Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 8861569, 13 pages
https://doi.org/10.1155/2021/8861569

Research Article

Enhancing the Distribution of Idle Cost for Scheduling Tasks without Setup Cost in Cloud Computing

Redwan A. Al-dilami[1], **Ammar T. Zahary**[2] **and Adnan Z. Al-Saqqaf**[3]

[1]*Faculty of Computing and IT, University of Science and Technology, Sana'a, Yemen*
[2]*Faculty of Computing and IT, Sana'a University, Azal University for Human Development, Sana'a, Yemen*
[3]*Faculty of Engineering, Aden University, Aden, Yemen*

Correspondence should be addressed to Redwan A. Al-dilami; aldilami200@gmail.com

Received 10 September 2020; Revised 28 March 2021; Accepted 8 April 2021; Published 24 April 2021

Academic Editor: Mahmoud Mesbah

Issues of task scheduling in the centre of cloud computing are becoming more important, and the cost is one of the most important parameters used for scheduling tasks. This study aims to investigate the problem of online task scheduling of the identified job of MapReduce on cloud computing infrastructure. It was proposed that the virtualized cloud computing setup comprised machines that host multiple identical virtual machines (VMs) that need to be activated earlier and run continuously, and booting a VM requires a constant setup time. A VM that remains running even though it is no longer used is considered an idle VM. Furthermore, this study aims to distribute the idle cost of the VMs rather than the cost of setting up them among tasks in a fair manner. This study also is an extension of previous studies which solved the problems that occurred when distributing the idle cost and setting up the cost of VMs among tasks. It classifies the tasks into three groups (long, mid, and short) and distributes the idle cost among the groups then among the tasks of the groups. The main contribution of this paper is the developing of a clairvoyant algorithm that addressed important factors such as the delay and the cost that occurred by waiting to setup VM (active VM). Also, when the VMs are run continually and some VMs become in idle state, the idle cost will be distributed among the current tasks in a fair manner. The results of this study, in comparison with previous studies, showed that the idle cost and the setup cost that was distributed among tasks were better than the idle cost and the setup cost distributed in those studies.

1. Introduction

Task scheduling is one of the basic topics discussed by many researchers to solve the problem of scheduling. Scheduling is a collection of policies and mechanisms for the sake of controlling and ordering the work of the computer system [1]. Further, reducing the total cost charged to the task is an effective motivation. The major benefit of moving to clouds is to achieve the applications' scalability. Unlike Grids, scalability of cloud resources permits the real-time provision of resources to meet the requirements of the application. Cloud services like computation, storage, and bandwidth resources are available at lower costs. Usually, the tasks are scheduled by user requirements. New scheduling strategies are proposed to overwhelm the problems made by network properties between users and resources [2].

Cloud computing is a new paradigm for provisioning computing instances, i.e., VMs to execute jobs in an on-demand manner. This paradigm shifts the location of the computing infrastructure from the user site to the network. Thereby, it reduces the capital and management costs of hardware and software resources [3]. Public cloud is available in a pay-as-you-go charging model that allows end-users to pay for VMs by an hour, e.g., $0.12 per hour. Two key criteria determine the quality of the provided service: (a) the maximum delay among all given tasks of a job which occurred by setup of the VM to become active and (b) the dollar price paid by the end-user for activating VM [4].

The number of machines is fixed in classical scheduling problems. Thus, we have to point out which job to process on which machine. However, the cloud presents a different model on which we can release and activate machines on demand, and thereby we could control the number of machines being used to process the jobs. This shows the trade-off among the number of machines used and the delay of processing the jobs. On the one hand, we could use one machine for each task of the job and reduce the delay to a minimum if we do not have to pay for each machine. On the other hand, we could only use a single machine for all tasks of the jobs in a work-conserving model if we want to minimize cost.

The importance of this paper is to study one of the most important challenges in cloud computing, where tasks scheduling to the VMs is one of those challenges. Tasks scheduling has many criteria that should be studied by researchers. As the results of those studies, it is reflected on the end user who requested the services. The most important criteria are the delay and cost. The response time of a task that is requested by the end-user depends on several times. One of those times is the activation time (setup time) of the VM to become ready to receive the tasks. So, the researcher was interested to reduce the delay caused by the activation of VM. Also, there is a processing cost and activation cost (setup cost) of the VM to become ready to receive tasks, and this cost charges tasks. So, the researcher was interested to minimize the setup cost of VM that tasks charge.

The main contribution of this paper is the developing of a clairvoyant algorithm that addressed the important factors such as the delay and the cost that can be occurred by waiting to setup VM (active VM). Also, when the VMs are run continually and some VMs become in idle state, the idle cost will be distributed among the current tasks in a fair manner.

2. Related Work

This section presents algorithms applied in the field of this study to show the significance of task scheduling as NP-hard. These kinds of problems have been studied a lot. The instruments used were surveys for scheduling algorithms and online scheduling which is found in [5–8].

In [9], scheduling problems include functions that must be scheduled on machines subject to certain constraints to optimize a given objective function. The goal of their study is to compute a schedule that identifies the time and type of each machine function executed.

In their model, the quality of the provided service depends on two criteria: the maximum delay among given tasks of a function and the price paid by the end-user for renting VMs. So, the main objective of their study is to provide a scheduling algorithm that aims to minimize the production cost and the delay of executing a function. In their paper, the attention is paid to the online problem where no information is known on future arrival of tasks, where the arrivals of tasks are independent of the scheduling. Both the following cases considered the known and unknown task duration model (i.e., clairvoyant and nonclairvoyant). In the clairvoyant case, the duration of a task is known at its arrival. In the nonclairvoyant model, the duration is unknown at its arrival becomes known only when the task has been completed. It is supposed that the duration of each task pi is known upon its arrival. Let $E = (2T_s/\varepsilon)$ (for $0 < \varepsilon < 1$); $T_s = T_{\text{setup}} + T_{\text{shutdown}}$. The result of their study showed the cost ratio of the clairvoyant algorithm is at most $(1 + \varepsilon)$. Moreover, a long task will have a delay of T_s. The delay of a short task is at most $2E$.

The researchers in [1] enhanced the clairvoyant algorithm with known duration of tasks and limited the delay that occurred by activating the VMs. In addition, they worked to activate the VMs earlier rather than use pay-as-you-go charging model. It assumed that all computing instances available for processing are initially active continuously. When the jobs arrive, they enter the VM without waiting to activate the VM. Moreover, when a VM is no longer in use, it should not be shut down. As a result, constant time for turning it off will not occur. Both setup and shut-down times are not included in the production cost of this service. Therefore, they will not be charged by the end-user. As a result, the number of VMs is activated continuously (without shutdown) for a specific job which has a major impact on the total cost. VMs in an idle state will be calculated and distributed among the current tasks because the goal is to distribute the idle costs among the current tasks in a fair manner. In their study, several VMs are assumed in an activation state continuously, so there is no need to set up and shut down VMs. This means the delay will be initially eliminated. However, there will be initially higher costs because many numbers of VMs will be in the activation state when no jobs are in process. Researchers conducted their algorithm of the known duration of tasks. They first assumed that the duration of each task pi is known upon its arrival. Let $E = 2Ts$. A task is classified upon its arrival. It is a long task if $pi \geq E$, middle task if $T_s \leq pi < E$, and otherwise it will be short.

Several VMs were assumed to be in a running state continually. On the one hand, there would be initially higher costs because those numbers of VMs will be in a running state. Based on this assumption, three main cases were investigated. Case 1: on the arrival time, the total number of tasks submitted is equal to the number of VMs in the running state. Thus, all tasks will be processed in a one-to-one method (one task in one VM). Case 2: on the arrival time, the total number of tasks submitted is more than the number of VMs in the running state. The extra tasks need to activate additional VMs. Tasks will be sorted by size. The longest tasks will inter the main VMs. The shortest tasks will be waiting to activate additional VMs. Case 3: on the arrival time, the total number of tasks submitted is fewer than the number of VMs in the running state. As a result, the following steps were executed in the following order: idle cost of VMs which are running without tasks was calculated then; current tasks were classified into long, mid, and short. The idle cost was distributed among tasks as follows: long task should charge a cost equal to or fewer than the cost of setting up one VM, mid-task should charge a cost equal to or fewer than two-thirds of the cost of setting up one VM, and short task should charge a cost equal to or fewer than one-third of the cost of setting up one VM.

The distribution process depends on the FTFC principle (a task that terminates firstly will charge firstly). When case 3 was executed, and the idle cost was still more than the cost of setting up one VM and there is no current tasks to charge some idle cost, the following steps were executed in the following order: idle VMs were sorted as the first idle; then, some idle VMs will be turned off with this principle "the First Idle, the First Shutdown (FIFS)" sequentially until the idle cost becomes equal to the cost of the setup of one VM or fewer. Any VM which is turned off and needs to be turned on again will be run by the methodology in [9].

Performance of their algorithm described by a competitive analysis where α is the cost ratio of their algorithm to the setup cost; $\alpha = (C_{\text{opt}}/C_{\text{setup}})$ and δ is the delay ratio of their algorithm to the delay of setup VM; $\delta = (D_{\text{opt}}/D_{\text{setup}})$: when $N_{\text{task}} = N_{\text{vm}}$ through time duration $[0, t]$ $\alpha = 0$ and $\delta = 0$.

When $N_{\text{task}} > N_{\text{vm}}$ through time duration $[0, t])$ $\alpha = 0$ for all tasks, and

$$\delta = \begin{cases} 1, & \text{for additional task,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

When $N_{\text{task}} < N_{\text{vm}}$ through time duration $[0, t])$,

$$\begin{cases} 0 \le \alpha \le 1, & \text{for the longest task duration,} \\ 0 \le \alpha \le \dfrac{2}{3}, & \text{for mid} - \text{task,} \\ 0 \le \alpha \le \dfrac{1}{3}, & \text{for the shortest task.} \end{cases} \tag{2}$$

Moreover, $\delta = 0$.

In [4], this study aims to schedule task groups in a cloud-computing platform, in which the resources have various resource costs and computation performance. The researchers' algorithm measures both computation performance and the resource cost and improves the computation/communication ratio by grouping the user tasks along with a particular cloud resource's processing capability and sends the grouped functions to the resource.

In [10], this study aims to use the conventional scheduling concepts to combine them to provide a solution for better and more efficient task scheduling, which is useful to both user and service provider. In their algorithm, the cost-based tasks are prioritized based on task profit in a descending order. The tasks with higher profit can be executed on a minimum cost based on the machine to achieve maximum profit. Then, the resource with minimum cost is selected and tasks are scheduled on it until its capacity supported. The selection of task and target resource is sequential when they prioritized according to user needs.

In [11], the main contributions of their study are the following: researchers suggest a throughput optimal scheduling and load balancing algorithm for a cloud data centre when the function sizes are unknown. Function sizes are supposed to be unknown at arrival and the beginning of service. This algorithm based on using queue lengths (number of functions in the queue) for weights in the max weight schedule instead of the workload as in [12]. Their algorithm does not waste any resources if the function sizes are known unlike the algorithm in [12] which forces a refreshing time every T time slots potentially to waste resources during the process. The cloud data centre comprises L servers or machines. There are K different resources. The server I has C_{ik} amount of the resources of type k. There are M different types of VMs that the users could request from the cloud service provider. Each type of VM is specified by the number of different resources (such as memory, CPU, and disk space) that it requests. Type M of VM requests R_{mk} amount of resources of type k. Given a server, an M-dimensional vector N is said to be a feasible VM-configuration if the given server can simultaneously host $N1$ type-1 VMs, $N2$ type-2 VMs, and NM type-M VMs. In other words, N is feasible at the server I if and only if

$$\sum_{m=1}^{M} N_m R_{mk} \le C_{ik}. \tag{3}$$

For all K's, they let $N_{\max}$ denote the maximum number of VMs of any type that can be served on any server.

In [13], files grouped under many small blocks and all blocks were replicated over many servers. To process files proficiently, each function was divided into many tasks and each task was assigned to a server to deal with a file block because network bandwidth is an unusual resource. Activity-based costing is the way to measure the performance of the object and its cost. Researchers have solved the problem like poor cost control, distorted product costs, and the starvation. The researchers group the task into two different groups. These groups are as follows. Available (independent & dependent)**:** it is a group of tasks which can be completely performed on a single data centre. Partially available: a group of tasks that require resources from other data centres. Furthermore, researchers classified them into cat1, cat2, cat3, . . ., and so on, and N stands for the number of categories. These categories are grouped based on data need. For example, cat1 tasks will need data from the same data centres. Similarly, cat2, cat3, . . ., cat$N$ tasks will need the data from the same data centres.

In [14], a more efficient algorithm for task scheduling is introduced by researchers based on Priority Based Scheduling in cloud computing and the implementation of it. Development of this algorithm should focus on discussing simultaneous instead of independent task scheduling in a cloud environment. Profit-based task scheduling method has its own advantages in comparison with the traditional way of task scheduling.

In [15], deadline Constrained Heuristic and Budget based upon Heterogeneous Earliest Finish Time (HEFT) are present by researchers to schedule workflow tasks over the available cloud resources. The proposed heuristic presents a useful trade-off between execution cost and execution time under given constraints. A heuristic is evaluated for various synthetic workflow applications by a simulation process and comparison is done with state-of-art algorithm, i.e., BHEFT. Simulation results reveal that the proposed heuristic scheduling can considerably minimize the execution cost

while producing Makespan as well as the best-known heuristic scheduling under the same deadline and budget constraints. Makespan is a technique used for network monitoring. The simulation results reveal that researchers' assigned algorithm outperforms the BHEFT algorithm in terms of monetary cost while producing the Makespan as well as that produced by the BHEFT algorithm.

In [16], the researchers showed the importance of setup time and setup cost; they define the setup time as the time required to prepare the necessary resource (e.g., machines, people) to perform a task (e.g., job, operation). And the setup cost is the cost to set up any resource used before the execution of a task.

In [17], the researchers showed the importance of the execution time of jobs and proposed a new job scheduling mechanism using the firefly algorithm to decrease the execution time of jobs. The proposed mechanism is based on information about jobs and resources such as identifiers, length of the job, and speed of resource. The scheduling function in the proposed job scheduling mechanism firstly creates resources and a set of jobs to generate the population by assigning the jobs to the resources randomly. Researchers evaluate the population using a fitness value that represents the execution time of jobs. Secondly, the function used iterations to regenerate populations based on firefly behaviour to produce the best job schedule that gives the minimum execution time of jobs.

In [18], the researchers showed the importance of splitting the ready queue into subqueues and proposed a novel hybrid task scheduling algorithm based on both shortest-job-first and round-robin schedulers using a dynamic variable task quantum named SRDQ. It splits the ready queue into two subqueues, Q1 and Q2. Assigning tasks to resources from Q1 or Q2 was done mutually, two tasks from Q1 and one task from Q1.

In [19], the researchers showed the impact of cost on choosing the best one of the tasks. They proposed a task schedule in the cloud environment. The principle of the proposed algorithm was to allocate the incoming task on the best resource during the runtime of some tasks based on measuring the current situation of each resource with respect to its availability level according to its cost, the number of running tasks, and processing power to know its fitness to receive the incoming task, then choosing the best one to the incoming task.

In [20], the researchers showed the scheduling process of tasks to the VMs. They proposed an Adaptive Cost-based Task Scheduling (ACTS) to provide data access to the virtual machines (VMs) within the deadline without increasing the cost. ACTS considers the data access completion time for selecting the cost-effective path to access the data.

In cloud service environment, several studies often ignored the uncertainties in the scheduling environment, such as the uncertain task start/execution/finish time. Ignoring these uncertain factors often leads to the violation of workflow deadlines and increases service-renting costs of executing workflows. A recent scheduling type of cloud service environment is the scheduling of real-time flow. Authors in [21] proposed a novel scheduling architecture to improve the performance for cloud service platforms by minimizing uncertainty propagation in scheduling workflow applications that have both uncertain task execution time and data transfer time. Its architecture is designed to control the count of workflow tasks directly waiting on each service instance (e.g., virtual machine). Based them architecture, they developed an unceRtainty-aware Online Scheduling Algorithm (ROSA) to schedule dynamic and multiple workflows with deadlines.

In brief, our scientific paper aimed to reduce the delay resulting from the preparation of virtual machines in [9] and distribute the idle cost of VMs among current tasks in a better way than [1]. We have adopted the continuous operation of VMs and looked to find a way to distribute the idle cost more efficiently.

3. Materials and Methods

The model introduced in this paper tries to enhance the distribution of the idle VMs among current tasks in a fair manner. Thus, in this model, the distribution of idle cost is among groups of tasks rather than among tasks only. The main modification was made in case 3 of the model in [1].

3.1. Model Assumption. The job input consists of multiple tasks that need to be executed. The tasks arrive over time. Task I has an arrival time T_a and a maximum duration P_i, assuming the arrival time is known.

At time t, there is a constant number of VMs inactivation state. The activation cost is free for the first time; i.e., the activation cost for each VM for the second time is not free. Activation time for each VM is a constant (T_{setup}), the delay is denoted by (D_{setup}) which occurred by (T_{setup}), and the cost activation of each VM is constant (C_{setup}). Each VM is homogeneous and all tasks belong to a specific job. Each machine can run a single task at a time. Tasks are nonpreemptive; i.e., a task has to run continuously without interruptions. Let $e_i = T_{ai} + P_i$ which is the earliest possible completion time of task i, denoted by c_i which is the actual completion time of task i and $d_i = \mathrm{c}_i - e_i$ as the delay that the task delays for some time to find an empty VM. The machine can be activated or shut down for the first time without fees (i.e., activation machine again is not free). Any machine in an idle state (VM_{idle}) (i.e., VM does not find any task to possess) should not be shut down. Inactivation of a machine: there is T_{setup} time until the machine is available for processing. In shut down, there is T_{shutdown} time to turn off the machine. Let $E = 2T_s$ and a task is classified upon its arrival; it is a long task if $P_i \geq E$, mid-task if $T_s < P_i < E$, and otherwise it will be short.

For simplicity, it is assumed that there is only an activation time $T_s = T_{\text{setup}} + T_{\text{shutdown}}$; the shutdown is free. So, their paper focuses on the known task duration model (duration of a task is known at its arrival).

3.1.1. The Proposed Model. The proposed model could be presented as follows:

Firstly, several VMs were assumed in a running state continually. Based on this assumption, three main cases were investigated.

Case 1: on the arrival time, the total number of the tasks submitted is equal to the number of VMs in the running state. Thus, all tasks will process in a one-to-one method (one task in one VM).

Case 2: on the arrival time, the total number of the tasks submitted is more than the number of VMs in the running state. The extra tasks need to activate additional VMs (VMs activate by demand). Tasks are sorted by size. The longest tasks interred the main VMs (VMs that activated early). The shortest tasks will wait to activate additional VMs

Case 3: on the arrival time, the total number of the tasks submitted is fewer than the number of VMs in the running state. As a result, the following steps are ordered as follows.

(i) The total idle time of the VMs in an idle state was calculated.
(ii) The idle cost of the VMs was calculated.
(iii) The current tasks are classified (long, mid, short).
(iv) The tasks in group are sorted by the first shortest remaining time (SRTF).
(v) The total of idle cost is distributed as follows: the long tasks group has 45% of the total idle cost, the mid tasks group has 35%, and the short tasks group has 20%. The distribution process in a group depends on the FTFC principle (the First Terminated the First Charged).
(vi) The idle cost was distributed among tasks starting with the long tasks group where the maximum cost that a task charges does not exceed the cost of one VM setup and then the mid tasks group where the maximum cost that a task charges does not exceed two-thirds of the cost of one VM setup. Finally, the short tasks group where the maximum cost that a task can charge does not exceed one-third of the cost of one VM setup.

Secondly, when case three is executed and the idle costs are still more than the cost of one VM setup and no current tasks charge any idle cost, the following steps are ordered as follows:

(i) The idle VMs were sorted by the first idle.
(ii) The idle VMs were turned off in sequence until the idle cost become equal to or fewer than the cost of one VM setup. VM is turned off with FIFS principle (VMs that idle firstly will be shutdown firstly).

Thirdly, any VM turned off and needed to be turned on again will be applied by the methodology in [9].

FIFS Principle: it means that the idle VM which is inserted in an idle state firstly should be shutdown firstly and so on.

FTFC or SRTF Principle: it means that the task terminated firstly will be charged the idle cost firstly and so on.

Figure 1 is a diagram that describes, in brief, the steps of the model (see bold shapes and font) below.

3.1.2. Notation. The cost of the activation is free for the first time because cloud computing is a service introduced to our customers, and it is better to get it ready by the service providers for the first time. So, our introduced model keeps this service going on and gives the provider the deserved cost when there is no need for it.

3.2. Inputs and Outputs of the Model

3.2.1. Input Variables. All tasks have a known time duration of arrival time (T_a) and time services ($P_{(i)}$).

3.2.2. Constant Values. Time duration of setting up of one VM; T_s, setup cost per time unit; $C_{\text{setup-unit}}$ and idle cost per time unit; $C_{\text{idle-unit}}$.

3.2.3. Desired Variables. Initiate time to process the task (i), $T_{\text{ini}(i)}$:

$$T_{\text{ini}(i)} = T_{si} + T_{ai}. \tag{4}$$

Terminate time of task (i), $T_{t(i)}$:

$$T_{t(i)} = T_{\text{ini}(i)} = P_i. \tag{5}$$

Elapsed time to process task (i) at time unit t, $T_{e(i)}$:

$$T_{e(i)} = t - T_{\text{ini}(t)}; \quad T_{e(i)} \geq 0. \tag{6}$$

Remained time to terminate task (i), $T_{r(i)}$:

$$T_{r(i)} = P_i - T_{e(i)}. \tag{7}$$

The total of the idle time of all VM, T_{idle}:

$$T_{\text{idle}} = \sum_i T_{\text{idle}(i)}. \tag{8}$$

The total of the idle cost of all VM, C_{vm}:

$$C_{\text{vm}} = \sum_i T_{\text{idle}} i * C_{\text{idle-unit}}. \tag{9}$$

3.2.4. Output Variables. The ratio of the setup cost in the proposed model (C_{opt}) to the setup cost (C_{setup}) in [9], α:

$$\alpha = \frac{C_{\text{opt}}}{C_{\text{setup}}}. \tag{10}$$

The ratio of the setup delay in the proposed model (D_{opt}) to the setup delay (D_{setup}) in [9], δ:

$$\delta = \frac{D_{\text{opt}}}{D_{\text{setup}}}. \tag{11}$$

The ratio of the idle cost of our model ($C_{\text{idle-f}}$) to the idle cost (C_{idle}) in [1], $\alpha 1$:

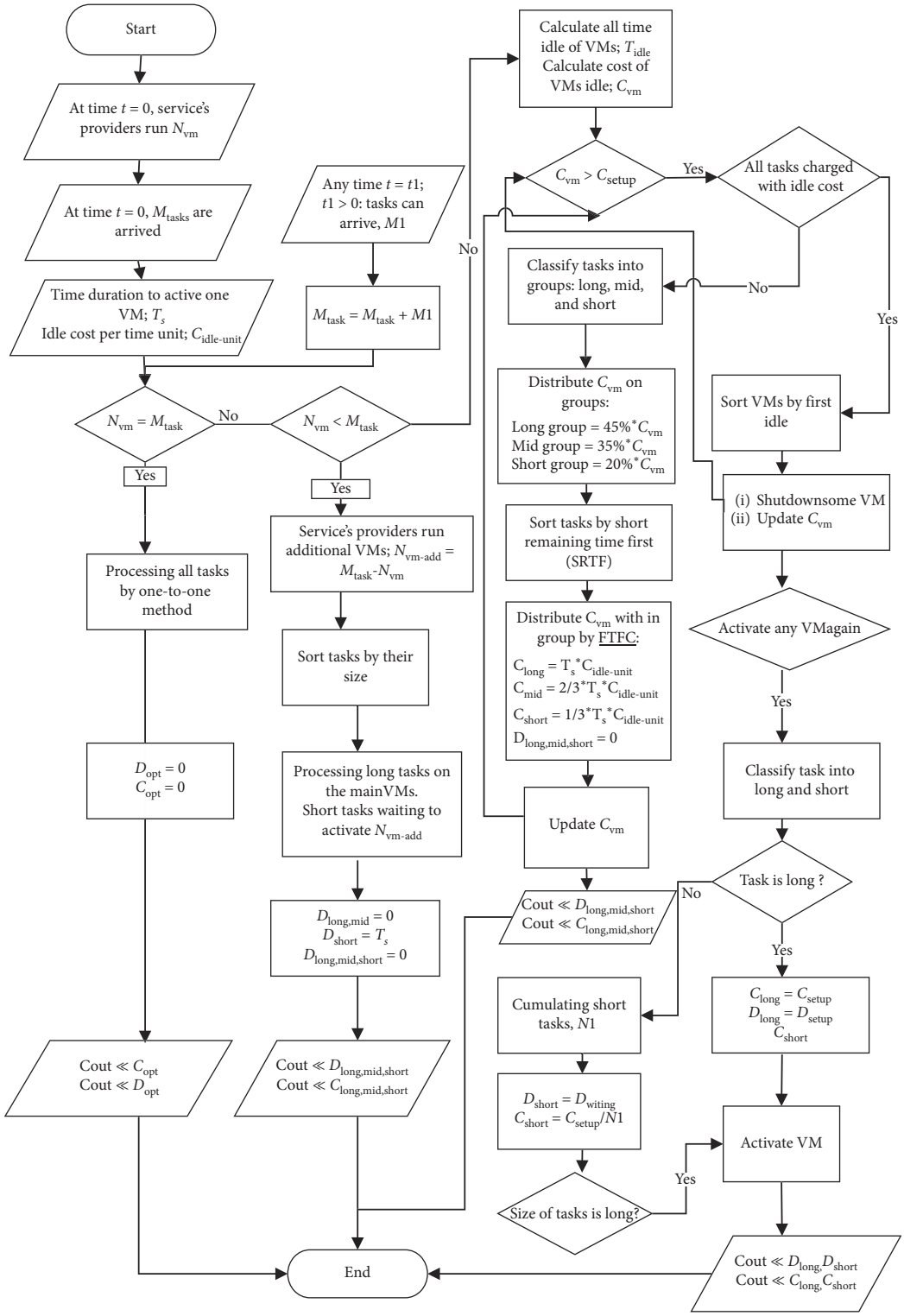


Figure 1: Steps of the model.

$$\alpha 1 = \frac{C_{\text{idle-f}}}{C_{\text{idle}}}. \tag{12}$$

It is assumed that the cost charged per unit of idle time is equal to the setup time; ($C_{\text{idle-unit}} = C_{\text{setup-unit}} = 0.002\$$). Also, the time duration to active one VM (T_s) equal = 4 time unit.

Thus, the total idle cost for each VM that is in an idle state (C_{vm}) is

$$C_{\text{vm}} = \sum_i T_{\text{idle}} i * C_{\text{idle-unit}}. \quad (13)$$

Through this equation, the total idle cost (C_{vm}) of VMs can be calculated for every moment. Thus, that cost was distributed among currently available tasks according to the proposed model.

3.2.5. The Data Used. Researchers used 21 tasks to know the duration upon the arrival at different times. Table 1 describes the data and its attributes. The researchers used C++ environment to simulate the data model.

Figure 2 describes, in brief, the procedures used to apply our enhancement using C++ environment.

4. Results

4.1. Results of the Delay and the Cost That the Tasks Charge When Setting up the VMs. The performance of the proposed model will be described in a competitive analysis where α is the idle cost ratio of the proposed model to the setup cost in [9] ($\alpha = (C_{\text{opt}}/C_{\text{setup}})$) and $\delta = (D_{\text{opt}}/D_{\text{setup}})$is the delay ratio.

Case 1. When the number of tasks is equal to the number of the activated VMs ($M_{\text{task}} = N_{\text{vm}}$) through time duration [0, 8]: $\alpha = 0$ and $\delta = 0$ as shown in Figure 3, where N_{vm} is activated early and where the VM finishes processing a task, another task comes at the same time. This means that there is no setup cost (activate cost) and there is no VMs in an idle state. This case called "optimal." Notice the curves of cost do not appear.

Case 2. When the number of the task is more than the number of the activated VMs ($M_{\text{task}} > N_{\text{vm}}$) through time duration [0, 8]:

$\alpha = 0$ and $\delta = 1$ only with additional tasks. Otherwise $\delta = 0$ as shown in Figure 4, where some tasks (short tasks) are waiting to activate additional VMs.

Notice, the delay appears only in the shortest tasks whose arrival is at $t = 0$ because the priority of processing is to the long tasks.

4.1.1. Notice. Through time duration [T_{ini}, t]; $T_{\text{ini}} > t$ Case 2 becomes Case 1.

4.2. Comparison between the Delay and the Cost of the Proposed Model Results and Another Model. This section deals with comparing the results of the proposed model that activates VMs early with the model that activates VMs on demand (the model in [9]). This comparison measures the delay and the cost that the tasks charge. This comparison uses the same data and the same implementation steps.

The following table shows the delay (T_d) and the idle cost (C_{idle}) that the tasks charge using the proposed model and the delay (D_{setup}) and the setup cost (C_{setup}) that the tasks charge by using the model in [9]. When $M_{\text{task}} < N_{\text{vm}}$ as in Table 2, the results are as follows:

Figure 5 compares the delay in the proposed model (T_d) with delay (D_{setup}) in the model [9].

Figure 6 compares the idle cost (C_{idle}) that the tasks charge in the proposed model with the setup cost (C_{setup}) that the tasks charged in the model of [9].

4.3. Results of Idle Cost That the Tasks Charge in the Three Groups of Tasks. The performance of the proposed model could be described in a competitive analysis where $\alpha 1$ is the idle cost ratio of the proposed model ($C_{\text{idle-f}}$) and the idle cost (C_{idle}) in [1]; $\alpha 1 = (C_{\text{idle-f}}/C_{\text{idle}})$;l the results are as follows.

Case 3. When the number of tasks is fewer than the number of VMs, ($M_{\text{task}} < N_{\text{vm}}$) through time duration [0, 8]), the cost ratio is $0 \le \alpha \le 1$.

The idle cost appears when the number of VMs is fewer than the number of processing tasks, i.e., $M_{\text{task}} < N_{\text{vm}}$. The idle cost of VMs is distributed among tasks rather than the cost of setting up VMs. The idle cost that is distributed does not exceed the cost of setting up the VMs.

Between $t = 7$ and $t = 9$, some VMs are in an idle state; thus, the idle cost is distributed among the current tasks. The idle cost is distributed among groups, where the long tasks group has 45% of the total idle cost, the mid tasks group has 35% of the total idle cost, and the mid tasks group has 20% of the total idle cost. FTFC principle is applied in such a process. The idle cost that the tasks charged ($C_{\text{idle-f}}$) is as follows.

(1) Table 3 shows that the maximum idle cost that the long task charges is equal to the cost of activating one VM

(2) Table 4 shows that the maximum idle cost that the mid tasks charge is equal to the two-third cost of activating one VM

(3) Table 5 shows that the maximum idle cost that the short task charges is equal to the one-third cost of activating one VM

Figure 7 describes the ratio of idle cost ($\alpha 1 = (C_{\text{idle-f}}/C_{\text{idle}})$) that the long tasks charge using the proposed model.

4.4. Comparison between the Idle Cost of the Proposed Model and Another Model. This section deals with comparing the results between the model in [1] that distributes the idle cost among current tasks by using the SRTF principle and the proposed enhancement that distributes the idle cost among current tasks by using the FTFC principle in a group of tasks. Table 6 describes the idle cost ($C_{\text{idle-f}}$) that the tasks charge in the proposed model and the idle cost (C_{idle}) that the tasks charge in the model [1].

Figure 8 shows the difference between the idle cost distribution using model [1] (C_{idle}) and using the proposed model ($C_{\text{idle-f}}$).

Table 1: Data input at different times.

N_{task}	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21
T_a	0	0	0	0	0	0	0	0	0	0	2	2	3	3	4	4	4	4	4	4	4
$P_{(i)}$	5	4	2	3	8	6	2	3	8	4	8	4	3	3	4	4	8	4	2	6	4

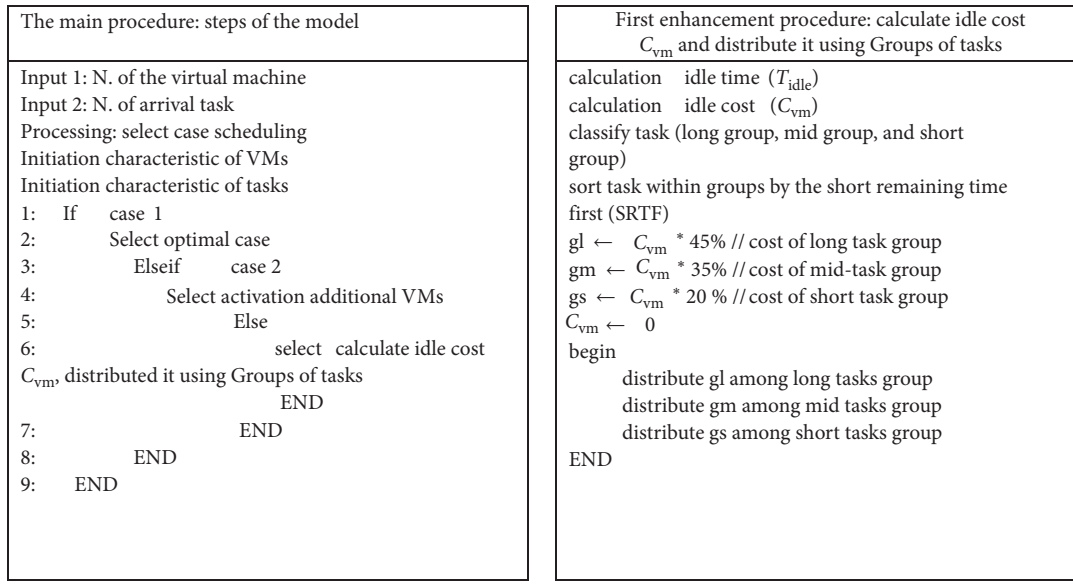


Figure 2: Procedures used in the model. (a) The main procedure. (b) Procedure of our enhancement.

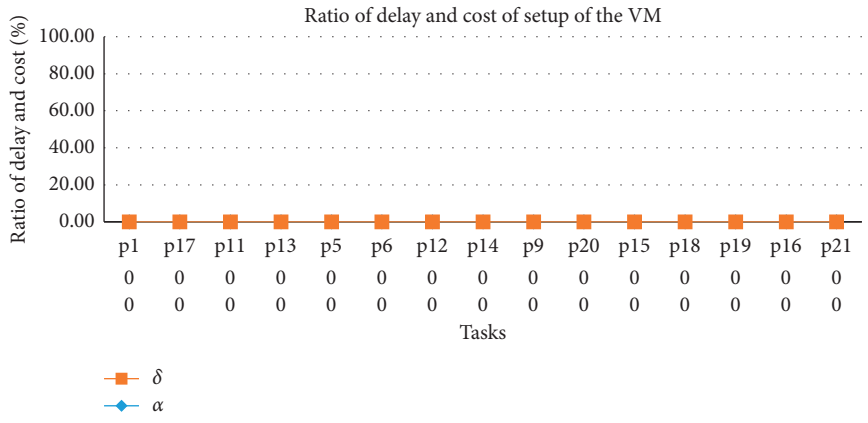


Figure 3: Ratio of delay and cost when $M_{task} = N_{vm}$ (case 1).

5. Discussion

5.1. Discussion of the Results of the Delay and the Cost of Setting up the VMs. Based on the analysis of data, the results could be discussed as follows.

The delay that occurs by setting up and shutting down VMs disappears in all cases because all VMs are activated early. In addition, the cost that occurs by setting up and shutting down VMs that the tasks charge in case 1 and 2 disappears in case 1 and case 2 because all VMs are run continually. See Figure 5, $T_d = 0$ whereas $D_{setup} = 4\ (t)$ on all tasks because all VMs are setting up on-demand. Thus, all tasks are waiting for duration time equal T_s for long tasks and $2T_s$ for short tasks. $T_d = 0$ because all VMs are activated early and run continually. Also, see Figure 6, $C_{idle} \neq 0$, and $C_{setup} \neq 0$. $C_{setup} \neq 0$ because of the cost of preparing the VM = 4∗0.002 = 0.008\$. $C_{idle} \neq 0$ because some VMs through the duration of time is idle (VMs running without any task to process it). The cost appears actually as idle cost of VMs still without processing any task. The idle cost is distributed among all current tasks where the long task charges the maximum cost of one VM setup which equals 0.008\$, the

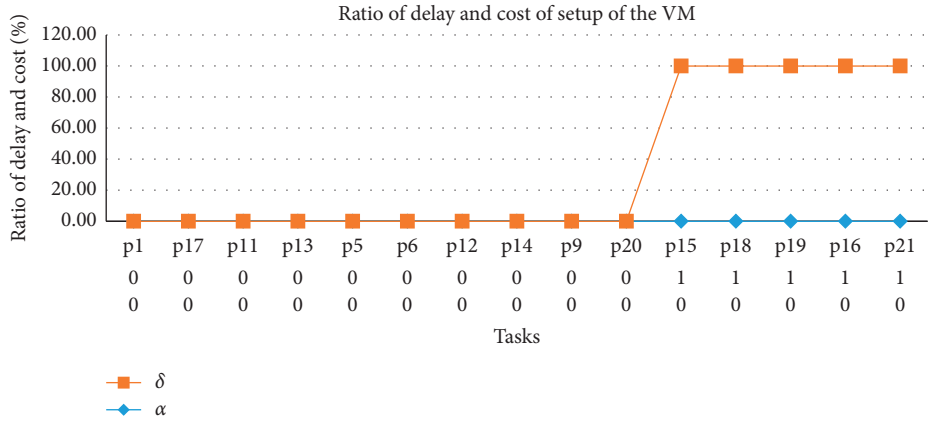


Figure 4: Ratio of delay and cost when $M_{\text{task}} > N_{\text{vm}}$ (case 2).

Table 2: Output of the delay and the cost in the proposed model and in the other when $M_{\text{task}} < N_{\text{vm}}$.

Type of results				With another model		With the proposed model	
N_{task}	N_{vm}	T_a	$P_{(i)}$	D_{setup}	C_{setup} ($)	T_d	C_{idle} ($)
p11	vm3	2	8	4	0.008	0	0.008
p5	vm5	0	8	4	0.008	0	0.008
p14	vm8	3	6	4	0.005	0	0.004

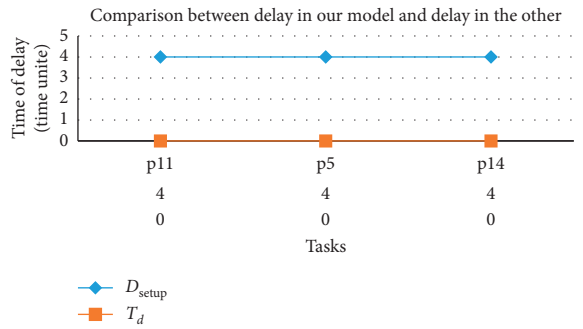


Figure 5: Comparison between the delay in the proposed model and the delay in another model.

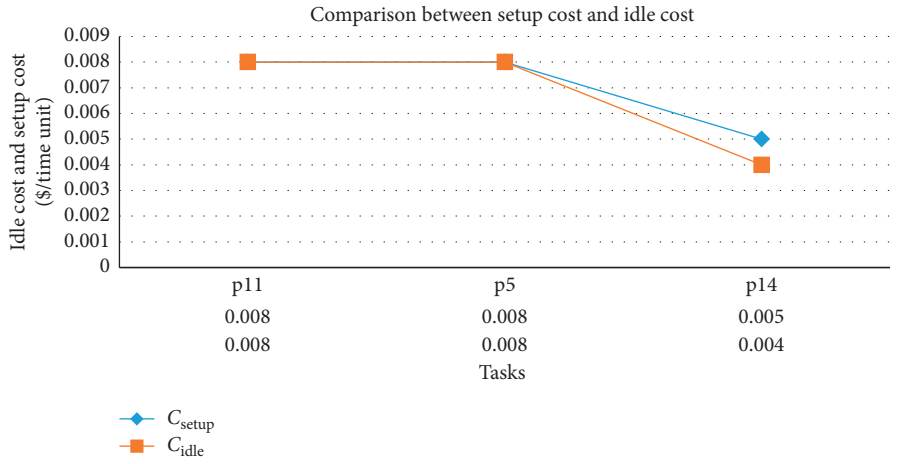


Figure 6: Comparison between idle cost in the proposed model and setup cost in another model.

Table 3: Output of the idle cost that the long tasks charge at $t = 8$.

N_{task}	N_{vm}	T_a	$P_{(i)}$	$T_{r(i)}$	$C_{idle\text{-}f}$ ($)	$\alpha 1$
p1	vm1	0	5	−4	0	0
p17	vm2	4	8	3	0.0045	0.5625
p11	vm3	2	8	1	0.0036	0.45
p13	vm4	3	2	−4	0	0
p5	vm5	0	8	−1	0.0054	0.675

Table 4: Output of the idle cost that the mid tasks charge at $t = 8$.

N_{task}	N_{vm}	T_a	$P_{(i)}$	$T_{r(i)}$	$C_{idle\text{-}f}$ ($)	$\alpha 1$
p1	vm1	0	5	−4	0	0
p6	vm6	0	6	−3	0	0
p12	vm7	2	4	−3	0	0
p14	vm8	3	6	0	0.0042	0.84
p9	vm9	4	6	1	0.0028	0.56
p20	vm10	4	6	1	0.0035	0.7

Table 5: Output of the idle cost that the short tasks charge at $t = 9$.

N_{task}	N_{vm}	T_a	$P_{(i)}$	$T_{r(i)}$	$C_{idle\text{-}f}$ ($)	$\alpha 1$
p13	vm4	3	2	−4	0	0
p12	vm7	2	4	−3	0	0
p15	vm11	4	4	3	0.0016	0.533333
p18	vm12	4	4	3	0	0
p19	vm13	4	2	1	0.0024	0.8
p16	vm14	4	4	3	0.002	0.666667
p21	vm15	4	4	3	0	0

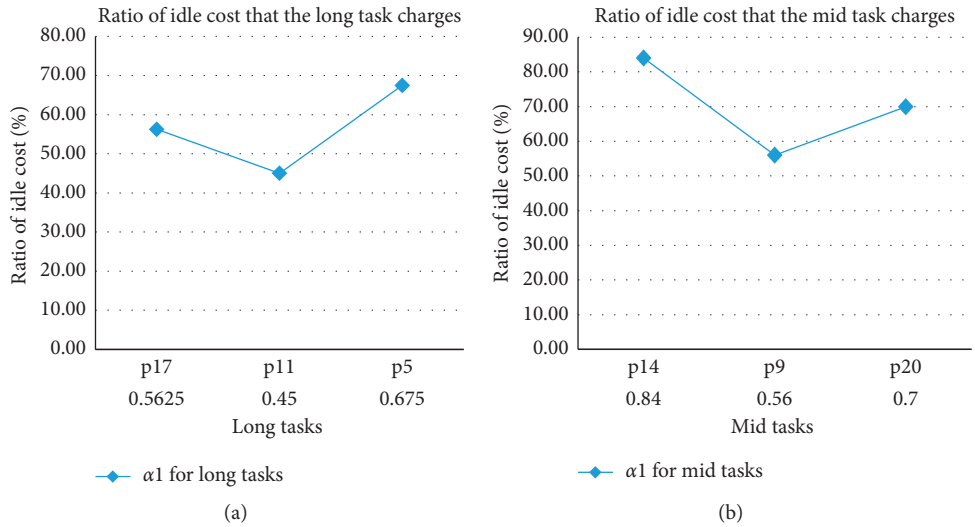
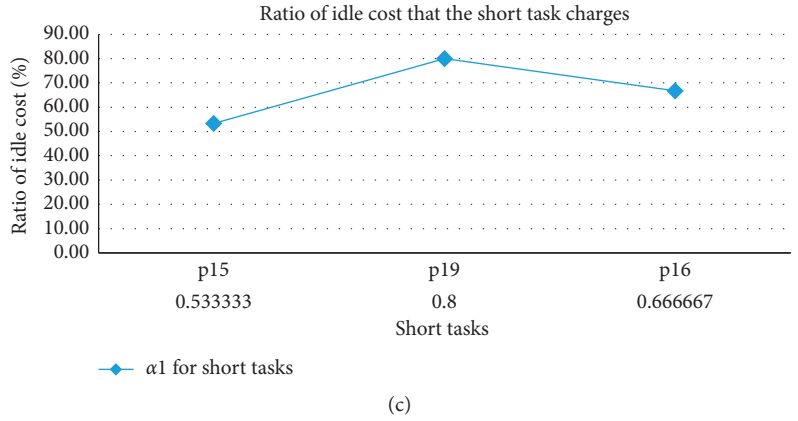


Figure 7: Continued.



(c)

Figure 7: Ratio of the idle cost that the tasks charge at $t = 9$: (a) ratio of the idle cost that the long tasks charged, (b) ratio of the idle cost that the mid tasks charged, and (c) ratio of the idle cost that the short tasks charged.

Table 6: Output of the idle cost in the proposed model and idle cost in another model at $t = 8$.

N_{task}	N_{vm}	T_a	$P_{(i)}$	C_{idle} ($)	$C_{idle\text{-}f}$ ($)
p11	vm3	2	8	0.008	0.0036
p5	vm5	0	8	0.008	0.0054
p14	vm8	3	6	0.004	0.0042
p9	vm9	4	6	0	0.0028
p15	vm11	4	4	0	0.0016
p19	vm13	4	2	0	0.0024

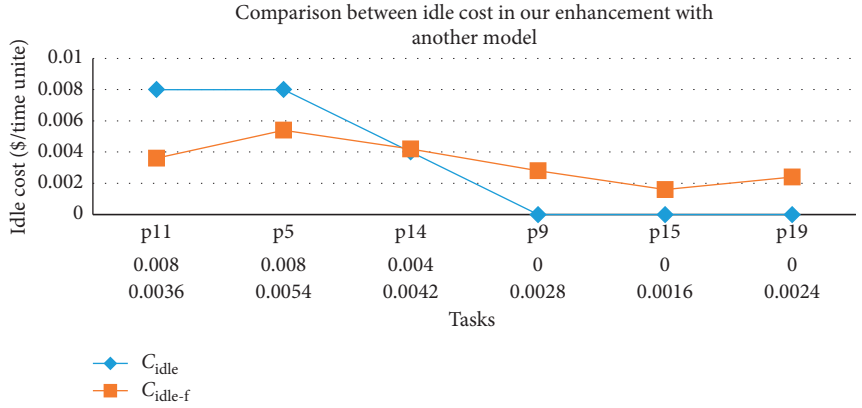


Figure 8: Comparison between the idle cost in the proposed model and the idle cost in another model at $t = 8$.

mid-task charges the maximum that is the two-thirds of the cost of one VM setup which equals 0.005$, and the short task charges the maximum that is the third cost of setting up of one VM which equals 0.003$.

5.2. Discussion of the Results of the Idle Cost That the Tasks Charge in Case 3. The enhancement of model [1] deals with the distribution process of idle cost. The objective of the proposed enhancement is to distribute the idle cost among tasks in a fair manner. This enhancement distributed the idle cost among three groups of tasks and distributes the idle cost in a group by SRTF or FTFC principle. The discussion of the results is as follows: the idle cost at any time $= t$ has been distributed among a large number of possible tasks rather than on a few of them. For example, in $t = 8$ the total idle cost that was distributed is 0.02$. This idle cost distributed by using the model in [1] among three tasks, but by using the proposed enhancement, it was distributed by six tasks (see Figure 8).

The idle cost that the tasks charge using this enhancement is fewer than the idle cost that the task charges by using the model in [1]. For example, in $t = 8$, the idle cost that the p11, p5, and p14 charge is 0.008$, 0.008, and 0.004$ using the model in [1], whereas by using the proposed enhancement, the idle cost is 0.0036$, 0.0054$, and 0.004$ in sequence (see Figure 8).

The ratio of the idle cost that the long task charges does not exceed 80%, the ratio of the idle cost that the mid- task charges does not exceed 84%, and the ratio of the idle cost that the short task charges does not exceed 80% (see Figure 7).

Table 7 summarizes, in general, the number of the tasks that have no delay. Also, the total cost was distributed using

Table 7: The results comparison with other models.

	With model (2013)	With our model
Number of tasks that have no delay	0	16
The enhancement rate of the delay	**0%**	**76.2%**
Total idle/setup cost that expanded	0.052$	0.02$
The cost rate expanded	24.7%	9.5%
Number of tasks charged by idle/setup cost	**3**	**6**
The enhancement rate in the distribution process of idle/setup cost	**14.3%**	**28.57%**

model in [9], using model in [1], and using our model through time duration [0, 8] (time unit) and Dataset = 21 task. In addition, Table 7 describes the enhancement ratio in our model and in the other.

The results in Table 7 mean the following:

(1) The best enhancement rate in the distribution process of idle/setup cost was occurred in our model compared with the model in [1] and the model in [9]

(2) The best enhancement rate in the delay that the tasks charged has occurred in our model and model in [1] compared with the model in [9]

6. Conclusions

Task scheduling without setup of cost that was proposed is an enhancement model for the clairvoyant algorithm in [9]. The proposed model activates the VMs early and runs them continually. The execution results showed that the contribution of this study performed in terms of delay and setup cost and the tasks charged is better than the delay and setup cost that the tasks charge produced in the model of [9].

This model distributed the idle cost of VMs among tasks in a fair manner. The execution results also showed that the enhancement of this study performed in terms of idle cost that the tasks charge is better than the idle cost that the tasks charge produced in the model of [1]. This study can be applied when to rent the VMs to schedule the tasks in cloud computing. This study reduced the delay and cost of setup of the VMs and distributed the idle cost instead of the setup cost in a more efficient way.

In the future, the researchers are supposed to study different functions, i.e., the cost and delay that the tasks charge in the case of heterogeneous tasks. Also, they are supposed to study the impact of cost and delay that the tasks charge when the VMs are heterogeneous; i.e., the VMs vary in, e.g., CPU Type, CPU Speed, cores, memory, renting cost, etc.

Data Availability

Data used in this paper have been generated by the authors based on standard conditions. A description of the data used in this paper has been presented in Section Inputs and Outputs of the Model.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

[1] R. Al-Dilami, A. Z. Al-Saqqaf, and A. Z. Zahary, "An enhanced algorithm for cloud scheduling with setup cost," *International Journal of New Computer Architectures and Their Applications*, vol. 8, no. 3, pp. 129–141, 2018.

[2] K. A. Deshmane and B. T. Pandhare, "Survey on scheduling algorithms in cloud computing," *International Journal of Emerging Trend in Engineering and Basic Sciences*, vol. 2, no. 1, pp. 210–215, 2015.

[3] B. Hayes, "Cloud computing," *Communications of the ACM*, vol. 51, no. 7, pp. 9–11, 2008.

[4] S. Selvarani and G. S. Sadhasivam, "Improved cost-based algorithm for task scheduling in cloud computing," in *Proceedings of the 2010 IEEE International Conference on Computational Intelligence and Computing Research*, Coimbatore, India, 2010.

[5] D. Karger, C. Stein, and J. Wein, *Scheduling Algorithms: Algorithms and Theory of Computation Handbook*, CRC Press, Boca Raton, FL, USA, 1999.

[6] K. Pruhs, J. Sgall, and E. Torng, "Online scheduling," *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, pp. 115–124, CRC Press, Boca Raton, FL, USA, 2003.

[7] M. M. Padmavathi, S. M. Basha, S. Mahabbob Basha, and M. S. Pothapragada, "A survey on scheduling algorithms in cloud computing," *IOSR Journal of Computer Engineering*, vol. 16, no. 4, pp. 27–32, 2014.

[8] J. Sgall, "On-line scheduling," in *Developments from a June 1996 Seminar on Online Algorithms: The State of the Art*, pp. 196–231, Springer, Berlin, Germany, 1998.

[9] Y. Azar, N. Ben-Aroya, N. R. Devanur, and N. Jain, "Cloud scheduling with setup cost," in *Proceedings of the 25th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 298–304þ, Montreal, Canada, 2013.

[10] Y. Chawla and M. Bhonsle, "Dynamically optimized cost-based task scheduling in cloud computing," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 2, no. 3, 2013.

[11] S. T. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," in *Proceedings of the 2013 IEEE INFOCOM*, Turin, Italy, 2013.

[12] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *Proceedings of the 2012 IEEE INFOCOM*, Orlando, FL, USA, 2012.

[13] A. Ingole, S. Chavan, and U. Pawde, "An optimized algorithm for task scheduling based on activity-based costing in cloud computing," *NCICT*, vol. 3, pp. 34–37, 2014.

[14] P. Anand and P. Goswami, "Cost based algorithm used in CloudSim," *IJEDR*, vol. 2, no. 3, 2014.

[15] A. Verma and S. Kaushal, "Cost-time efficient scheduling plan for executing workflows in the cloud," *Journal of Grid Computing*, vol. 13, no. 4, pp. 495–506, 2015.

[16] A. Allahverdi and H. M. Soroush, "The significance of reducing setup times/setup costs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 978–984, 2008.
[17] D. I. Esa and A. Yousif, "Scheduling jobs on cloud computing using firefly algorithm," *International Journal of Grid and Distributed Computing*, vol. 9, no. 7, pp. 149–158, 2016.
[18] M. J. Hazar, "Improving tasks scheduling in cloud computing," Master thesis, Mansoura University, Mansoura, Egypt, 2015.
[19] I. Elhossiny, N. A. El-Bahnasawy, and F. A. Omara, "Dynamic task scheduling in cloud computing based on the availability level of resources," *IJGDC*, vol. 10, no. 8, pp. 21–36, 2017.
[20] M. A. S. Mosleh, M. A. G. Hazber, and S. H. Hasan, "Adaptive cost-based task scheduling in cloud environment," *Scientific Programming*, vol. 2016, Article ID 8239239, 9 pages, 2016.
[21] H. Chen, X. Zhu, G. Liu, and W. Pedrycz, "Uncertainty-aware online scheduling for real-time workflows in cloud service environment," *IEEE Transactions on Services Computing*, p. 1, 2018.