Multimed Tools Appl (2018) 77:9979–9994
DOI 10.1007/s11042-017-4685-y

Multi-objective scheduling of MapReduce jobs in big data processing

Ibrahim Abaker Targio Hashem[1] · **Nor Badrul Anuar**[1] · **Mohsen Marjani**[1] · **Abdullah Gani**[1] · **Arun Kumar Sangaiah**[2] · **Adewole Kayode Sakariyah**[1]

Received: 22 January 2017 / Revised: 14 March 2017 / Accepted: 3 April 2017 /
Published online: 3 May 2017
© Springer Science+Business Media New York 2017

Abstract Data generation has increased drastically over the past few years due to the rapid development of Internet-based technologies. This period has been called the big data era. Big data offer an emerging paradigm shift in data exploration and utilization. The MapReduce computational paradigm is a well-known framework and is considered the main enabler for the distributed and scalable processing of a large amount of data. However, despite recent efforts toward improving the performance of MapReduce, scheduling MapReduce jobs across multiple nodes has been considered a multi-objective optimization problem. This problem can become increasingly complex when virtualized clusters in cloud computing are used to execute a large number of tasks. This study aims to optimize MapReduce job scheduling based on the completion time and cost of cloud service models. First, the problem is formulated as a multi-objective model. The model consists of two objective functions, namely,

✉ Nor Badrul Anuar
badrul@um.edu.my

✉ Arun Kumar Sangaiah
arunkumarsangaiah@gmail.com

Ibrahim Abaker Targio Hashem
targio@siswa.um.edu.my

Mohsen Marjani
marjanimohsen@gmail.com

Abdullah Gani
abdullah@um.edu.my

Adewole Kayode Sakariyah
adewole.ks@siswa.um.edu.my

[1] Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia

[2] School of Computing Science and Engineering, VIT University, Vellore 632014, India

(i) completion time and (ii) cost minimization. Second, a scheduling algorithm using earliest finish time scheduling that considers resource allocation and job scheduling in the cloud is proposed. Lastly, experimental results show that the proposed scheduler exhibits better performance than other well-known schedulers, such as FIFO and Fair.

Keywords Hadoop · MapReduce · Cloud computing · Big data · Scheduling algorithms

1 Introduction

Recently, large volumes of data or "big data" have been continuously produced from daily activities, such as those involving smartphones, sensors, factories, and business transactions; these big data affect nearly every aspect of modern society [1, 17, 21]. These data can be stored in low-cost, commodity computers in a distributed network and used for analytics to extract knowledge as well as for other purposes. The term "big data" refers to "a set of analytical techniques and technologies that require new forms of integration to uncover largely hidden values from large datasets that are diverse, complex, and in a massive scale" [11]. Big data analytics assists data scientists in uncovering hidden patterns and other useful information from large volumes of data. The use of the analyzed data helps increase the understanding of organizations of the information contained within the data to improve their business decisions.

Big data are normally processed in a distributed parallel manner across a large number of machines [26, 28, 32]. The MapReduce computational paradigm is a well-known framework and is considered the main enabler for the distributed and scalable processing of a large amount of data [5, 6, 34]. Moreover, the data center in a large-scale organization can include more than one MapReduce jobs running simultaneously. Each job frequently consists of multiple tasks, many of which are periodically scheduled. Thus, deciding on which tasks to schedule at a certain time is a critical factor in a scheduling process [4].

Scheduling plays an important role in big data, mainly to reduce the execution time and cost of processing. The MapReduce scheduling model assumes that the time to process a task of a particular node is fixed and can perform work at approximately the same rate [35]. Nevertheless, the MapReduce scheduling model requires additional resources apart from the nodes to process jobs in real applications, and the processing time of a job is determined internally by the amount of allocated resources. Moreover, Hadoop MapReduce assumes that resources are similar and data locality is frequently the only scheduling constraint [20]. However, the use of virtual machines (VMs) leads to the Hadoop cluster becoming increasingly heterogeneous, such that a cloud may have several clusters with different characteristics. Resource heterogeneity in a cluster may either be heterogeneous or homogeneous. In homogeneous clusters, the nodes have similar resources in terms of the central processing unit (CPU), memory, storage, and networking capabilities. By contrast, in heterogeneous clusters, the nodes have different resources in terms of CPU, memory, storage, and communication speeds [24].

Although MapReduce can improve its performance by adding more compute nodes from the cloud to speed up computation, this approach of "renting more nodes," particularly the cost in a pay-as-you-go environment, is not too effective [18]. Furthermore, MapReduce adopts a runtime scheduling scheme. The scheduler assigns data blocks to the available nodes for individual processing. This scheduling strategy introduces runtime cost and may slow down the execution of the MapReduce job [18], as shown in Fig. 1. In this situation, the time to complete the tasks and the cost of the allocated resources should be considered.

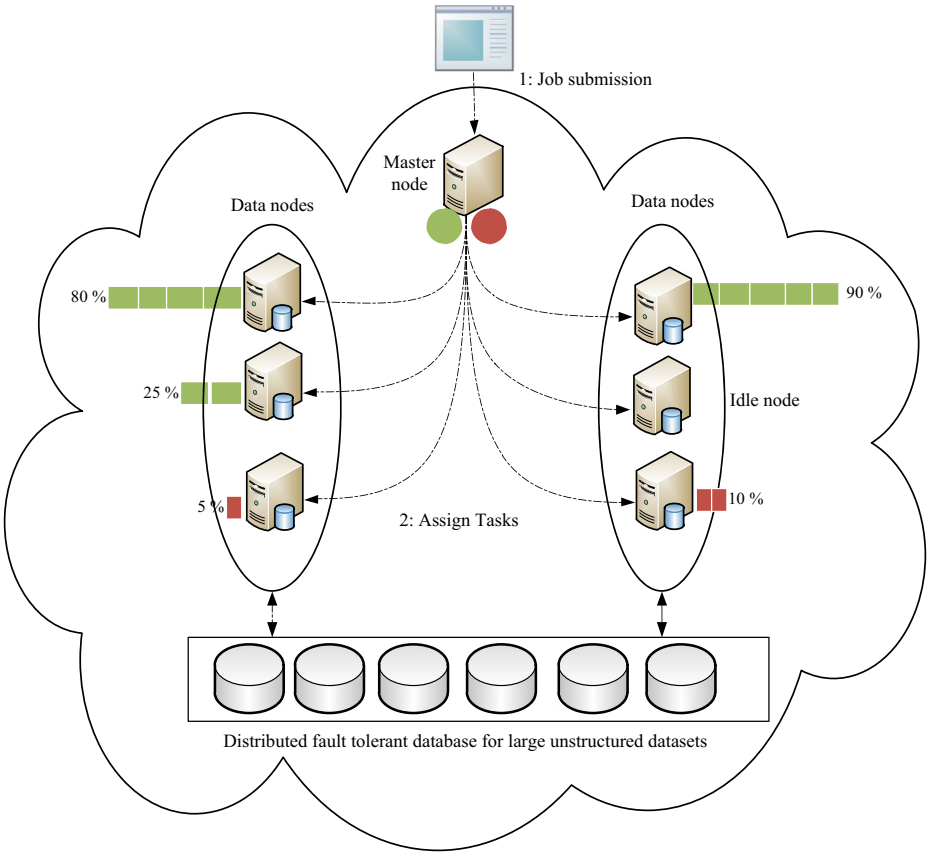


Fig. 1 Big data scheduling process using the Hadoop system

The scenario illustrated in Fig. 1 presents the big data scheduling process using Hadoop MapReduce in a cloud computing environment. Hadoop is an open source program that supports the processing and storage of large scale data. It comprises of two main components, that is, the distributed file system called Hadoop Distributed File System (HDFS) and MapReduce engine. HDFS consists of NameNode called master, Secondary NameNode called checkpoint, and several DataNode called slaves. NameNode can only stores the metadata of HDFS and DataNode stores data in the HDFS. While MapReduce consists of a single master JobTracker and one slave TaskTracker. The tasks are divided across numerous virtual nodes in the cloud and will be executed in parallel. However, a few nodes can possibly slow down the overall execution of the tasks due to various reasons, such as software misconfiguration and hardware degradation. When a client submits jobs to the master node, the jobs will be broken down into tasks. These tasks will be executed, in which the entire execution process is dominated by the slowest data node in the cluster.

The earliest study Medhane and Sangaiah [25] have proposed multi objective optimization algorithm for wireless networks. However, this study aims to optimize MapReduce job scheduling in the cloud by proposing a multi-objective model. The proposed model is designed based on the combination of two main models, namely, completion time and cost, to fulfill the performance objectives and maximize the

efficiency of a Hadoop cluster in the cloud. A scheduling algorithm is also proposed based on the adopted earliest finish time algorithm to establish the relationship between resource allocation and job scheduling. The proposed algorithm is evaluated using Hadoop with a scheduling load simulator. The simulation results obtained from the experiments show the effectiveness of the proposed algorithm framework.

The remainder of the paper is structured as follows. Section 2 discusses related works. Section 3 describes a problem formulation process for MapReduce scheduling and the motivation in scheduling in the MapReduce framework. Section 4 presents the proposed scheduling model for MapReduce and describes the scheduling algorithm framework. The proposed model aims to identify the importance of job scheduling and resource allocation in the cloud by considering the completion time and cost minimization models. Section 5 provides the experimental results. Section 6 presents the discussion and concluding remarks.

2 Related works

Scheduling is definitely not a new problem and has been widely investigated in the distributed computing literature. In MapReduce jobs or tasks, scheduling is given access to resources (e.g., processing time, CPU, communication, and bandwidth) for execution and to achieve optimum quality of service. Solving the scheduling problem may require making a discrete choice to obtain a desirable solution among different alternatives [4]. The most relevant studies related to the current study consider time and cost [3, 23, 27, 29].

A few attempts have been made in the past to assess scheduling in big data platforms. The report of Tiwari et al. [31] provided a comprehensive and structured survey of scheduling algorithms used for big data platforms. Their study proposed a multidimensional classification framework based on quality requirements, scheduling entities, and the adaptation of dynamic environments. Moreover, works related to scheduling big data in MapReduce, which all aim to improve the performance of big data platforms, have been published [5–7, 22, 30]. Nevertheless, certain areas should still be explored, particularly with respect to scheduling and resource management in cloud heterogeneous environments. To date, existing studies have focused on reducing execution time, overhead, resource utilization, and data locality ([9, 35]; X. [36]). Ibrahim et al. [15] proposed a scheduling algorithm called Maestro, which was designed to improve the performance of MapReduce computation. The current Hadoop schedulers perform inefficient scheduling of map tasks by degrading replica distribution. The Maestro scheduler has two objectives. First, each data node is equipped with empty slots based on the replication scheme for their input data and the number of hosted map tasks. Second, the runtime of each scheduling task is considered and the replicas of the input data of the task determine the scheduling of the map task on a particular node. With these objectives, the scheduler can achieve high data locality for the map tasks and balance the intermediate data distribution for the shuffling phase. The results of the Maestro algorithm are promising compared with the current Hadoop scheduler. Isard et al. [16] addressed the problem of scheduling jobs on distributed computing clusters that are close to the application data stored in the compute node. Each job in the node is managed by a root task that is assigned by the scheduler in the cluster. Such node is responsible for submitting a list of workers to the scheduler, in which these works exhibit no dependency relationship. For each worker, the root is calculated based on the preference list of computers and racks with a high data rate in the

rack of computers. Quincy was originally designed for DryadLINQ, but can be applied to other systems, such as MapReduce. This scheduler implemented the queue concept based on the hierarchical nature of the cloud network to allow data to be executed locally and close to the computation. Queues exist for machines, racks, and the system. Quincy works well when data locality is even and job lengths are approximately equal. Nita et al. [27] proposed a multi-objective scheduling algorithm, called MOMTH, that could be applied to many tasks in Hadoop for big data processing. Accordingly, two objective functions related to users and resources are considered with constraints, such as deadline and budget. A collaboration platform, called MobiWay, which exposes interoperability between a large number of sensing mobile devices and a wide range of mobility applications, is used for the performance analysis of MOMTH to evaluate the algorithm in the scheduling load simulator. When compared with FIFO and Fair schedulers, this algorithm exhibited a similar performance for the same approach.

3 Problem formulation

Several common assumptions are made in this study given the relatively high complexity of MapReduce job scheduling. Some of these assumptions have been used in Nita et al. [27] and Wang and Shi [33]. These assumptions are as follows. (i) One or more free slot(s) are available at a given time in each node $N = \{n_1, n_2, \ldots, n_m\}$ in the cluster, where the minimum number of tasks for the map is reduced to less than or equal to the available slots. (ii) Big data processing for each query is translated into one or more MapReduce job(s) $J = \{j_1, j_2, \ldots, j_h\}$, where each job has multiple tasks $T = \{t_1, t_2, \ldots, t_n\}$, which consist of a known number of map tasks N_m and reduce tasks N_r. (iii) The reduce tasks can only be launched when all the map tasks have been completed. (iv) For each map task, the exact amount of data processed S_m is known from the beginning and is equally distributed among map nodes. (v) Each job has arrival time A, deadline D, and allocated budget B for using the node. (vii) Sufficient resources are allocated for each task in the cloud, which implies that a node is never completed by more than one tasks, and its allocation is charged based on the actual time that it is used and the fixed service rate. Thus, before discussing the model for completion time and monetary cost, the definition of the problem is described as follows: A MapReduce job J is modeled as a workflow that consists of multiple tasks T. This workflow is a collection of independent map and reduce tasks executed in parallel and denoted as $t = \{t_{m_1}, t_{m_2}, \ldots, t_{m_u}, t_{r_1}, t_{r_2}, \ldots . t_{r_u}\}$. Each map/reduce task is run in a cloud VM known as a "node" with a possibly distinct performance configuration, and a different charge rate for each machine is deployed in the cluster. Each job has a particular number of slots assigned; these slots can be used by map and reduce tasks at any given time, where no reduce task can be started until all the map tasks for the job are completed. However, the same slots can be used by the mapper and the reducer. For each task, t_i, $0 \le i \le j$, where $t_i^u \le u \le N$ represents the time to run tasks t_i^u on node N.

4 Multi-objective proposed model

The proposed model aims to identify the importance of resource allocation and job scheduling in the cloud by considering completion time and cost minimization models. These models are

based on similar works by Kc and Anyanwu [19], Nita et al. [27], and W. Zhang et al. [37], who have proposed models that assist MapReduce jobs to meet the performance deadline with the monetary cost of using the cloud. Suitable schemes to adopt in the algorithms should be selected, particularly when the technical aspects of the chosen approach are considered. The proposed multi-objective scheduling algorithm is proposed to establish the relationship between resource allocation and job scheduling. This combination of resource allocation and task scheduling helps improve performance when processing a large amount of data using the MapReduce framework.

The multi-objective earliest finish time algorithm has been used to optimize the workflow in the cloud and to iteratively map the workflow tasks onto the resources. Aside from mapping every task onto the resource, the algorithm also maps resources onto tasks to establish a trade-off among the considered objectives. This algorithm is described in the study of Durillo and Prodan [8], in which a positive value should be returned by the service function if the mappers and reducers are sufficient to complete the tasks for a specific job within the given budget and deadline. The pseudocode described in Algorithm 1 presents the multi-objective earliest finish time algorithm, which begins with the required inputs of all the tasks that belong to a particular job in the cluster. The tasks are then split into map and reduce tasks represented by the job. The map tasks will be scheduled first, followed by the reduce tasks. The total of both tasks are scheduled in some nodes, depending on the availability of the slots. Subsequently, the mapping and reducing phases of the algorithm begin by iterating over the list of tasks of the map and reduce tasks sorted according to their order in the queue. The tasks are assigned to available resources in the cluster. Therefore, only trade-off solutions are saved to avoid assigning performance degradation.

Algorithm 1: Earliest finish time scheduling

Required

Q: the task queue, where all tasks $T \in J$ and $T \leq t_m + t_r$

N: some nodes in the cluster, where $N \geq Slots$

1. **For** each task t ∈ T_{m+r} **do**
2. Allocate T to $resources$
3. **Else**
4. Repeat the same process
5. **End**
6. **While all** t_m & t_r **do**
7. Allocate ready task t_i to any available slots
8. **End While**
9. Report all the mapped tasks
10. Wait for tasks_queue
11. Update the running tasks_queue
12. Tasks <list> = 0.

First, the tasks are assigned to available resources. The map tasks without parents, which will be on top of the list, are assigned to the first available resources. These available resources should be able to accept new tasks for execution and should not exceed the limit for accepting tasks to new slots. All of the tasks are stored in a queue and updated throughout the scheduling process. Afterward, the tasks that are ready will be assigned by the scheduler to the available cloud resources and slots. The optimum choice for the earliest finish time depends on the number of tasks in the application, the scheduling policy, and the decision model, which are configurable by the user before executing any workflow application in the cloud. Once the map and reduce tasks have completed the execution, the current workload information should be updated, as shown in Algorithm 2. After the map and reduce tasks are completed, the

execution time is collected and reported to the "Job Tracker" in the current Hadoop system. The following subsection describes the process of the scheduling algorithm framework.

Algorithm 2: Workload information gathering

1. **If** a t_m of job j is finished
2. **Then**
3. Update the overall time of t_m in the log
4. **If** reduce tasks of job j is finished **then**
5. Update overall time of t_r in the log

4.1 Completion time with the budget constraint model

Modeling completion time is an essential part of this study because it is the basis for the rest of the work, other calculations, and the proposed algorithms. This procedure is one of the most widely accepted methods for modeling the optimization problem [12]. Many variants of this model are available, but one variant is particularly related to the map and reduce task assignment problem with budget constraints [33]. In this problem, the goal is to minimize the Makespan given a particular budget constraint. To achieve this goal, the execution time $T_{(t_i,b)}$ of a task t_i with a specific budget $b = \frac{\text{the total budget } (B)}{\text{the total numer of tasks } (T)}$ should be defined as the time required to complete the task within the specific budget. The shortest time to complete the task is denoted as

$$T_{(t_i,b)} = t_t^u, c_i^{u+1} < B < c_i^{u-1}, \tag{1}$$

where the estimation of the budget per map/reduce task can be described as.

$$\boldsymbol{b} \leq 1, \forall \boldsymbol{t_i} \in \boldsymbol{J}. \tag{2}$$

The time to complete task t_i with budget b, denoted as $t_i(b)$, is defined as the time consumed when all the tasks are completed within the given budget as follows:

$$\boldsymbol{t_i(b)} = \boldsymbol{max}_{\boldsymbol{t_i} \in \boldsymbol{J}} \left\{ \boldsymbol{T}_{(\boldsymbol{t_i},\boldsymbol{b})} \right\}. \tag{3}$$

For the query, the reduce task is started immediately after the map tasks are completed. Therefore, the total sum of all the task times of the Makespan with budget B to complete all the tasks for a particular job is defined. The aim is to decrease execution time within the particular budget B.

$$\boldsymbol{t(B)} = \min_{\sum_{t_i} \in J \boldsymbol{b} \leq \boldsymbol{B}} \sum_{\boldsymbol{t_i} \in \boldsymbol{J}} \boldsymbol{T(B)} \tag{4}$$

4.2 Cost with the deadline constraint model

Pay-as-you-go is a well-known pricing model implemented by cloud service providers to charge users based on quality of service (QoS) requirements. The charges for some resources in cloud-like network bandwidth and storage are at a particular rate.

The pricing model implemented in the cloud is a pay-as-you-go model, where services are charged as per the QoS requirements of the users. The resources in the cloud, such as network bandwidth and storage, are charged at a specific rate [14]. Thus, cost has become an important objective in scheduling. Total cost incurred by processing big data can comprise many cost components, such as computation and data transfer costs. Cloud computing offers a variety of

resources and services per manner of use. These computational resources are basically used per time quantum pricing scheme. This quantum is typically 1 h, although recently, an alternative seems to be receiving increasing interest.

When the deadline for the job is given, the minimum cost to complete all the tasks is derived as

$$\boldsymbol{C}_{(\boldsymbol{N_m}+\boldsymbol{N_r})}\ (\boldsymbol{D}) = \sum_{\boldsymbol{t_i}\in \boldsymbol{J}} \boldsymbol{C_i}\ (\boldsymbol{D_i}), \tag{5}$$

where C_i (D_i) is the minimum cost to complete the task within D_i. Thus, $\boldsymbol{t_m} \le \boldsymbol{D_i}$ **and** $\boldsymbol{t_r} \le \boldsymbol{D_i}$.

$$\boldsymbol{C}_{(\boldsymbol{N_m}+\boldsymbol{N_r})}\ (\boldsymbol{D}) = \underset{\boldsymbol{t_i}\in \boldsymbol{J}}{\boldsymbol{min}} \sum_{\boldsymbol{t_i}\in \boldsymbol{J}} \boldsymbol{C_i}\ (\boldsymbol{D_i}) \tag{6}$$

The computation cost is defined based on resource R_j, such that, for each task t_i executed on resource R_j, two timestamps will be recorded, that is, A when the task starts and E when the task finishes its execution. The value E can be defined as $A + t_{(i,b)} + \max_{i\in J} \frac{\text{Size of the data}}{Ba}$ndwith. These timestamps indicate the period during which the resources should to be utilized because of the execution of task i.

Symbol	Definition
J	The number of jobs $j = 1, \ldots, n$
N	The number of nodes $N = \{n_1, n_2, \ldots, n_m\}$
T	The number of tasks $T = \{t_1, t_2, \ldots, t_n\}$
c_j^u	The cost for each job
C_m	The cost of executing a single map task
C_i	Completion time of each task
D_i	Deadline of each task
$t(B)$	The total budget of all tasks during the execution
k_i	Performance degradation perimeter
$t_i(b)$	The time consumed when all task is completed within the given budget
R_j	Resources

5 Experimental results

The experiments were conducted using the Hadoop cluster with 10 VMs installed on Linux Ubuntu 14.04. One of the VMs runs NameNode and ResourceManager, whereas the other VMs run DataNode and DataManager. Each VM has the following configuration: 2.80 GHz processor, 8 GB main memory, and 1000 GB disk space. Hadoop version 2.6.0 was used for the high-level query. The maximum replication factor "dfs.replication.max" was applied to set the replication limit of data blocks. A benchmark representative set of CPU and IO intensive applications included in the Hadoop distribution, such as WordCount and Sort, for performance analysis was used to efficiently evaluate the MapReduce task scheduling algorithms [13]. The performance of the proposed work was compared with those of the default Hadoop and Fair scheduler algorithms. The Sort and WordCount benchmarks were run on the Hadoop Scheduler Load Simulator.

A series of performance indicators should be defined to describe and compare the performance characteristics of MapReduce in the cluster. This section mainly focuses on measuring the working capability of the MapReduce jobs, including the measurement of the throughput and execution time of each MapReduce job, the processing duration, and the CPU utilization of the node. Thus, an experiment is conducted to illustrate the performance of MapReduce scheduling under different scenarios.

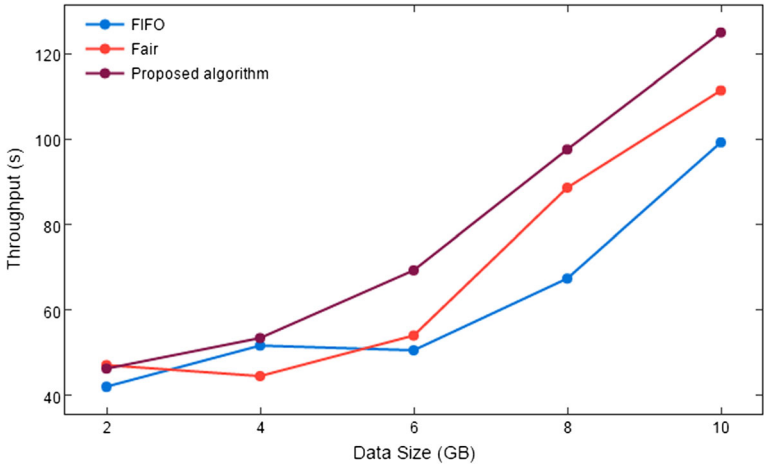


Fig. 2 Throughput using the WordCount benchmark

5.1 Throughput

Figures 2 and 3 present the throughput of different data sizes to be processed by the MapReduce framework in a cloud computing environment using the WordCount and Sort benchmarks, respectively. This processing of datasets is scheduled by different algorithms, that is, FIFO scheduler, Fair scheduler, and the proposed scheduling algorithm. Data size affects the type of scheduler required to execute the tasks at a targeted performance level. This metric significantly influences task scheduling, where the execution time of each task has to be minimized considering the heterogeneity of the cluster.

Figure 2 shows that the proposed algorithm can provide higher throughput compared with the other scheduling algorithms, namely, FIFO and Fair. The main purpose of achieving high throughput is to reduce the processing time of the workload, particularly when a large amount of data is involved. The resource utilization rate is the reflection of system throughput, which is

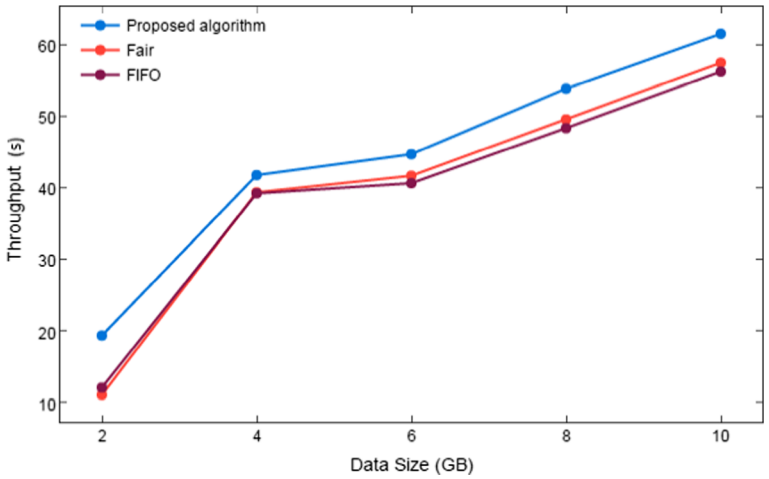


Fig. 3 Throughput using the Sort benchmark

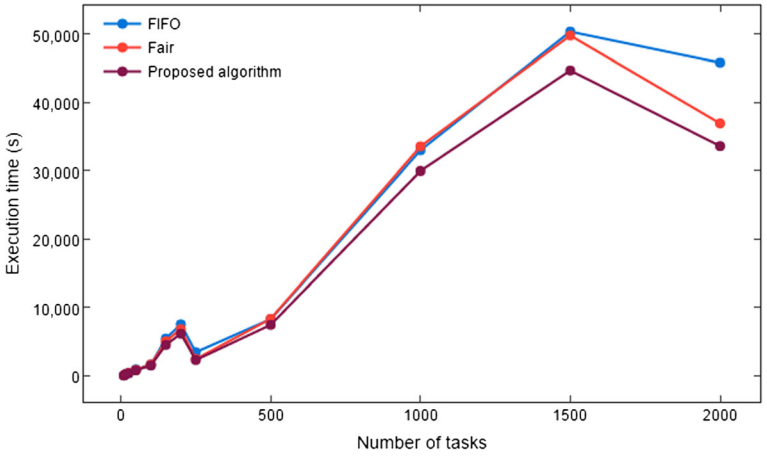


Fig. 4 Execution time using the WordCount benchmark

the useful computation cost over the total cost, including the overhead for starting up the cluster. Many cloud service providers offer hour-based or minute-based charges to users who are availing of computing service on the cloud to reduce the unnecessary CPU cycles spent on overhead, which may consume a large amount of resources to be allocated elsewhere to meet the demands of users [2].

Figure 2 presents the number of allocated data inputs in the cluster to test the proposed algorithm. The experiment conducted using the WordCount benchmark is similar for FIFO, Fair, and the proposed scheduling algorithm.

Figure 3 presents the throughput obtained by executing datasets with the same amount of data using the Sort benchmark. The tested result of the three scheduling algorithms also used a dataset with the same size as that used for the WordCount benchmark. Figure 3 shows that the proposed algorithm has high throughput compared with FIFO and Fair schedulers. However, a simple technique to achieve good performance in the FIFO and Fair algorithms is to assign an available slot to the pool with the least amount of running tasks [10]. The overall throughputs are insignificantly different under FIFO and Fair.

Figures 2 and 3 show that the amount of resources consumed by each node increases as throughput time becomes longer during the execution. Thus, the task in Hadoop scheduling should be matched carefully to the VM in the cloud environment to achieve good performance. In this manner, the system can effectively use the resources to improve the progress of executing the tasks in the Hadoop cluster.

5.2 Execution time

This section presents the results of the execution time of the MapReduce job in the cloud using the WordCount and Sort benchmarks. The design of the experiment is based on the MapReduce framework running on the cloud. The execution of the MapReduce job depends on the scheduling algorithms deployed for each experiment, which includes the proposed algorithm, FIFO, and Fair. Data related to execution time are collected using benchmarking in this section.

Figure 4 shows the differences in execution time. The straight line denotes the difference in achievement and the correlation between workload and execution time. Initially, execution is unstable in terms of time due to the small size of the data. However, execution becomes

relatively stable with the increase in data size and the number of tasks to be executed by the framework on the cloud.

Figure 4 presents the execution time of several tasks using the WordCount benchmark with FIFO, Fair, and the proposed scheduling algorithms in virtual cluster nodes with 12 Hadoop jobs of different sizes. The figure shows that the completion time of the overall processing is also increased. In the first scenario, the default algorithm FIFO is used on the Hadoop nodes without tuning the Hadoop parameters. Figure 4 illustrates that the FIFO algorithm slightly degrades the performance of Hadoop in terms of execution time and resource utilization, where data are shared among multiple users. The Fair scheduler and the proposed algorithm appear to exhibit better performance compared with the FIFO algorithm, but the data locality feature is hindered. The proposed algorithm can finish the tasks faster than the other two schedulers using WordCount to process the data. The completion times change based on the type of workload given that different workloads have various resource demands. The sharing of resources during workflow execution regardless of the size are typically relayed on the structure, a number of modules of the workflow, and the complexities. However, only a limited amount of resources that are shared among the nodes can be utilized by the small number of modules in each layer.

Figure 5 shows the use of the Sort benchmark to simulate the FIFO, Fair, and proposed scheduling algorithms. The result slightly differs from that of the WordCount benchmark, where the proposed algorithm achieves a noticeable reduction in task execution time. The Sort benchmark consumes more resources than the WordCount benchmark because of the intensive data flow and the computation of aggregate functions that must perform the Sort benchmark.

The comparison of the aforementioned algorithms indicates that performance has significantly improved using the Sort benchmark, which relies completely on the sharing of resources. Thus, the number of map and reduce tasks is scaled. The proposed algorithm occasionally exhibits better performance compared with the other algorithms, such as FIFO, given the limited resources to be shared among active nodes.

Figures 4 and 5 show that the proposed algorithm has a short execution time with regard to the number of tasks for each job in most of the cases compared with the FIFO and Fair

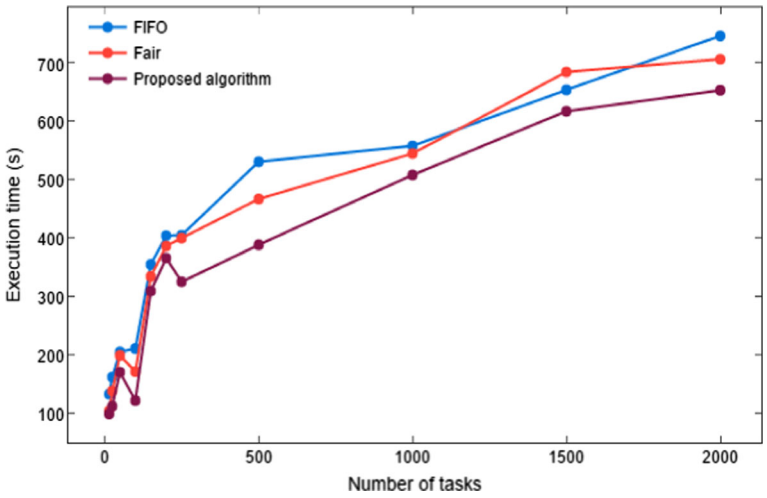


Fig. 5 Execution time using the Sort benchmark

schedulers. The default Hadoop scheduler FIFO has a long execution time compared with the other scheduling algorithms, such as Fair and the proposed algorithm.

6 Discussion and conclusion

The goal of the multi-objective algorithm proposed in this study is to optimize task scheduling in the MapReduce framework in the cloud to minimize the time and cost objectives. A good performance was achieved in different scenarios using the Hadoop benchmarks based on throughput and execution time. The evaluation metric showed that the runtime of multiple tasks in a parallel environment was reduced under the proposed algorithm, and the throughput indicated that the scheduling could offer low latency with high throughput.

The WordCount and Sort benchmarks were used to measure and calculate the performance of each node on the cluster. Moreover, formulas were used in the experiment to confirm the correctness and to verify the effectiveness of the obtained results. For the evaluation, the Hadoop MapReduce program was used on a heterogeneous parallel virtual computer.

Overall, the results of execution time show a significant improvement in processing big data with the MapReduce framework in the cloud when the proposed algorithm is used. This significant achievement can be attributed to various factors, including the flexibility of utilizing the cloud resources when executing a large amount of data, high throughput, and low latency of the deployed Hadoop cluster on the cloud. The results show that the proposed algorithm outperforms existing algorithms, such as the FIFO and Fair schedulers. Finally, the completion time of the MapReduce job can achieve a high probability prediction with the proposed algorithm on the cloud and ensure good trade-off decisions when using the multi-objective mechanism.

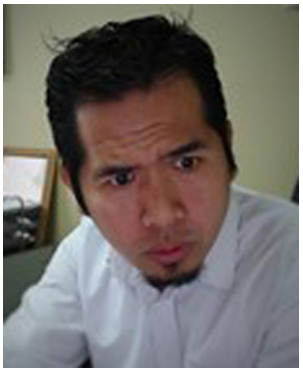
Acknowledgments This paper is financially supported by by University Malaya Research Grant Programme (Equitable Society) under grant RP032B-16SBS.

References

1. Abouzeid A, Bajda-Pawlikowski K, Abadi D, Silberschatz A, Rasin A (2009) HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. Proc VLDB Endowment 2(1): 922–933
2. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A et al (2010) A view of cloud computing. Commun ACM 53(4):50–58
3. Bittencourt LF, Madeira ERM (2011) HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. J Internet Serv Appl 2(3):207–227
4. Chang H, Kodialam M, Kompella RR, Lakshman T, Lee M, Mukherjee S (2011) Scheduling in mapreduce-like systems for fast completion time. Paper presented at the INFOCOM, 2011 Proceedings IEEE
5. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
6. Dean J, Ghemawat S (2010) MapReduce: a flexible data processing tool. Commun ACM 53(1):72–77
7. Doulkeridis C, Nørvåg K (2014) A survey of large-scale analytical query processing in MapReduce. VLDB J 23(3):355–380
8. Durillo JJ, Prodan R (2014) Multi-objective workflow scheduling in amazon EC2. Clust Comput 17(2): 169–189
9. Guo Z, Fox G, Zhou M, Ruan Y (2012) Improving resource utilization in mapreduce. Paper presented at the CLUSTER computing (CLUSTER), 2012 I.E. international conference on

10. Hadoop A (2009) Fair Scheduler https://hadoop.apache.org/docs/stable1/fair_scheduler.html
11. Hashem IAT, Yaqoob I, Anuar NB, Mokhtar S, Gani A, Khan SU (2015) The rise of "big data" on cloud computing: review and open research issues. Inf Syst 47:98–115
12. Heintz B, Chandra A, Sitaraman RK (2012) Optimizing mapreduce for highly distributed environments. arXiv preprint arXiv:1207.7055
13. Huang S, Huang J, Dai J, Xie T, Huang B (2011) The HiBench benchmark suite: characterization of the MapReduce-based data analysis. New Frontiers in Information and Software as Services,Springer, pp 209–228
14. Hussain H, Malik SUR, Hameed A, Khan SU, Bickler G, Min-Allah N et al (2013) A survey on resource allocation in high performance distributed computing systems. Parallel Comput 39(11):709–736
15. Ibrahim S, Jin H, Lu L, He B, Antoniu G, Wu S (2012) Maestro: replica-aware map scheduling for mapreduce. Paper presented at the cluster, cloud and grid computing (CCGrid), 2012 12th IEEE/ACM international symposium on
16. Isard M, Prabhakaran V, Currey J, Wieder U, Talwar K, Goldberg A (2009) Quincy: fair scheduling for distributed computing clusters. Paper presented at the Proceedings of the ACM SIGOPS 22nd symposium on operating systems principles
17. Jagadish H (2015) Big data and science: myths and reality. Big Data Res 2(2):49–52
18. Jiang D, Ooi BC, Shi L, Wu S (2010) The performance of mapreduce: an in-depth study. Proc VLDB Endowment 3(1–2):472–483
19. Kc K, Anyanwu K (2010) Scheduling hadoop jobs to meet deadlines. Paper presented at the cloud computing Technology and science (CloudCom), 2010 I.E. Second international conference on
20. Krish K, Anwar A, Butt AR (2014) [phi]Sched: a heterogeneity-aware Hadoop workflow scheduler. Paper presented at the Modelling, Analysis & Simulation of computer and telecommunication systems (MASCOTS), 2014 I.E. 22nd international symposium on
21. Laurila JK, Gatica-Perez D, Aad I, Blom J, Bornet O, Do T-M-T,. .. Miettinen M (2012) The mobile data challenge: big data for mobile computing research. Paper presented at the Proceedings of the Workshop on the Nokia Mobile Data Challenge, in Conjunction with the 10th International Conference on Pervasive Computing
22. Li J-J, Cui J, Wang D, Yan L, Huang Y-S (2011) Survey of MapReduce parallel programming model. Dianzi Xuebao (Acta Electron Sin) 39(11):2635–2642
23. Long S-Q, Zhao Y-L, Chen W (2014) MORM: a multi-objective optimized replication management strategy for cloud storage cluster. J Syst Archit 60(2):234–244
24. Lopes RV, & Menasce D (2016) A taxonomy of job scheduling on distributed computing systems. IEEE Transactions on Parallel and Distributed Systems 27(12):3412–3428
25. Medhane DV, Sangaiah AK (2017) Search space-based multi-objective optimization evolutionary algorithm. Comput Electr Eng 58:126–143
26. Mundkur P, Tuulos V, Flatow J (2011) Disco: a computing platform for large-scale data analytics. Paper presented at the Proceedings of the 10th ACM SIGPLAN workshop on Erlang
27. Nita M-C, Pop F, Voicu C, Dobre C, Xhafa F (2015) MOMTH: multi-objective scheduling algorithm of many tasks in Hadoop. Clust Comput 18:1–14
28. Philip Chen CL, Zhang C-Y (2014) Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf Sci 275:314–347. doi:10.1016/j.ins.2014.01.015
29. Rasooli A, Down DG (2014) COSHH: a classification and optimization based scheduler for heterogeneous Hadoop systems. Futur Gener Comput Syst 36:1–15
30. Sakr S, Liu A, Fayoumi AG (2013) The family of MapReduce and large-scale data processing systems. ACM Comput Surv (CSUR) 46(1):11
31. Tiwari N, Sarkar S, Bellur U, Indrawan M (2015) Classification framework of MapReduce scheduling algorithms. ACM Comput Surv (CSUR) 47(3):49
32. Valvag SV, Johansen D (2008) Oivos: simple and efficient distributed data processing. Paper presented at the high performance computing and communications, 2008. HPCC'08. 10th IEEE international conference on
33. Wang Y, Shi W (2014) Budget-driven scheduling algorithms for batches of MapReduce jobs in heterogeneous clouds. Cloud Comput, IEEE Trans 2(3):306–319
34. Yoo D, Sim KM (2011) A comparative review of job scheduling for MapReduce. Paper presented at the cloud computing and intelligence systems (CCIS), 2011 I.E. international conference on
35. Zaharia M, Konwinski A, Joseph AD, Katz RH, Stoica I (2008) Improving MapReduce performance in heterogeneous environments. Paper presented at the OSDI
36. Zhang X, Zhong Z, Feng S, Tu B, Fan J (2011) Improving data locality of MapReduce by scheduling in homogeneous computing environments. Paper presented at the parallel and distributed processing with applications (ISPA), 2011 I.E. 9th international symposium on
37. Zhang W, Rajasekaran S, Wood T, Zhu M (2014) Mimp: Deadline and interference aware scheduling of hadoop virtual machines. Paper presented at the Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on

Ibrahim Abaker Targio Hashem is currently a Ph.D. degree candidate at the Department of Computer Systems and Technology, University of Malaya, Kuala Lumpur, Malaysia; he received his M.S. degree in computing in 2012, Malaysia, and the B.E. degree in computer science in 2007, Sudan. Hashem obtained professional certificates from CISCO (CCNP, CCNA, and CCNA Security) and APMG Group (PRINCE2 Foundation, ITIL v3 Foundation, and OBASHI Foundation). He worked as a Tutor at CISCO Academy, University of Malaya. His main research interests include big data, cloud computing, distributed computing, and network.



Nor Badrul Anuar obtained his Master of Computer Science from University of Malaya in 2003 and a Ph.D. at the Center for Information Security & Network Research, University of Plymouth, UK. He is a senior lecturer at the Faculty of Computer Science and Information Technology at University of Malaya, Kuala Lumpur. He has published a number of journal papers related to security areas locally and internationally. He has a good profile of publications in renowned Journals. His research interests include Intrusion Detection System (Intrusion Detection Systems, Intrusion Response Systems, Security Event and Management, Digital Forensic and Network Security), High Speed Network (Switching, Routing, IPV6, and Multicast) and Management Information System (E-thesis, Library Systems and Online Systems). He is also a member of IEEE Communications Society, IEEE Young Professionals and IEEE Computer Society.

Mohsen Marjani is currently a PhD candidate and a member of the High Impact Research Project (Mobile Cloud Computing: Data center architecture) which is fully funded by the Malaysian Ministry of Higher Education. He received his Master of Information in Multimedia Computing in 2011 from Multimedia University, Malaysia, and BE (Mathematics) in 2003 in Iran. He has been teaching fo\r more than 10 years in different institutions in Iran. He can be contacted at: marjanimohsen@gmail.com or marjanimohsen@yahoo.com, for further information.

Abdullah Gani is Professor at the Department of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. His academic qualifications were obtained from the University of Hull, UK for bachelor and master degrees, and the University of Sheffield, UK for Ph.D. He has vast teaching experience due to having worked in various educational institutions locally and abroad—schools, teaching college, ministry of education, and universities. His interest in research started in 1983 when he was chosen to attend the Scientific Research Course in RECSAM by the Ministry of Education, Malaysia. More than 100 academic papers have been published in conferences and respectable journals. He actively supervises many students at all level of study—Bachelor, Master and Ph.D. His interest of research includes self-organized system, reinforcement learning, wireless-related networks. He is now working on mobile cloud computing with High Impact Research Grant of USD 500,000 (RM 1.5M) for the period of 2011–2016. He is a senior member of IEEE. Currently, he is a director of the Centre for Mobile Cloud Computing Research, which focuses on high impact research. He is also a visiting Professor at the King Saud University, Saudi Arabia as well as serves as Adjunct Professor at the COMSATS Institute of Information Technology, Islamabad, Pakistan.

Arun Kumar Sangaiah has received his Master of Engineering (ME) degree in Computer Science and Engineering from the Government College of Engineering, Tirunelveli, Anna University, India. He had received his Doctor of Philosophy (PhD) degree in Computer Science and Engineering from the VIT University, Vellore, India. He is presently working as an Associate Professor in School of Computer Science and Engineering, VIT University, India. His area of interest includes software engineering, computational intelligence, wireless networks, bio-informatics, and embedded systems. He has authored more than 100 publications in different journals and conference of national and international repute. His current research work includes global software development, wireless ad hoc and sensor networks, machine learning, cognitive networks and advances in mobile computing and communications. Also, he was registered a one Indian patent in the area of Computational Intelligence. Besides, Prof. Sangaiah is responsible for Editorial Board Member/Associate Editor of various international journals.

Adewole Kayode Sakariyah received B.Sc. and M.Sc degrees in Computer Science from University of Ilorin, Nigeria. He is an academic staff at the Department of Computer Science, Faculty of Communication and Information Sciences, University of Ilorin, Nigeria. Adewole is currently on his PhD program in the Department of Computer System & Technology, Faculty of Computer Science & Information Technology, University of Malaya, Malaysia. His Ph.D. research is in Network Security with specific focus on social networks. Adewole has many publications in both local and international journals with high impact. His research interests include Network Security, Biometrics, Machine learning, and Big Data analytics.