Hadoop MapReduce is an implementation of the MapReduce programming model that provides a highly scalable solution for large-scale data processing (Kulkarni &

Khandewal, 2014).

2.3 Hadoop MapReduce Architecture

Hadoop Map reduce is an implementation of Map reduce programming model with managed fault tolerance support, with the intent to simplify and automate programs' parallel processing on vast amount of data. Figure 2, shows a typical Hadoop MapReduce cluster integration architecture, which consists of two types of nodes, Master node and Worker node (Master/Slave architecture). Master node represented by the Resource Manager that is responsible for the management and coordination between jobs and clusters, Data Node is represented by the Application Master, which is responsible for job coordination, and the Node Manager for managing nodes local resources and launch containers (Vavilapalli et al., 2013).

Once a new task is submitted by the client it is redirected to the master node which in turn launches the Application Master (AM) as a process on one of available Data Nodes. The Application Master communicates the resources required to the Resource Manager in order to successfully handle and run both phases; Map and Reduce phases. The Resource

Manager allocates the requested resources to containers, then the application master launches the task on the container. The resulting output of the tasks is sent to the Application Master which is then sent back to the Master Node (Bressoud & Tang, 2016).

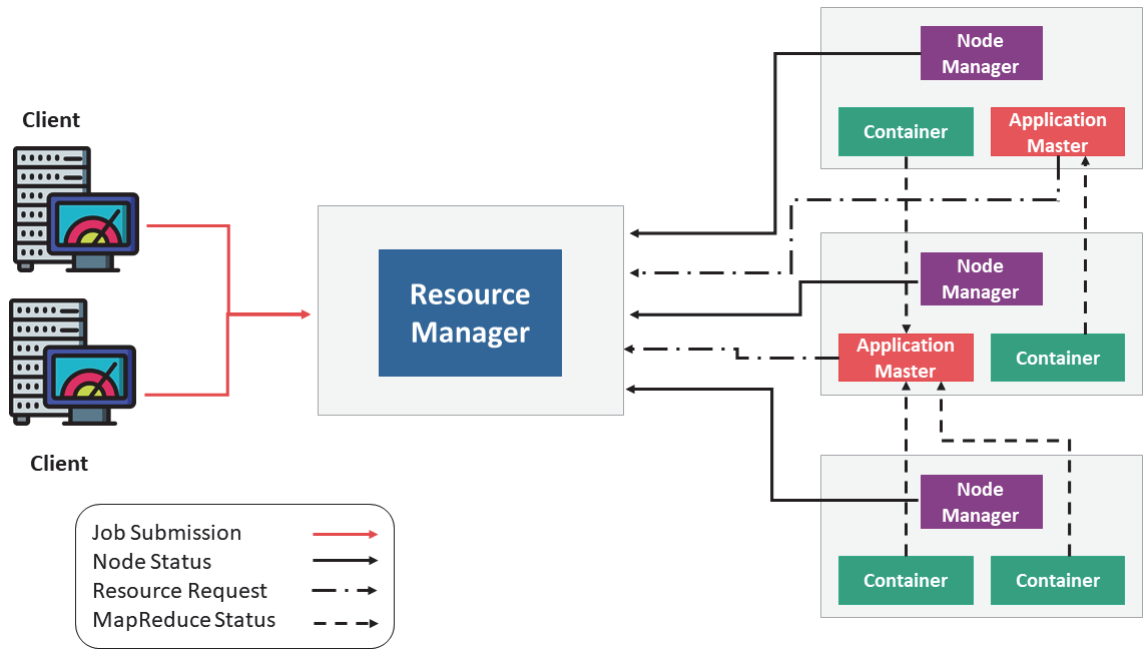

Figure 2. Hadoop Map Reduce

2.4 Hadoop MapReduce Job Scheduling

The adopted programming model in Hadoop MapReduce is parallel programming within a distributed homogeneous cluster environment. The data and the processing are considered to be distributed while the processing is run on parallel. To support this paradigm, scheduling is a necessity to this model.

Scheduling has been a persistent issue for a long time in various systems and clusters, the need to run the tasks and allocate the resources to said tasks summarizes the idea behind Job Scheduling, which has been and continuous to be an area of interest in the research field(Mohamed & Hong, 2016). The main goal and desired contribution behind this interest is to tackle the issues facing job scheduling.

Next, we discuss the most popular scheduling considerations that the scheduling algorithms are trying to meet:

locality, synchronization and fairness (Yoo & Sim, 2011).

2.4.1 Locality

It is the ability to direct the process or task to the data node holding the data needed to complete that task. This consider- ation is important as it aims to save resources (I.e. bandwidth, networking costs), it does as well eliminate the presence of bottlenecks in the ecosystem (Mohamed & Hong, 2016).

2.4.2 Synchronization

This consideration is where we identify the reduce phase as dependent on the map phase due to the fact that in MapReduce model, the output of the map task is fed into the reduce task as input, this process is subject to latency which can lead to delay in the whole reduce sub-task (Yoo & Sim, 2011).

2.4.3 Fairness

This consideration describes the technique upon which allocating multiple users to multiple resources in a fair manner is accomplished. New users and new tasks are taken into consideration once a heart-beat is detected (Yoo & Sim, 2011).

3. Job Scheduling Algorithms

3.1 Default FIFO Scheduler First-In-First-Out Scheduler, the Mechanism Adopted within This Scheduler Assumes a Job

queue, serving the first queued job. This scheduler is one of the most popular, it was popular with Hadoop as well, and FIFO was the default scheduler for Hadoop MapReduce. FIFO is considered to be non-preemptive, with no regard to order or priority, FIFO serves the first job queued. FIFO is considered to be best suited for execution of jobs where the order is not a key factor or not important. Presumably, if a short task that is requiring the least resources allocation falls last in a queue, it will remain pending until it reaches the first rank in the queue (Jone, 2011).

3.2 Fair Scheduler

The fair scheduler technique is the technique that assigns the resources available to the jobs in a manner that ensures fairness or exploiting resources during the entire processing time. The jobs are grouped into pools, and each pool has its share of the resources. This technique ensures that each user get a part of the processing capabilities. If there is a single task to be processed it is assigned all the resources hence it supports parallelism, once a new task is to be processed it receives a slot of the resources. (”Hadoop: Fair Scheduler”, 2017)

3.3 Capacity Scheduler

Capacity scheduler is a pluggable scheduler, it can be integrated within Hadoop ecosystem. This scheduler uses a technique by which multiple users or tenants are assigned shared resources or share a significantly big cluster in a way that ensures that all the cluster is exploited to users’ applications (”Hadoop: Capacity Scheduler”, 2017).

The main characteristics distinguishing Capacity Scheduler are the way queues are laid out, and the way resources are allocated to said queues. It ensures the shared cluster capacity among users rather than jobs (Rani, Rana, & Barjtya, 2015).

3.4 Delay Scheduler

Delay Scheduler is a scheduler that is configured to wait for configurable, acceptable amount of time in a try to allocate a container to the requested node to achieve locality and the ultimate goal is using the full potential of a cluster(Gautam et al., 2015), it addresses both primary job scheduling considerations; locality and fairness.(Zaharia et al., 2010)

3.5 Matchmaking Scheduler

Matchmaking Scheduler is best suited for MapReduce cluster, the goal of the main technique is to enhance locality rate of the data at the ”Map” Phase.

When the rate of data locality is low, the rate of moving data within the cluster and among other nodes increases, resulting in undesirable traffic and potentially bottlenecks. This techniques targets data locality rate taking into consideration an acceptably low response time in the MapReduce clusters.

In order for this technique to work, tasks need be assigned to nodes, mapping the tasks locally is preferred over mapping the tasks globally (Not a local mapping). The job to which the task belongs is not of importance to this process, one key aspect to notice, though, is the use of a marker to track locality within the nodes, and most importantly to guarantee that the nodes are assigned their tasks (locally preferred over globally) (He, Lu, &

Swanson, 2011).

3.6 LATE: "Longest Approximate Time-to-end Scheduler"

The technique behind this scheduler aims to complete all the tasks within the same job as fast as possible and are consid- ered of higher priority as the entire job will be marked completed faster and is de-queued, making the next job and its tasks the new priority. In the case of a slow running task, a new copy of this task is started using different resources which could result in faster completion pace hence an enhanced performance. Within this technique, delay is always considered as execution problem, it does not seek a diagnostic solution for the faulty node causing the delay, it just creates a speculative copy (Mohamed & Hong, 2016).

3.7 Deadline Constraint Scheduler

In Deadline Constraint Scheduler, scheduling the jobs is based on deadline constraint provided by user, the user is imme- diately notified of whether the submitted job could be completed within the given deadline or not, This scheduler makes sure to meet the deadlines (Kc, & Anyanwu, 2010), having a different deadline for the job does not make any difference, the technique remains the same. Two data processing models used to achieve the objectives of the scheduler, job execution cost model and constraint-based Hadoop scheduler (Chen, Lin, & Kuo, 2018).

3.8 Resource Aware Scheduler

Resource Aware Scheduler technique seeks to utilize the resources (network, disk, CPU) within the cluster. Instead of applying the concept of resource contention on the computing devices used by some of the previously mentioned techniques: FIFO, Capacity and Fair scheduler. Resource aware scheduler, on the other hand, merely demonstrates the strengths of resources utilization in case of having varying types of workloads or tasks running on a cluster (Yong, Garegrat, & Mohan, 2009). The RSA consists of two main activities arise: The first of which being Dynamic Free Slot Advertisement Activity, this is an alternative to fixing the number of the slots used for computations. With the use of the second activity that is User Free Slot Filtering/Prioritizing, were the resources metrics are retrieved from the nodes separately, this number is calculated dynamically. It could be rather expressed as Filtering for Advertising. (Kapil & Kamath, 2013)

3.9 COSHH: " A classification and optimization based scheduler for heterogeneous Hadoop systems"(Rasooli & Down, 2014)

COSHH is a Multi-objective Hadoop job scheduler, considering the different scales on both clusters resources and jobs, to improve jobs average completion time with competitive levels of fairness, locality and minimum share satisfaction measurements(Rasooli & Down, 2014). The technique was designed to support heterogeneous environments on both levels, application level and cluster level. The primary idea behind this implementation is to use the available data in the system to make better scheduling decisions which ultimately does enhance the performance. The technique depends on the presence of two operations, each of which receives a message that is known as the heartbeat message. Once a job is queued by the scheduler, a heartbeat is received that consequently lead to triggering the proper routing to assign free resource to the queued job, as a result the mean completion time of the job is decreased (Nagina & Dhingra, 2016)

3.10 MOMTH:"Multi-objective scheduling algorithm of many tasks in Hadoop" (Nita, Pop, Voicu, Dobre, & Xhafa, 2015)

MOMTH is a Multi-objective/Multi-constrains Hadoop job scheduling approach, aims to improve the processing of big data. MOMTH algorithm takes into consideration two objectives, first one of which is an optimal workload of the cluster and the second objective is avoiding resources contention. With respect to two constraints, for example deadline and budget. This Approach provides a performance measurements that are similar to the ones of FIFO and Fair schedulers (Nita et al., 2015). As a general rule the two objectives are users' related objective and resources related objective (Nagina & Dhingra, 2016).

4. Job Scheduling Algorithms Comparison

4.1 Job Scheduling Classification

Job Scheduling algorithms can be analyzed from different angles, jobs priority, allocation strategy, resource awareness to name a few. The work in this section is a comparison conducted based on the following criteria: Target, Job Allocation Strategy and Optimization Strategy.

Job Allocation Strategy

4.1.1 Static Scheduling (Offline Scheduling)

The strategy according to which static scheduling algorithms allocate jobs to the processors is at compilation time and before the beginning of the processing, where all the needed information is already defined and known (Mohamed & Hong, 2016). The main advantage for using this type of algorithms is the minimal execution time of the running jobs (Gautam et al., 2015). And resources contention could be considered as this strategy's drawback.

4.1.1.1 Dynamic Scheduling (Online Scheduling)

In Dynamic Scheduling Strategy, all the jobs allocated to the processors during program execution time also known as online scheduling. By using this strategy, it is assumed that the scheduler has little knowledge about the resources, and the environment within which the jobs are to be executed is unknown (Mohamed & Hong, 2016), the dynamic behavior came from the scheduler's ability to dynamically adopt new jobs or handle the changes regarding the processors' availability (Gautam et al., 2015). This strategy, unlike static allocation strategy, supports resource sharing.

4.1.1.2 Optimization Strategy

This strategy's main goal is to find, minimize or maximize a preset objective, below is a listing along with a brief descrip- tion of different optimization strategies.

4.1.2 Single Objective

Single objective optimization focal point is finding the optimal solution that minimizes or maximizes a single predefined objective, which is the response time, locality, fairness or CPU time (Hashem et al., 2018). Specifying the requirements plays a critical role in determining the required optimization approach. Setting objectives apart from constraints is con- sidered a major factor in specifying the main parameter, one way to achieve this is by testing the parameter, if it needs to have a specific value it is an objective, whereas, if a parameter needs to be less or greater than a value then it is considered to be a constraint(Nita et al., 2015).

In Table 1 we run a detailed comparison among the aforementioned algorithms on the criteria that is specified as: Objec- tive, Job Allocation, and Optimization Strategy.

Table 1. Job Scheduling Comparison

Algorithm	Target	Job Allocation	Optimization Strategy
FIFO Scheduler	Minimize jobs execution time	Static	Single Objective
Fair Scheduler	Fair resource sharing among jobs	Static	Single Objective
Capacity Scheduler	Fair resource sharing among users	Static	Single Objective
Delay Scheduler	Achieving high data locality with low impact on fairness	Static	Single Objective
Matchmaking Scheduler (He & Swanson, 2011)	Improve data locality and the response time	Static	Single Objective
LATE Scheduler (Zaharia, Konwinski, Joseph, Katz, & Stoica, 2008)	Improve response time	Static	Single Objective
Deadline Constraint Scheduler	Meet jobs Deadlines	Dynamic	Single Objective
Resource Aware Scheduler	Improve resource utilization	Dynamic	Single Objective
COSHH Scheduler	Improve completion time considering Fairness and locality	Dynamic	Multi-Objective
MOMTH Scheduler	Achieve optimal workload and avoiding resource contention	Dynamic	Multi-Objective

Discription: Algorithms' Comparison among different criteria: Target, Job Allocation Strategy and Optimization

Strategy

4.1.3 Multi-objective

Multi-objective optimization focal angle is on finding the optimal solution for procedures that are managing multiple objectives (Hashem et al., 2018). Many Multi-objective problems considered as a single objective optimization problem by setting one objective and convert the other remaining objectives to constraints (Nita et al., 2015).

4.2 Hadoop Schedulers Use Cases

The relevance of Hadoop systems has now moved from long running batch jobs to interactive ad-hoc queries, graph handling, AI applications, and so forth, where the selection of the scheduler is dependent on the criteria of the environment to achieve the desired results/objectives and achieve the maximum benefit. Job Scheduling selection must comply with heterogeneity, dynamicity of resources and variety of applications requirements that takes into consideration the key factors contributing to the performance (Zaharia et al., 2008) in order to obtain the optimal scheduling for the resources of the cluster.

FIFO Scheduler is used when the order of the jobs' execution is not important and the job size is small. It might not be the best choice if the size of the jobs is relatively larger or if the jobs are varying in size, this could excessively abuse the computing power on the cluster, and cause large delays for other jobs(Tang, Liao, 2018) as the shorter jobs will have to wait until longer job are processed if they appeared earlier in the queue, FIFO scheduler is not suitable for interactive jobs or shared clusters as no preemption is made, unlike Fair and Capacity techniques, that have equal utilization of resources(Cheng, Rao,Jiang & Zhou, 2015), the fair scheduler provides faster response to small jobs than larger ones and can perform well in both small and large clusters.

Capacity scheduler guarantees the exploitation of the unused capacity for the jobs within queues, its usefulness radiates along with the presence of multiple clients within a large Hadoop cluster (Usama, Liu, & Chen, 2017). It provides a simple approach to commonly utilize a cluster among different multiple users.

The Resource Aware Scheduler and LATE scheduling, both, have in common is that they dont achieve the locality issue. LATE Scheduling is usually deployed when the minimal jobs response time and the optimality of the Jobs performance are the two main aspects to consider, it is highly powerful for heterogeneous environments. Locality, though, is not a key aspect that is tackled using the LATE Schedulers, as its mere goal is to have the job done as fast as possible with no visible delay.

The Resource Aware Scheduler is best deployed when the resources utilization is the key aspect while working with multi job workloads nodes (Jord et al., 2011). It supports an architecture that is customizable/adapting at runtime depending the vastly developing demands within a Heterogeneous Hadoop Cluster. When locality and fairness are a must have and a minor delay is acceptable (Zaharia et al., 2010), Delay Job Scheduling is the ideal choice as its main goal is to address locality consideration, with the assumption: the shorter the jobs the better, for the longer jobs the delay could be undesirably high ("Page survey on job schedulers in hadoop cluster", 2013). It suggests to forcibly impose a delay on the execution of the next job, coming from the same user, for a configurable amount of time in a try to have the job executed on the node that executed the users last job (Yoo & Sim, 2011). The Delay scheduler is implemented by Facebook for the lowest recorded response time for map task. Similarly, Matchmaking Scheduler handles the locality issue but it is best deployed among MapReduce cluster ("Performance evaluation of job schedulers on Hadoop YARN", 2016) (Mapreduce scheduler: A 360-degree view, 2016).

All of the previously showcased algorithms are serving a single objective, but as the requirements are getting more sophisticated, the need to achieve more than one objective has emerged. Any number of combinations could be developed, merging the most important factors to answer to the business needs at hand. There are plenty of examples for multi- objective, multi constraint combinations all answering to customized needs. MOMTH, for example, has the objectives to achieve optimal workloads of the cluster and to avoid resources contention. And has deadline and budget as its constraints. It is mostly appropriate in a cluster that is classified as Heterogeneous answering to users and resources limitations. COSHH, is a different implementation of the same concept, it is best used with a heterogeneous environment that supports two levels of heterogeneity: application level and cluster level (Rasooli & Down, 2014).

Table 2 demonstrates use cases for each of the algorithms discussed throughout this work. The table compares and describes the environment within which the algorithms are best run, The implementation of the algorithms aspire to comply with some quality standards based on the specifics of the solution being developed and the objectives need to be met, use cases are described using some of these quality standards: Heterogeneity Support, Locality, Fairness, Response Time, Execution Time, Resources Utilization and Ease of Configuration.

Heterogeneity Support(Usama et al., 2017) in Hadoop cluster environments is classified to Homogeneous and Heteroge- neous, according to the availability of the resources on each node, if the nodes among the cluster share the same resource specifications such as CPU cores and network bandwidth, this means the cluster environment is Homogeneous, and nodes performance rates are forecasted to be the same, otherwise, if the environment is Heterogeneous then special job schedul- ing algorithms need to be used. Locality (Murali Brindha, 2018) (Patel, 2015),is a task-data localization measure, it ensures that the data is accessed from a local node during task processing. Fairness (Murali Brindha, 2018), is a measure of schedulers fairness in distributing jobs among available resources. While Response Time (Murali Brindha, 2018) (Pati, Mehta, 2015), is the time needed from the moment the request is submitted until it receives the first response, minimizing the response time enhances schedulers effectiveness. Execution Time (Murali Brindha, 2018) (Pati, Mehta, 2015), is the time the job actually spends in executing and using processor resources. Resources Utilization (Seethalakshmi, Govin- dasamy, Akila, 2018), is about making the better use of available resources to achieve the desired objective. Ease of Configuration, this area describes the efforts needed to integrate/plug the algorithm into the ecosystem, it is considered to be easier for the single objective algorithms, as those algorithms are either built in or ready to plug-in. On the other hand, multi objective algorithms are mostly external tools that needs extra work to integrate.

Table 2. Algorithm Areas of Applicability

Algorithm	Hetero- geneity	Locality	Fairness	Response Time	Execution Time	Resource Utiliza- tion	Ease of Config- urability
FIFO Scheduler (Rasooli & Down, 2012)	X	X	X	X	√	X	√
Fair Scheduler (Srivastva, Singh, & Ahmad, 2017)	X	X	√	√ [For small jobs]	X	√	√
Capacity Scheduler	X	X	√	√ [For small jobs]	X	√	√
Delay Scheduler	X	√	√ [Less than Fair Sched- uler]	√ [For small jobs]	X	X	√
Matchmaking Scheduler	X	√	√	√ [Average Re- sponse Time]	X	√	√
LATE Scheduler	√	X	X	√	√	√	√
Deadline Constraint Scheduler	√	X	X	X	√	√	√
Resource Aware Scheduler	√	X	X	√	X	√	√
COSHH Scheduler	√	√	√	X	√	√	X
MOMTH Scheduler	√	√	√	√	X	√	X

Discription: Best Environments/Setup to run each algorithm

5. Conclusion

In this work, we tackle the topic of: "Job Scheduling Algorithms in Big Data Hadoop Environment". A Job scheduling Algorithm can be considered as a great asset contributing to the success of the whole ecosystem and achieving a predefined" high performance" while performing within big data environment, it is an essential base step for realizing optimality in the utilization of cluster resources. We discuss the most popular scheduling considerations that the scheduling algorithms are trying to meet: locality, synchronization and fairness. We, as well, discuss Hadoop schedulers based on various eval- uation metrics such as locality, response time and fairness. Our work discusses how different Job Scheduling Algorithms help in gaining better outcome in Hadoop cluster by showcasing use cases, and a result we believe, to achieve optimality a hybrid of algorithms is the solution to handle different aspects and problems faced.

References

Amir, G., & Murtaza, H. (2015). Big data concepts, methods and analytics. *International Journal of Information Management, 35*, 140. https://doi.org/10.1016/j.ijinfomgt.2014.10.007

Apache Hadoop. Retrieved from https://www.ibm.com/analytics/hadoop Apache Software Foundation, (2017). *Hadoop: Capacity Scheduler*. Retrieved from https://hadoop.apache.org/docs/r2.7.4/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html

Apache Software Foundation. (2017). *Hadoop: Fair Scheduler*.Retrieved from https://hadoop.apache.org/docs/r2.7.4/hadoop-yarn/hadoop-yarn-site/FairScheduler.html.

Bhadani, A. K., & Jothimani, D. (2016). *Big data: challenges, opportunities, and realities. In Effective Big Data man- agement and opportunities for implementation* (pp. 1-24). IGI Global. https://doi.org/10.4018/978-1-5225-0182- 4.ch001

Big Data and Hadoop. (2018). Ashish Bakshi Retrieved from www.edureka.coblogmapreduce-tutorial#what is mapreduce IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 15, Issue 1 (Sep. -Oct. 2013), PP 46-50 www.iosrjournals.org www.iosrjournals.org 46 — Page *Survey on Job Schedulers in Hadoop Cluster* Bincy P Andrews1 , Binu A2 1 (Rajagiri School of Engineering and Technology, Kochi.

Bressoud, T. C., & Tang, Q. (2016, December). *Analysis, modeling, and simulation of Hadoop YARN MapReduce*. In 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS) (pp. 980-988). IEEE. https://doi.org/10.1109/ICPADS.2016.0131

Chen, C. H., Lin, J. W., & Kuo, S. Y. (2018). MapReduce scheduling for deadline-constrained jobs in heterogeneous cloud computing systems. *IEEE Transactions on Cloud Computing, 6*(1), 127-140. https://doi.org/10.1016/j.dcan.2017.07.008

Cheng, D., Rao, J., Jiang, C., & Zhou, X. (2015). Resource and deadline-aware job scheduling in dynamic hadoop clusters. Proc. IEEE Int. Symp. Parallel Distrib. Process., pp. 956-965, 2015.

Das, R., Singh, R. P., & Patgiri, R. (2016). Mapreduce scheduler: A 360-degree view. *International Journal of Current Engineering and Scientific Research (IJCESR), 3*(11), 88100.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM, 51*(1), 107-113. https://doi.org/10.1145/1327452.1327492

Dean, J., & Ghemawat, S. (2010). MapReduce: a flexible data processing tool. *Communications of the ACM, 53*(1), 72-77. https://doi.org/10.1145/1629175.1629198

Dongjin, Y., & Kwang, M. S. (2011). *A comparative review of job scheduling for MapReduce.* CCIS, Sept 2011.

Gautam, J. V., Prajapati, H. B., Dabhi, V.K., & Chaudhary, S. (2015, March). *A survey on job scheduling algorithms in Big data processing.* In 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT) (pp. 1-11). IEEE. https://doi.org/10.1109/ICECCT.2015.7226035

Hashem, I. A. T., Anuar, N. B., Gani, A., Yaqoob, I., Xia, F., & Khan, S. U. (2016). MapReduce: Review and open challenges. Scientometrics, *109*(1), 389-422. https://doi.org/10.1007/s11192-016-1945-y

Hashem, I. A. T., Anuar, N. B., Marjani, M., Ahmed, E., Chiroma, H., Firdaus, A., ... Gani, A. (2018). *MapReduce scheduling algorithms: a review*. The Journal of Supercomputing, 1-31.

Hashem, I. A. T., Anuar, N. B., Marjani, M., Gani, A., Sangaiah, A. K., & Sakariyah, A. K. (2018). *Multi-objective scheduling of MapReduce jobs in big data processing. Multimedia Tools and Applications*, *77*(8), 9979-9994.

https://doi.org/10.1007/s11042-017-4685-y

He, C., Lu, Y., & Swanson, D. (2011, November). *Matchmaking: A new mapreduce scheduling technique*. In 2011 IEEE Third International Conference on Cloud Computing Technology and Science (pp. 40-47). IEEE. https://doi.org/10.1109/CloudCom.2011.16

International Journal of Engineering and Computer Science, 6(9), 22457-22462.

Jone, M. (2011). *Scheduling in Hadoop*, Retrieved from https://developer.ibm.com/articles/os-hadoop-scheduling/

Jord, P., Claris, C., David, C., Yolanda, B., Ian, W., Malgorzata, S., Jordi, T., Eduard, A. (2011). *Resource-aware adaptive scheduling for mapreduce clusters*, Proceedings of the 12th ACM/IFIP/USENIX international conference on Middleware, December 12-16, 2011, Lisbon, Portugal. https://doi.org/10.1007/978-3- 642-258213 10

Kapil, B. S., & Kamath, S. S. (2013). *Resource aware scheduling in Hadoop for heterogeneous workloads based on load estimation*. In 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT) (pp. 1-5). IEEE.

Kc, K., & Anyanwu, K. (2010, November). *Scheduling hadoop jobs to meet deadlines*. In 2010 IEEE Second International Conference on Cloud Computing Technology and Science (pp. 388-392). IEEE.

Kulkarni, A. P., & Khandewal, M. (2014). Survey on Hadoop and Introduction to YARN. *International Journal of Emerging Technology and Advanced Engineering, 4*(5), 82-87.

Mohamed, E., & Hong, Z. (2016, November). *Hadoop-MapReduce job scheduling algorithms survey*. In 2016 7th Interna- tional Conference on Cloud Computing and Big Data (CCBD) (pp. 237-242). IEEE. https://doi.org/10.1109/CCBD.2016.054

Murali, J. A., & Brindha, T. (2018, April). *Analysis of Scheduling Algorithms in Hadoop*. In International Conference on Soft Computing Systems (pp. 25-34). Springer, Singapore. https://doi.org/10.1007/978-981-13-1936-5 3

Nagina, D., & Dhingra, S. (2016). Scheduling algorithms in big data: a survey. *Int. J. Eng. Comput. Sci, 5*(8), 17737- 17743.

Nita, M. C., Pop, F., Voicu, C., Dobre, C., & Xhafa, F. (2015). MOMTH: multi-objective scheduling algorithm of many tasks in Hadoop. *Cluster computing, 18*(3), 1011-1024.

Patel, H. M. (2015). A comparative analysis of MapReduce scheduling algorithms for Hadoop. *Int. J. Innov. Emerg. Res.Eng, 2*(2)

Pati, S., & Mehta, M. A. (2015, June). *Job aware scheduling in Hadoop for heterogeneous cluster*. In 2015 IEEE Interna- tional Advance Computing Conference (IACC) (pp. 778-783). IEEE. https://doi.org/10.1109/IADCC.2015.7154813

Performance evaluation of job schedulers on Hadoop YARN, J. -C. Lin, M. -C. Lee, *"Performance evaluation of job schedulers on Hadoop YARN"*. Concurrency and Computation: Practice and Experience, 28(9), 2711-2728. https://doi.org/10.1002/cpe.3736

Rasooli, A., & Down, D. G. (2012, November). *A hybrid scheduling approach for scalable heterogeneous hadoop systems*. In 2012 SC Companion: High Performance Computing, Networking Storage and Analysis (pp. 1284-1291). IEEE. https://doi.org/10.1109/SC.Companion.2012.155

Rasooli, A., & Down, D. G. (2014). COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems. *Future generation computer systems, 36,* 1-15. https://doi.org/10.1016/j.future.2014.01.002

Seethalakshmi, V., Govindasamy, V., & Akila, V. (2018, March). *Job Scheduling in Big Data-A Survey*. In 2018 Inter- nat2018 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC) ional conference on computation of power, energy, Information and Communication (ICCPEIC) (pp. 023-031). IEEE.

Senthilkumar, M., & Ilango, P. (2016). A survey on job scheduling in big data. *Cybernetics and Information Technologies, 16*(3), 35-51. https://doi.org/10.1515/cait-2016-0033

Srivastva, P., Singh, H. K., & Ahmad, S. (2017). Job Attentive Scheduling Algorithm in Hadoop. *International Journal, 7*(4).

Tang, X., & Liao, X, (2018). *Application-aware deadline constraint job scheduling mechanism on large-scale computational grid*. PLoS ONE 13(11): e0207596. https://doi.org/10.1371/journal.pone.0207596

Usama, M., Liu, M., & Chen, M. (2017). Job schedulers for big data processing in Hadoop environment: Testing real-life schedulers using benchmark programs. *Digit. Commun. Netw., 3*(4), 260-273. https://doi.org/10.1016/j.dcan.2017.07.008

Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., ... & Saha, B. (2013, October). *Apache hadoop yarn: Yet another resource negotiator*. In Proceedings of the 4th annual Symposium on Cloud Computing (p. 5). ACM. https://doi.org/10.1145/2523616.2523633

Yong, M., Garegrat, N., & Mohan, S. (2009, December). *Towards a resource aware scheduler in hadoop*. In Proc. ICWS (pp. 102-109).

Yoo, D., & Sim, K. M. (2011, September). *A comparative review of job scheduling for MapReduce*. In 2011 IEEE Interna- tional Conference on Cloud Computing and Intelligence Systems (pp. 353-358). IEEE. https://doi.org/10.18535/ijecs/v6i9.11

Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., & Stoica, I. (2010, April). *Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling*. In Proceedings of the 5th European conference on Computer systems (pp. 265-278). ACM.

Zaharia, M., Konwinski, A., Joseph, A. D., Katz, R. H., & Stoica, I. (2008, December). *Improving MapReduce performance in heterogeneous environments*. In Osdi (Vol. 8, No. 4, p. 7).

Copyrights

Copyright for this article is retained by the author(s), with first publication rights granted to the journal.

This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).