Revue d'Intelligence Artificielle
Vol. 38, No. 6, December, 2024, pp. 1391-1397
Journal homepage: http://iieta.org/journals/ria

Analysis of the MapReduce Performance in Hadoop

Nour-Eddine Bakni [1*], Ismail Assayad [2]

[1] LIS Lab, Faculty of Sciences, University Hassan II of Casablanca, Casablanca 20100, Morocco
[2] LIS Lab, Faculty of Sciences, ENSEM, University Hassan II of Casablanca, Casablanca 20100, Morocco

Corresponding Author Email: ne.bk.info@gmail.com

https://doi.org/10.18280/ria.380601

Received: 22 June 2024
Revised: 12 October 2024
Accepted: 1 November 2024
Available online: 21 December 2024

Keywords:
big data, job scheduling, Hadoop, MapReduce

ABSTRACT

The need for Big Data platforms in recent years is increasing steadily, given the amount of data produced or consumed every second by millions of users and machines, and this huge volume of data has to be processed, managed, or stored. Several constraints must be taken into consideration when allocating this data and processing it on big data platforms, and among the major concerns of big data clients who are always looking to reduce their costs remains time and budget. We can say that time is among the major factors that determine the performance of a processing model of a big data platform and which has a direct effect on other allocation constraints. In this paper, we conducted an analytical study of the performance of MapReduce which is the processing model of the Hadoop platform. Our study shows that the estimation of MapReduce performance remains difficult and depends not only on the scheduler used but also on other factors including the type of workload itself.

1. INTRODUCTION

Big data includes all the technologies and utilities that constitute a solution for huge data and an alternative to the traditional database. It refers to processing and analyzing a complex and large data set to uncover valuable information that can be useful for businesses and organizations. It is an IT solution that offers big data management, processing and storage, distributed computing, analytic tools, and data infrastructure, it further promotes custom development, queries, and systems scalability. The use of big data is widespread in many fields due to its advantages compared to traditional databases, especially for large amounts of data which put more pressure on clusters and data management systems. A major concern for the clients of big data platforms in general is how to estimate the processing time of their workloads, which allows them to predict an approximate allocation budget on a particular platform, and helps them to make a good decision by selecting the most suitable platform for their workloads based on the time/budget ratio, or in another sense, knowing in advance the execution time of the workloads will allow the clients to deduce an approximate allocation budget on a given platform, and to make cost comparisons between platforms to choose the one that provides a reduced allocation cost or the one offering the best cost/performance compromise. However, scheduler performance may be subject to various factors that may have an effect on the processing time of workloads.

Given the importance of big data and the need for it, especially in recent years, if we take into account the large volumes of data that need to be processed or stored, and also the difficulty of managing data of this magnitude via traditional systems, the number of studies and research conducted in this field is constantly increasing, the objective is always to provide studies or solutions that can contribute to improve the data processing and storage at the Big Data platform level, and to increase systems effectiveness in different sides.

In this context, our work is an analytical study of the performance of MapReduce in processing different types of workloads on Hadoop Framework, and how the following factors can have an effect on the performance of schedulers in Hadoop MapReduce:

- The scheduler used.
- The nature of workloads.
- The setup configuration (CPU speed, Memory Size, Number of slots, …).

The objective of our work is to give a performance study of MapReduce and the factors that affect it, the contributions of the study can be summarized as follows:

- Identify the main factors that affect the processing time of workloads on Hadoop MapReduce, and being aware of these factors that can affect the efficiency of the processing model can help to improve the performance of Big Data platforms.
- Show that the adaptation between the scheduler used and the workloads can have an impact on the processing time, and therefore on the necessary allocation budget, which can be useful to big data clients to make good decisions regarding the allocation of their workloads.
- Provide a study that can be useful to researchers in this field of big data or that can serve as a basis for further studies, the aim of which is always to improve the performance of the Hadoop system.

The results of our study have shown that the accurate estimation of the actual processing time for a given workload remains apparently difficult and depends not only on the scheduler used, but it differs according to many factors including the type of the workload itself.

The organization of the remaining sections is as follows. In Section 2, we reveal the background and present the related work. In Section 3, we introduce our experiments and detail their environment. In Section 4, we present and discuss the results obtained. Finally, in Section 5, we conclude the paper and highlight some future work.

2. BACKGROUND AND RELATED WORK

In this section, we provide an overview of the Hadoop Framework, and MapReduce processing model, and discuss the related work.

2.1 Hadoop

Apache Hadoop is an open-source software framework for distributed processing and data intensive management based on Java, used to process and analyze huge amounts of data. It allows big data analytical processing tasks to be divided into smaller tasks that can be executed in parallel mode using a processing model (such as the MapReduce model) and distributed across a Hadoop cluster. Knowing that this kind of cluster is a collection of computing machines called nodes, networked to perform parallel calculations on large sets of data. Unlike other computing clusters, Hadoop clusters are specifically designed to host and analyze massive volumes of structured and unstructured data in a distributed computing environment.

Hadoop ecosystems include open-source software frameworks and a range of complementary tools for big data. They are also distinguished from other types of computing clusters by their specific structure and architecture. Hadoop clusters consist of a network of master and subordinate nodes that leverage high-availability, low-cost generic hardware, as well as linear scalability, so that it is easy to add or remove nodes depending on demand. The master nodes orchestrate and manage the resources, worker nodes are dedicated to processing tasks and storing data. Master nodes include a NameNode, a secondary NameNode, and a JobTracker. Worker nodes that offer the DataNode and TaskTracker services that store data and process tasks. The Hadoop architecture is described in Figure 1.

Hadoop is popular because it is both accessible and easy to learn. Economical and practical, it offers modules with many options. Hadoop can easily scale and be deployed across multiple machines to accommodate virtually any dataset size. Additionally, its method of storing and processing data presents an attractive enterprise solution in the context of growing storage, and also for its accessibility and flexibility in the use of the Hardware, especially when we are talking about the storage of a large volume of data which constitutes a real problem, because it entails high maintenance costs in terms of resources and hardware in order to be able to manage the workload. The Hadoop framework is resilient to failures. Data stored on one server is automatically replicated to other nodes. When one node in the Hadoop cluster fails, others take over to perform calculations or data analysis. Hadoop has various applications including data warehousing, big data analytics and cloud computing.

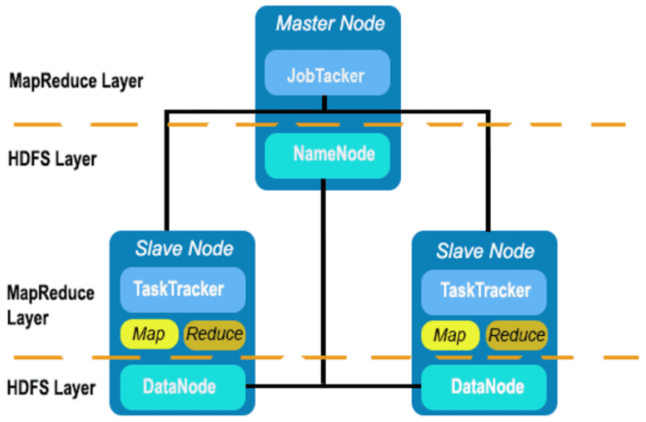

Figure 1. Hadoop architecture

2.2 MapReduce

MapReduce is a model for parallel processing of large quantities of data on multiple machines organized into a cluster, it's designed and deployed by Apache Foundation, it is a component of the Hadoop framework along with HDFS, the storage system, and YARN, the resource management module, and Hadoop Common, which includes utilities and libraries used and shared by other Hadoop modules. MapReduce contains two main phases, named map and reduce. The mappers take a set of data and convert it to a series of key-value pairs. Those intermediate data will then be sorted and merged. The reducers receive them as input and combine them into a smaller set of tuple pairs as the final output, and store it in the HDFS system. Among the advantages of MapReduce are its flexibility, security, fault tolerance, and its scalability in data processing.

Data processing in the MapReduce model goes in general through several successive stages one after another, and these phases are shown in Figure 2, which explains as an example the process of treating the WordCount application by MapReduce.

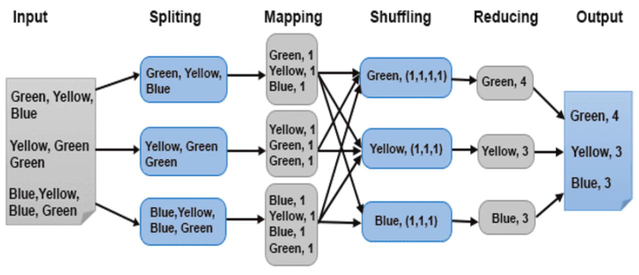

Figure 2. MapReduce processing example for Hadoop WordCount

- **Input Phase:** reading and translating a block of data to an input for the mappers.

As shown in the example in Figure 2, which allows us to count the number of occurrences of each word in the input file, we divide the input into three parts (Green, Yellow, Blue); (Yellow, Green, Green); (Blue, Yellow, Blue, Green).

- **Mapping Phase:** processing the input data by a mapper function and generating a sequence of key-value pairs.

Assign a hard-coded value (1) to each of the tokens or words. The reason we assign a hard-coded value of 1 is that each word, by itself, will appear once, and so for example, for the first line (Green, Yellow, Blue) we get 3 key-value pairs (Green, 1); (Yellow, 1); (Blue, 1).

- **Shuffling Phase:** It consists of merging and sorting the output of the mapping, and it produces a sequence of combined key-value pairs.

So, each reducer will receive a unique key and a sequence of values corresponding to that same key: Green, (1,1,1,1); Yellow, (1,1,1); Blue, (1,1,1).

- **Reducing Phase:** applying a Reducer function to the output of the shuffling phase, and returning a single output value. This result is stored in the HDFS system.

In this phase, each reducer counts the number of ones in the list it received, as shown in the figure, the first reducer for example receives the list green, (1,1,1,1) and returns the key/value green, 4.

YARN has greatly increased the potential use cases of the Hadoop framework by decoupling resource management and scheduling from MapReduce's data processing component. It also allowed Hadoop to support more applications and different types of processing. YARN architecture consists of two daemon services: the Resource Manager (Master) which is the master daemon (main program), It manages the allocation of resources such as CPU, memory and network bandwidth, and the Node Manager (Slave) which is the slave daemon (secondary program), and it reports the resource usage to the resource manager. These two daemons are responsible for negotiating resources and working jointly to execute and monitor tasks, and they also manage the parallel processing and fault tolerance components of all MapReduce tasks. Resource Manager and Node Manager combine together to form a data-computation framework.

Both HDFS and MapReduce run on the same set of nodes, resulting in very high overall bandwidth across the entire cluster. All Hadoop components are synchronized to enable improved utilization of the cluster.

2.3 Related work

Time is one of the important criteria that evaluates the performance of MapReduce and determines its efficiency, and has a significant influence on other allocation constraints including budget, and as we know, the budget remains a criterion of utmost importance to big data clients when allocating their workloads. Generally, in the domain of Big Data, being aware of the job timespan remains essential for later decisions, here are some researches that were done in this context:

In the study [1], the proposed model allows for predicting the processing time of a job and estimating the number of resources that a job requires to be finished within a deadline, the task execution history is the basis of this model which aims to estimate the job runtime using the locally weighted linear regression (LWLR) technique.

Paper [2] proposes a two-phase regression (TPR) method to estimate precisely the completion time of the executed tasks, an analysis report is used to give detailed data of each job. The TPR algorithm is based on the following steps: data preprocessing, data smoothing, data regression, and data estimation.

The research study given by Gohil et al. [3] shows experimental results of MapReduce performance with some applications in Cloud based Hadoop.

In the study [4], we present a new method that can predict the job runtime in the case where the job starts running for the first time with no history about it, by using mathematical expressions to formulate each stage of MapReduce process and calculate the job runtime, or when the job has previous running and its profile or history becomes available, in this case, we refer to the job's profile or history in the database in processing time estimation.

The model presented in the study [5] can estimate the total job runtime by using Amdahl's law regression method with respect to input size variation, this model requires code instead of a trace-base simulator. Also, the paper evaluates and analyzes the performance of WordCount application under variant types of processors.

Paper [6] presents a model for predicting the performance of Hadoop MapReduce, this model consists of two modules, a job analyzer and a prediction module, the first one collects information around processed jobs to serve the prediction module, and the second one will use this data to predict the job performance.

In the research [7], we give a benchmarking approach that allows us to build a MapReduce performance model which will attempt to estimate the procession time for a MapReduce workload by combining the job profiling and the derived platform performance model.

Paper [8] compares different job scheduling techniques and discusses their performance impact on Hadoop multi-cluster environment, and their efficiency in terms of certain hardware specifications.

In the study [9], we present a simulator called MRPerf that facilitates performance analysis for the MapReduce model, and it can be used as a planning tool for evaluation and fine-tuning Hadoop systems.

The performance study conducted by Jiang et al. [10] allowed us to determine the main factors that can impact the performance of MapReduce, and showed that by a careful setting of those factors we can improve the performance.

Paper [11] presents an analytical model to estimate the performance of MapReduce for a Hadoop workload, through a combination of a priority graph model and a queuing network model with the aim of predicting the processing time of tasks.

The benchmark suite for Hadoop called HiBench introduced by Huang et al. [12] brings together a set of Hadoop programs encompassing both synthetic microbenchmarks and real-world applications, it is used to assess and identify the performance of Hadoop workloads.

Paper [13] provides a detailed analytical study and performance characterization for Hadoop K-means using a set of processors.

In the study [14], we present a comprehensive analytical study of the impact of configuration settings on Hadoop performance in query processing.

We conducted an experimental study of the Hadoop system [15] using a set of input files. The study included the volume of data bytes read and written by Hadoop and MapReduce, and analyzed the behavior of the map and reduce functions as the number of files and the volume of bytes read and written increased.

In the research [16], we provide an analysis of Hadoop performance of the Hadoop and the issues that affect it on heterogeneous clusters, and suggest some guidelines to overcome these obstacles, the aim of which is to improve the performance of Hadoop framework.

The previous studies aimed to provide an analysis of Hadoop MapReduce performance or to study some operational shortcomings, especially those affecting the processing time, but most of them did not give a global view of MapReduce performance and the different factors that impact it, they focused on a specific factor or parameter, they also ignored to study the effect of workload and its adaptability with the scheduler on the performance of the MapReduce model.

3. EXPERIMENTS

In the current section, we introduce the environment setup, and reveal the objective of the experiments and their conditions.

In our experiments, we used a Hadoop cluster containing 1 master node and 10 slave nodes (homogenous machines with the same hardware and software specifications), all nodes are in the same rack. The configuration of the cluster is detailed in Table 1, the installed version of Hadoop in the system is 3.3.1, the data block size in HDFS system was set to 64 MB and the replication factor was set to 3, each node as a Task Tracker was configured with 2 maps and 2 reduce slots. In the runtimes, we use typical MapReduce applications such as WordCount, Grep, and QuasiMonteCarlo, with different dataset sizes of 5, 10, 15, and 20 GB respectively. The results of each experiment are averages of 10 to 15 executions for each application with each dataset size, in order to accurately measure the execution time of each application. The total number of executions is about 700 for all experiments.

Table 1. Cluster configuration

Nodes	Quantity	Configuration
Master Node	1	2 single-core 2.6Ghz 4GB RAM
Slave Node	10	2 single-core 2.6Ghz 4GB RAM

Table 2. Specifications of schedulers

Scheduler	Advantages	Disadvantages
FIFO Scheduler	- Easy implementation - simple to execute	- Long wait for jobs - Starvation
Fair Scheduler	- Resource allocation according to jobs priority - Reduced waiting time for small jobs	- Complex configuration
Capacity Scheduler	- Good for working with multiple users - Maximizing the throughput and the utilization of the cluster	- Not easy to configure - Complexity

The purpose of our analysis is to study how certain factors can affect the performance and efficiency of the schedulers in Hadoop MapReduce, by benchmarking the performance of these schedulers in processing different types of workloads and under various conditions. These factors may be related to hardware configuration, software, or the workload itself. The schedulers studied in these experiments are: the FIFO scheduler, the Fair scheduler and the Capacity scheduler (Some specifications of the schedulers used are shown in Table 2). We would add more schedulers to the study in future work.

For workloads, we chose the usual applications (WordCount, Grep, QuasiMonteCarlo) to calculate and compare the processing time under the studied schedulers. Table 3 shows the descriptions of the applications. Each application on Hadoop has typically its own characteristics and resource requirements. For example, some applications are CPU-bound, memory-bound, etc. CPU-bound applications are those that require a significant amount of computing resources, they involve performing complex calculations, data processing and CPU-intensive algorithmic operations. Memory-bound applications require a lot of data in memory, so most of their execution time is spent reading and writing data. The data processing in MapReduce as shown in Figure 2, goes through several stages, in order to simplify the calculations in this study, we will only consider the three main stages (mapping phase, shuffling phase, and reducing phase), the total job processing time can be calculated using the following equation:

$$T_j^{Total} = T_m^{Total} + T_{sh}^{Total} + T_r^{Total} \tag{1}$$

where,

- T_j^{Total}: the total processing time of the entire job.
- T_m^{Total}: the total processing time of the mapping phase.
- T_{sh}^{Total}: the total processing time of the shuffling phase.
- T_r^{Total}: the total processing time of the reducing phase.

Table 3. Applications description

Application	Description
WordCount	A simple application which is used to count the number of occurrences of each word in a given input file.
Grep	a function that extracts matching strings from text files and counts how many times they occurred.
QuasiMonteCaro	It is a program that estimates the value of pi using a quasi-Monte Carlo method.

4. RESULTS AND DISCUSS

In this section, we try to analyze and discuss the different results of the experiments and come up with some interpretations and conclusions.

As part of our analytical study of the MapReduce performance and in order to identify factors that influence it, we started our experiments by comparing the schedulers' performance in terms of processing the applications (WordCount, Grep, QuasiMonteCarlo) to find out how well they handle different types of workloads, the results of this experiment are shown in Figure 3.

According to the output in Figure 3, it appears clearly that there is a variation in the results of schedulers in job processing time, and they do not always perform with all workloads in the same way, but their performance depends on the type of the workloads and differs from one job to another. For example, if we observe in Figure 3, the results of the Fair scheduler in processing WordCount and Grep applications, it performs better than the Capacity scheduler in processing the same applications, but in the case of the QuasiMonteCarlo job, we notice that the Capacity scheduler is much better than the Fair scheduler, with more than 40% difference between the results of the two schedulers in terms of processing time. These outcomes lead us to question the existence of a certain relationship of influence between the performance of schedulers and the type of workload, according to the results of Figure 3, which shows that the performance of schedulers is not all the time stable and that it changes depending on the type of workload.

To answer this question and to have a clearer idea of how workload itself impacts the performance of the schedulers, and to validate the interpretation of the results obtained in Figure 3, we conducted a performance comparison of the previous schedulers, this time by running a single WordCount application per each scheduler with different types of datasets. The results of comparing schedulers in terms of processing time are shown in Figure 3. The goal of these experiments is to see how each scheduler handles a specific type of workload each time.

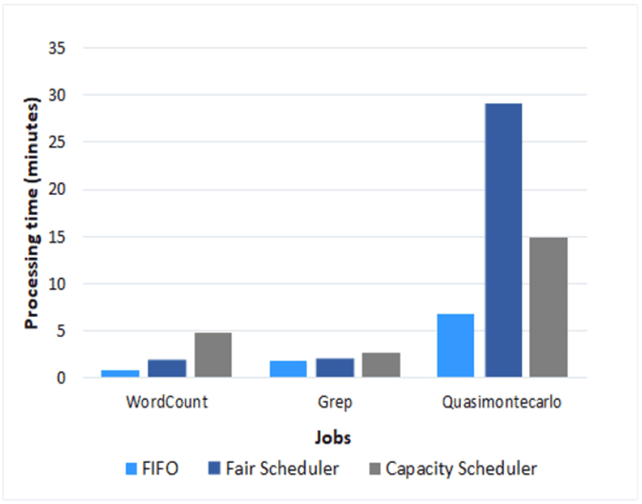

Figure 3. Comparing jobs processing time by every scheduler

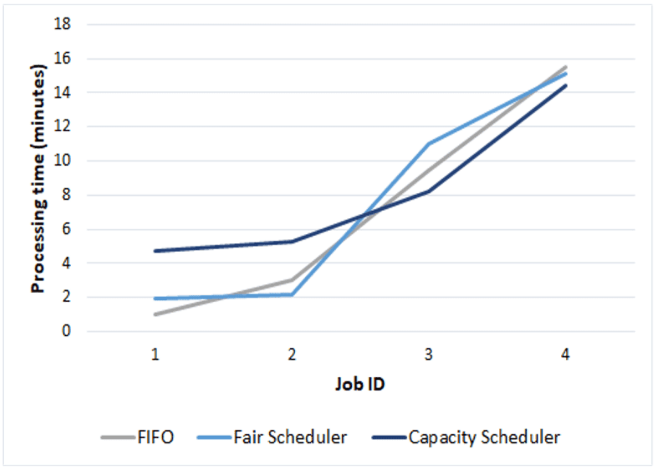

Figure 4. Processing time of WordCount application with different datasets

From the result in Figure 3, we can clearly observe that there is a variation in the performance of the schedulers, for example, fair scheduler performs well for the job 1 and 2 compared to the Capacity scheduler, but if we take jobs 3 and 4, Capacity scheduler was better than Fair scheduler. From the result of Figures 3 and 4, it can be deduced that scheduler can perform well for certain types of workloads but this is not always the case with other types, which makes it difficult to accurately estimate the timespan for a given workload in Hadoop MapReduce. We can infer that the performance of MapReduce depends not only on the scheduler used, but also on the type of workload itself which becomes like an impact factor.

The variation in scheduler performance under different workloads may also be related to the design of these schedulers, for example for a FIFO scheduler where the processing order of jobs is consistent with their arrival order, small jobs will remain pending for a long time, which may lead to system performance degradation, and as a solution to this shortcoming, the fair scheduler proposes to share cluster slots fairly among jobs, but in the case where sequences of small jobs dominate the workloads, the FIFO scheduler may outperform the fair scheduler [17].

When we talk about big data, we are talking about enormous quantities of data to be processed, and the processing time becomes a critical criterion in this case, especially if we take into account its impact on allocation costs, and hence as shown in the results, selecting the most appropriate scheduler for a given workload, not just the highest performer, remains very important when allocating data across Hadoop platforms to improve data processing time results. In another sense, having some sort of adaptation between the scheduler used and the workload to be executed guarantees better performance of MapReduce.

As mentioned before, workloads are of different types: CPU-bound, memory-bound, I/O-bound, ..., and hence processing CPU-bound workloads like WordCount application which depends on CPU speed will definitely be different from processing memory-bound workloads where memory speed or size becomes an important factor, or processing I/O-bound workloads which are limited by I/O speed (like reading/writing from storage device, or from network, ...), and these differences in characteristics between different workloads can lead to variation in Hadoop MapReduce performance.

Among the factors that can affect the performance of MapReduce is the Hardware it runs on, particularly the CPU and memory which are considered as the main hardware components, and also the number of map and reduce slots per node. To assess the impact of the hardware configuration on MapReduce performance, we conducted experiments to find out the effect of the processor frequency and memory size and the number of slots on the data processing time results, firstly, by examining the schedulers' performance with another CPU frequency, the results are shown in Figure 5, secondly, by testing the performance of MapReduce in processing the WordCount application under different memory sizes. The results obtained are shown in Figure 6, and finally, by changing each time the number of map and reduce slots and comparing the MapReduce performance, the results are presented in Figure 7.

As shown in Figure 5, increasing the CPU frequency optimized the processing time results, which proves that CPU speed can impact the performance of schedulers, and the performance increase rate exceeds 12% from the results obtained.

We can clearly notice from Figure 6, that as the memory size increases, the processing time decreases, if we take the increase in memory size from 4 GB to 8 GB or from 8 GB to 16 GB, we obtain an important performance increase value, which ranges between 16% and 21%. This suggests that memory is one of the hardware resources that has a remarkable effect on task execution time.

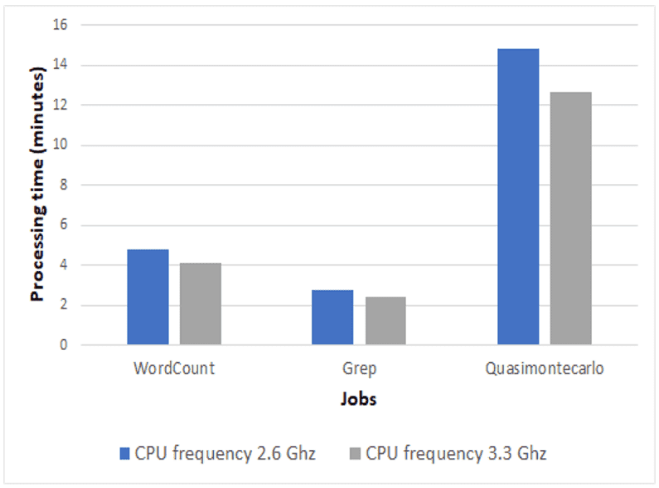

Figure 5. Jobs processing time relative to processor frequency

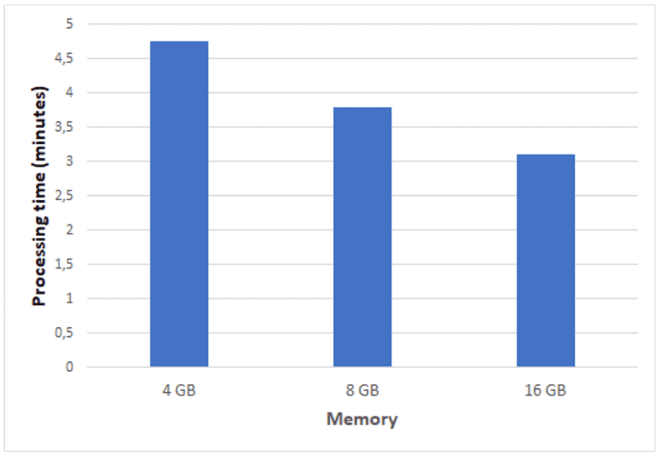

Figure 6. Processing time of WordCount relative to available memory

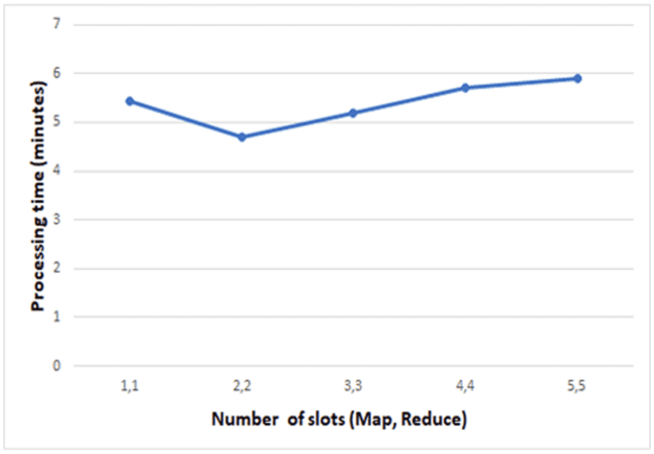

Figure 7. Processing time of WordCount relative to number of slots

According to the results in Figure 7, we see two aspects for the performance of MapReduce in terms of processing time, the case where the number of slots is less than 2, we notice that the more this number increases, the more the processing time decreases, but when the number of slots exceeds 2, we observe that the increase in the number of slots implies a slowdown in processing time. We can say that choosing the optimal number of map and reduce slots can improve the processing time results.

As a summary of this section, what can be concluded from these experiments and the analysis of the results obtained is that the type of workload itself and the hardware configuration of the cluster nodes have an inherent impact on MapReduce performance especially in job processing time. There are other factors that have not been addressed in this study that can affect the performance of the schedulers, such as (network bandwidth, concurrent workload, data locality, data skew, ...). Among the factors, we also find the heterogeneity of the clusters, since we can observe some variation in the performance of the MapReduce processing model, due to differences between nodes in the cluster in terms of configuration, performance, and computing capacity. This variation makes the estimation of the execution time of the workloads a bit more difficult in this type of system. So, we can say that heterogeneity of the cluster can also be one of the factors impacting the performance of Hadoop MapReduce.

The obtained results generally give a clearer idea about the performance of MapReduce in executing applications. They show that MapReduce performance can be affected by several factors, including the adopted architecture (physical configuration), the scheduler used, but also the level of adaptation between the scheduler and the workloads. This can be useful for big data platforms and their clients to choose the best architectures and also to look for some adaptation between the scheduler and the workloads in order to obtain better processing time values and good performance of MapReduce in general. These results can also be exploited by big data researchers for further studies.

5. CONCLUSIONS

In order to identify some factors that can affect the performance of Hadoop MapReduce, we presented in this paper a performance analytical study carried out using certain types of schedulers and with different types of workloads, our analysis also studied the impact of hardware configuration on MapReduce performance. The results obtained showed that the accurate runtime estimation is still difficult, because the MapReduce performance in processing a given workload depends not only on the scheduler used, but also on other factors, notably the nature of the workloads themselves, and the hardware on which they operate such as CPU speed or Memory capacity, or the number of map and reduce slots, as proven by experiments. In conclusion, we can say that the choice of the scheduler should not only depend only on its performance, but also on its adaptation to our workload, certainly with a good hardware configuration to optimize the results of data processing time, which can contribute to improving the performance of MapReduce and increasing the efficiency of the Hadoop framework.

For future work, we aspire to expand our analysis by studying more hardware components and more schedulers, with other types of workloads, and also under other types of constraints, in order to get a clearer idea of the different factors that can affect the data processing to improve the performance of Hadoop MapReduce.

REFERENCES

[1] Khan, M., Jin, Y., Li, M., Xiang, Y., Jiang, C.J. (2016). Hadoop performance modeling for job estimation and resource provisioning. IEEE Transactions on Parallel and Distributed Systems, 27(2), 441-454. https://doi.org/10.1109/tpds.2015.2405552

[2] Liu, Q., Cai, W.D., Jin, D., Shen, J., Fu, Z., Liu, X., Linge, N. (2016). Estimation accuracy on execution time of run-time tasks in a heterogeneous distributed environment. Sensors, 16(9): 1386. https://doi.org/10.3390/s16091386

[3] Gohil, P., Garg, D., Panchal, B. (2014). A performance analysis of MapReduce applications on big data in cloud-based Hadoop. International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, India, pp. 1-6. https://doi.org/10.1109/icices.2014.7033791

[4] Peyravi, N., Moeini, A. (2020). Estimating runtime of a job in Hadoop MapReduce. Journal of Big Data, 7(1): 44. https://doi.org/10.1186/s40537-020-00319-4

[5] Issa, J. A. (2015). Performance evaluation and estimation model using regression method for Hadoop WordCount. IEEE Access, 3: 2784-2793. https://doi.org/10.1109/access.2015.2509598

[6] Song, G., Meng, Z., Huet, F., Magoules, F., Yu, L., Lin, X.L. (2013). A Hadoop MapReduce performance prediction method. In 2013 IEEE 10th International Conference on High Performance Computing and Communications &amp; 2013 IEEE International Conference on Embedded and Ubiquitous Computing, Zhangjiajie, China, pp. 820-825. https://doi.org/10.1109/hpcc.and.euc.2013.118

[7] Zhang, Z.Y., Cherkasova, L., Loo, B.T. (2013). Benchmarking approach for designing a mapreduce performance model. In Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, pp. 253-258. https://doi.org/10.1145/2479871.2479906

[8] Dhulavvagol, P.M., Totad, S.G., Sourabh, S. (2019). Performance analysis of job scheduling algorithms on Hadoop multi-cluster environment. Emerging Research in Electronics, Computer Science and Technology, 457-470. https://doi.org/10.1007/978-981-13-5802-9_42

[9] Wang, G., Butt, A. R., Pandey, P., Gupta, K. (2009). Using realistic simulation for performance analysis of MapReduce setups. In Proceedings of the 1st ACM Workshop on Large-Scale System and Application Performance, pp. 19-26. https://doi.org/10.1145/1552272.1552278

[10] Jiang, D., Ooi, B.C., Shi, L., Wu, S. (2010). The performance of MapReduce. Proceedings of the VLDB Endowment, 3(1-2): 472-483. https://doi.org/10.14778/1920841.1920903

[11] Vianna, E., Comarela, G., Pontes, T., Almeida, J., Almeida, V., Wilkinson, K., Kuno, H., Dayal, U. (2013). Analytical performance models for MapReduce workloads. International Journal of Parallel Programming, 41(4): 495-525. https://doi.org/10.1007/s10766-012-0227-4

[12] Huang, S., Huang, J., Dai, J., Xie, T., Huang, B. (2010). The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010), Long Beach, CA, USA, pp. 41-51. https://doi.org/10.1109/icdew.2010.5452747

[13] Issa, J. (2016). Performance characterization and analysis for Hadoop K-means iteration. Journal of Cloud Computing, 5(1). https://doi.org/10.1186/s13677-016-0053-0

[14] Wlodarczyk, T.W., Han, Y., Rong, C. (2011). Performance analysis of Hadoop for query processing. In 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications, pp. 507-513. https://doi.org/10.1109/waina.2011.130

[15] Pal, A., Jain, K., Agrawal, P., Agrawal, S. (2014). A performance analysis of MapReduce task with large number of files dataset in big data using Hadoop. In 2014 Fourth International Conference on Communication Systems and Network Technologies, Bhopal, India, pp. 587-591. https://doi.org/10.1109/csnt.2014.124

[16] Rao, B.T., Sridevi, N.V., Reddy, V.K., Reddy, L.S.S. (2012). Performance issues of heterogeneous hadoop clusters in cloud computing. arXiv preprint arXiv:1207.0894. https://doi.org/10.48550/arXiv.1207.0894

[17] Chen, Y., Ganapathi, A., Griffith, R., Katz, R. (2011). The case for evaluating MapReduce performance using workload suites. In 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, Singapore, pp. 390-399. https://doi.org/10.1109/mascots.2011.12