An Enhanced Algorithm for Cloud Scheduling with Setup Cost

Redwan Al-Dilami[1], Adnan Zain Al-Saqqaf[2], Ammar Thabit Zahary[3]

[1]Lecturer. Faculty of Computing and IT, University of Science and Technology, Sana'a, Yemen
[2] Associate Prof. Faculty of Engineering, Aden University, Aden, Yemen
[3]Assistant Prof. Faculty of Computer and IT, Sana’a University, Sana'a, Yemen
[1] r.aldilami@ust.edu, [2]adnan_zain2003@yahoo.com, [3]zahary@su.edu.ye

ABSTRACT

The study aimed at investigating the problem of online task scheduling of identifying job of MapReduce on cloud computing infrastructures. It was proposed that the virtualized cloud computing setup comprised machines that host multiple identical virtual machines (VMs) under pay-as-you-go charging and booting a VM requires a constant setup time. The problem occurs when the VMs turning off after finishing processing a task and running it again for another. The goal is to limit the delay resulted from setting up the VMs, and to minimize the idle cost (VM does not find any task to possess) when VM is continuing in an active state without processing tasks. It was a constant number of VMs in an activation state continuity, and the cost of idle was distributed over existing tasks where the cost should be less than the cost of setting up one VM. The researchers' algorithm limited the delay resulted from setting up the VMs, the cost resulted from the continuing VMs in an active state without processing tasks was distributed to the current tasks fairly. The known cases was discussed where the duration of each task was known upon its arrival.

KEYWORDS

Cloud computing, clairvoyant, scheduling, Known Duration of Tasks scheduling, setup cost.

1. INTRODUCTION

Scheduling is a collection of policies and mechanisms for the sake of controlling and ordering the work of the computer system.Task scheduling is one of the basic topics discussed by many researchers towards solving the problem of scheduling. The researchers introduced a number of techniques for the sake of solving this problem [1].

Reducing the response time for cloud services will increase the confidence of customer that requests cloud services and the great motivation is minimizing the total cost of job processing.

The goal is to limit the delay resulted from setting up the VMs, and the cost resulted from continuing VMs in an active state without processing tasks was distributed to the current tasks fairly. In online scheduling, the scheduler receives jobs that arrive over time, and generally must schedule the jobs without any knowledge of the future. 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 reducing the capital and management costs of hardware and software resources [2]. Public cloud is available in a pay-as-you-go charging model that allows end-users to pay for VMs by the hour e.g., $0.12 per hour. Two-key criteria determine the quality of the provided service: (a) the dollar price paid by the end-user for activating VMs and (b) the maximum delay among all given tasks of a job which occurred by setup the VM to become active. The goal is to provide a scheduling algorithm that aims to minimize the delay and the activation's setup cost of VM.

This paper focuses on arbitrary jobs such as job identified of MapReduce. In classical scheduling problems, the number of machines is fixed, and the sequence we have to decide which job to process on which machine. However, the cloud introduces a different model in which we can activate and release machines on demand, and thus control the number of machines being used to process the jobs. This highlights the tradeoff between the number of machines used and the delay of processing the jobs. On the one hand, if the researchers does not have to pay for each machine, they could use one machine for each task of the job, and reduce the delay to a minimum. On the other hand, if we want to minimize cost, we could only use a single machine for all tasks of the jobs in a work-conserving model.

In this paper, it is assumed that all computing instances available for processing are initially active. When the jobs arrive, they inter to the VM without waiting to activate the VM. Moreover, when a VM is no longer in use, it should not be shut down, a constant time for turning off will not occur. Both setup and shutdown times are not included in the production cost of this service. Therefore, they will not be charged to the end-user. As a result, the number of VMs activated continuously (without shutdown) for a specific job which has a major impact on the total cost.

The goal is to minimize both maximum delay (setup time) and setup cost. The problem lies in finding the right balance between delay and cost. In this study, a number of VMs are assumed in activation state continuously, so that no need to setup and shutdown 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.

The remaining sections of this paper are organized as follows:

Section 2 presents the related work to setup cost scheduling. System model, system assumption, cost scheduling model, and algorithm of known duration of tasks are presented in Section 3. The results of the study and comparison are presented in Section 4 and 5, and the conclusion is presented in Section 6.

2. RELATED WORK

Due to the significance of task scheduling and being often NP-hard, these kinds of problems have been much studied. The instrument technique was surveys for scheduling algorithms and online scheduling which are found in [3],

[4], [5] and [6]. It could be the most intuitive measure of Quality of Service (QoS) received by an individual job is the flow time. That is, the flow time Fi of the i th job from the difference between its completion time and its release date. However, this measurement is used to the delay attribute in [7]. The main goal in the last reference is to give a scheduling algorithm that aims to minimize the delay and the production cost of executing a job. They assume that all computing instances available for processing are initially inactive to assign a task to any machine; It should be activated first, activation (Set up) Time: $\mathrm{T_{setup}}$. To activate a machine, there is a constant duration of setup until the machine is ready to process, Shut down Time: $\mathrm{T_{shutdown}}$. This means that Activation Time $\mathrm{T_s = T_{setup} + T_{shutdown}}$, Both setup and shutdown times are involved in the production cost of this service, and therefore they will be charged to the end-user. In the researchers' work, the focus is on the online problem because no information is known on future arrival of tasks, but that the arrivals of tasks are independent of the scheduling. Focusing on the cases below, it is clear that they are known and unknown task duration model (i.e. clairvoyant and non-clairvoyant). In the clairvoyant case the duration of a task is known at its arrival. In the non-clairvoyant model the duration is unknown at its arrival and becomes known only once the task has been completed. The results of the researchers' work (1) the cost ratio of Algorithm Clairvoyant is at most$(1+\epsilon)$; (0< ϵ <1). Moreover a long task will have a delay of $\mathrm{T_s}$. The delay of a short task is at most $2\mathrm{E} = \frac{4\mathrm{T_s}}{\varepsilon}$. (2) If an online non-clairvoyant algorithm is limited to a cost of $(1+\epsilon)$, then its delay ratio is $\Omega\left(\frac{\log\mu}{\varepsilon}\right)$, where μ is the ratio of the longest task duration of any job to the shortest task duration of any job.

Known Duration of Tasks: It is assumed that the duration of each task $\mathrm{p_i}$ is known upon its arrival. Let$\mathrm{E} = \frac{2\mathrm{T_s}}{\epsilon}$ (for $0 < \epsilon < 1$).

Algorithm Clairvoyant: Classify a task upon its arrival. It is a long task if $\mathrm{p_i} \leq \mathrm{E}$ or otherwise short.

Once the arrival of each new long task, activate a new machine. That is to say, accumulate short tasks, activate a machine to process those tasks at the earliest between case 1: the first time the volume of the accumulated tasks becomes above E . In this case, assign the tasks to a new allocated machine and restart the accumulation. Case 2: $(\mathrm{E}-\mathrm{T_s})$ time passed from the earliest release time of those tasks. In this case, continue the accumulation until the machine is ready (at timeE), assign the tasks to the machine, and then restart the accumulation. If the volume of the tasks exceeds E by the time the machine is ready, stop the accumulation and go to case 1 (these tasks will be classified as case 1). Processing the tasks on their assigned machine according to their arrival order. Note that the volume assigned to a machine is below 2E and each task will start its processing within at most E time after the assigned machine is ready. Shutdown a machine once it completes tasks assigned to it.

Unknown Duration of Tasks: Algorithm Non-Clairvoyant Divide the time into epochs of $F = \frac{4T_s}{\varepsilon}$. Let B_0 be the set of tasks given initially

(Time 0) and for $k \geq 1$, let $B_k = \{i | (k-1) \leq a_i \leq kF\}$. All tasks B_k are handled at time kF separately from tasks of $B_{k'}$ for $k' \neq k$. Let $n_k = |B_k|$ is the number of tasks arrived in epoch k. Let $m_k = \left\lceil \frac{n_k p}{F} \right\rceil$ and activate m_k machines. Processing tasks on machines in arbitrary order for $T_s + F$ time (this also includes setup times of newly activated machines). If after $T_s + F$ time there are still waiting tasks then activate additional m_k machines set $m_k \longleftarrow 2m_k$ and repeat this step (note that tasks that are already running will continue to run with no interruption). Shutdown a machine once it becomes idle and there are no waiting tasks.

In [8] files are divided into many small blocks and all blocks are replicated over several servers. To process files efficiently, each job is divided into many tasks and each task is allocated to a server to deal with a file block because network bandwidth is a scarce resource.

Activity Based Costing as a Solution: Activity based costing is evaluated separately for every task. It is decided on the basis of resources, space and time taken by every activity of every task.

Activity Based Costing in Cloud Computing: One way to measure both cost of the object and its performance is the activity based costing. In this view, researchers have solved the problem like poor cost control, distorted product costs and also the starvation. They divide the task into different groups. These groups are (1) Available (dependent & independent): is the group of tasks which can be complete performed on a single data center. (2) Partially available: is the group of tasks which need resources from other data centers. Hence others classify them into cat1, cat2, cat3… and so on till N number of classification. They are done on the basis of data need. Cat1 tasks will need the data from same data centers. Similarly, cat2, cat3….cat N tasks will need the data from same data centers.

Activity Based Costing (Implementation): Researchers deal with algorithm in a tree diagram. one parent queue in which researchers stored the tasks according to their arrival time, tested for data and requested resources by the task and sorted them into two different queues, available and partially available. For available queue, they tested if the task is dependent or independent and according to that researchers stored them in their respective queues. For partially available queue, they sorted tasks in a multiple queues named cat1, cat2… and so on. To decide the priority, researchers considering four major factors i.e. time, space, resources and profit, and they have derived the following formula for deciding the priority of the task: $\sum_{j=0}^{n}(T_{i,j} + s_{i,j} + C_{i,j})/P_i$. The notations are explained below:

Ki : Priority of the ith task.

Ti, j: Time required to complete jth activity of ith task.

Si, j: Space needed to operate jth activity of ith task.

Ci, j: cost of jth activity in terms of resources of $i\ th$ task.
Pi : Profit from complete $i\ th$ task.
n : It is the total number of activities of any ith task.

In [9], the goal of this paper is to schedule task groups in cloud computing platform, in that resources have different resource costs and computation performance. However, researchers' algorithm measure both resource cost and computation performance, it also develops the computation/communication ratio by collecting the user tasks based on a particular cloud resource's processing capability and sends the grouped jobs to the resource. They improved Activity Based Costing (ABC)**,** their problems are reduction of makespan and reducing cost. They used CloudSim 1.0b to simulate the algorithm of task scheduling, simulated the algorithm with six nodes, five seconds of granularity time, average MI (Number of Machine Instructions) of tasks 10. They can be seen that for ABC Scheduling the time taken to complete tasks after grouping the tasks is very less when compared with time taken to complete the tasks without grouping the tasks.
In [10], researchers introduce Budget and Deadline Constrained Heuristic based upon Heterogeneous Earliest Finish Time (HEFT) to schedule workflow tasks over the available cloud resources.
In [11], researchers propose a scheduling algorithm which addresses these major challenges of task scheduling such as task completion time or task execution cost etc. The proposed model is implemented and tested on simulation tool kit. Results validate the correctness of the framework and show a significant improvement over sequential scheduling.
In [12], the goal of this study is to use the conventional scheduling concepts to merge them to provide solution for better and more efficient task scheduling which is beneficial to both user and service provider.
In [13], researchers introduce a more efficient algorithm for task scheduling based on Priority Based Scheduling in cloud computing and the implementation of it. Improvement of this algorithm should concentrate on discussing simultaneous instead of independent task scheduling in Cloud environment.

3. SYSTEM MODEL AND ASSUMPTION

3.1 System Assumption

The job input consists of multiple tasks that need to be executed. Tasks arrive over time. A task i has an arrival time a_i and maximum duration p_i, assuming the arrival time is known.
At time t, there is constant number of VMs in activation state. The cost activation is free for the first time, I.e. cost activation for each VM in the second time is with fees. Time activation for each VM is a constant (T_{setup}), the delay is denoted by ($\mathrm{D_{setup}}$) which occurred by T_{setup}, and the cost activation of each VM is constant ($\mathrm{C_{setup}}$). Each VM and its task are homogeneous. Each task runs on a single

machine (instance). Each machine can run a single task at a time. Tasks are non-preemptive, i.e., a task has to run continuously without interruptions. Let $e_i = a_i + p_i$ which is the earliest possible completion time of task i. Denoted by c_i which is the actual completion time of task i and $d_i = c_i - e_i$ as the delay that the task delays for some time to find empty VM. Machine can be activated or shut down for the first time without fees otherwise with fees (i.e. activation machine again with fees). Any machine in 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. For simplicity, it is assumed that there is only activation time Ts = T_{setup} + $T_{shutdown}$ and the shutdown is free. This paper focuses on the known task duration model (duration of a task is known at its arrival).

3.2 Cost Scheduling Model

At any time t, the algorithm needs to decide how much the actual idle time in machines to shut down some of them. In addition, it should decide for each task what the cost idle time to be distributed.

Goal Function:

The goal function consists of two parts: setup cost and delay that occurred by setting up VM. It is assumed that the cost charged per machine per unit of idle time (T_{idle}) is \$ 0.002. Then the actual cost for each VM which is in idle state (C_{idle}) is:

$$C_{idle} = \sum_{i} VM_{idle} i * 0.002 \qquad (1)$$

Through equation No. (1), the total idle cost (C_{idle}) of VMs can be calculated for every task for every moment. Then, that cost is distributed to current available tasks according to our model. When all VMs are decided to be in activation state, an algorithm will be available, its idle time cost should be less than cost of its setup when activation of VM occurs again. Let D be the maximum delay of the task when it waits to activate a new machine (i.e. $d_i < D$ for all tasks i). Formally, the performance of an algorithm will be described by α- the cost ratio of our algorithm (C_{opt}) to the setup cost (C_{setup}):

$$\alpha = \frac{C_{opt}}{C_{setup}} \qquad (2)$$

Where C_{setup} *is* constant.

Through equation No. (2), the cost average ratio (α) can be calculated for every task via the researchers' algorithm (C_{opt}) comparing cost resulted from activating VM again (C_{setup}).

and δ- the delay ratio:

$$\delta = \frac{D_{opt}}{D_{setup}} \qquad (3)$$

where D_{setup} is constant.

Through equation No. (3), the delay average ratio (δ) can be calculated for every task via the researchers' algorithm ((D_{opt}) comparing delay resulted from activating VM again (D_{setup})

3.3 Algorithm of Known Duration of Tasks

In this section, the researchers' algorithm of known duration of tasks is conducted. The researchers first assume 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 $Ts \leq pi < E$ and otherwise it will be short.

Step 1

The researchers' determined the No. of VMs which are turning on continually (N_{vm}) during the period [0,t], running on the VMs will be free for the first time.

Step2

A number of tasks of known duration are generated (for each task, certain characteristics, arrival time (T_a), service time (T_s), terminate time (T_t) …etc. Which occur in separated times during the [0, t].

Step3

The tasks are processed in the VMs as follows:

1. If the number of the tasks (N_{task}) in the period [0,t] is equal to the number of the VMs which are in the turning on state; that is $N_{vm} = N_{task}$, each task enters VM will enter and its setup cost will be equal to zero. However, the proper delay for setting up the VM will be zero because the VMs are already active:

$$D_{setup} = 0$$
$$C_{setup} = 0$$

2. If ($N_{task} > N_{vm}$), then additional VMs (N_{vm-add}) that will be running equal $N_{task} - N_{vm}$ (i.e. $N_{vm-add} = N_{task} - N_{vm}$), and the tasks of the VMs will process as the task of the longest size will be the first.
3. If ($N_{task} < N_{vm}$), there is a definite number of these VMs are idle. In this case, the time of idle state of all VMs should be gathered: $T_{idle} = \sum_i VM_{idle} i$, and the cost of idle (C_{idle}) will be distributed over existing tasks in accordance with the following principles:
   a- Long task ($pi \geq E$) should cost less than the cost of setting up one VM.
   b- Middle task ($Ts \leq pi < E$) should cost less than the two-third cost of setting up one VM.
   c- Short task (0$< pi < Ts$) should cost less than the half cost of setting up one VM.

4- The case (3) continues until the total idle cost of VMs is greater or equal to the total cost of their setup. Hence, some VMs will be shut down in the principle of what gets idle first, should be shutdown first.

Step 4

Any VM gets turned on for the second time (i.e. activating it after being shut down), it should be turned on in line with algorithm clairvoyant in [7]. The following chart summarizes what is mentioned above:

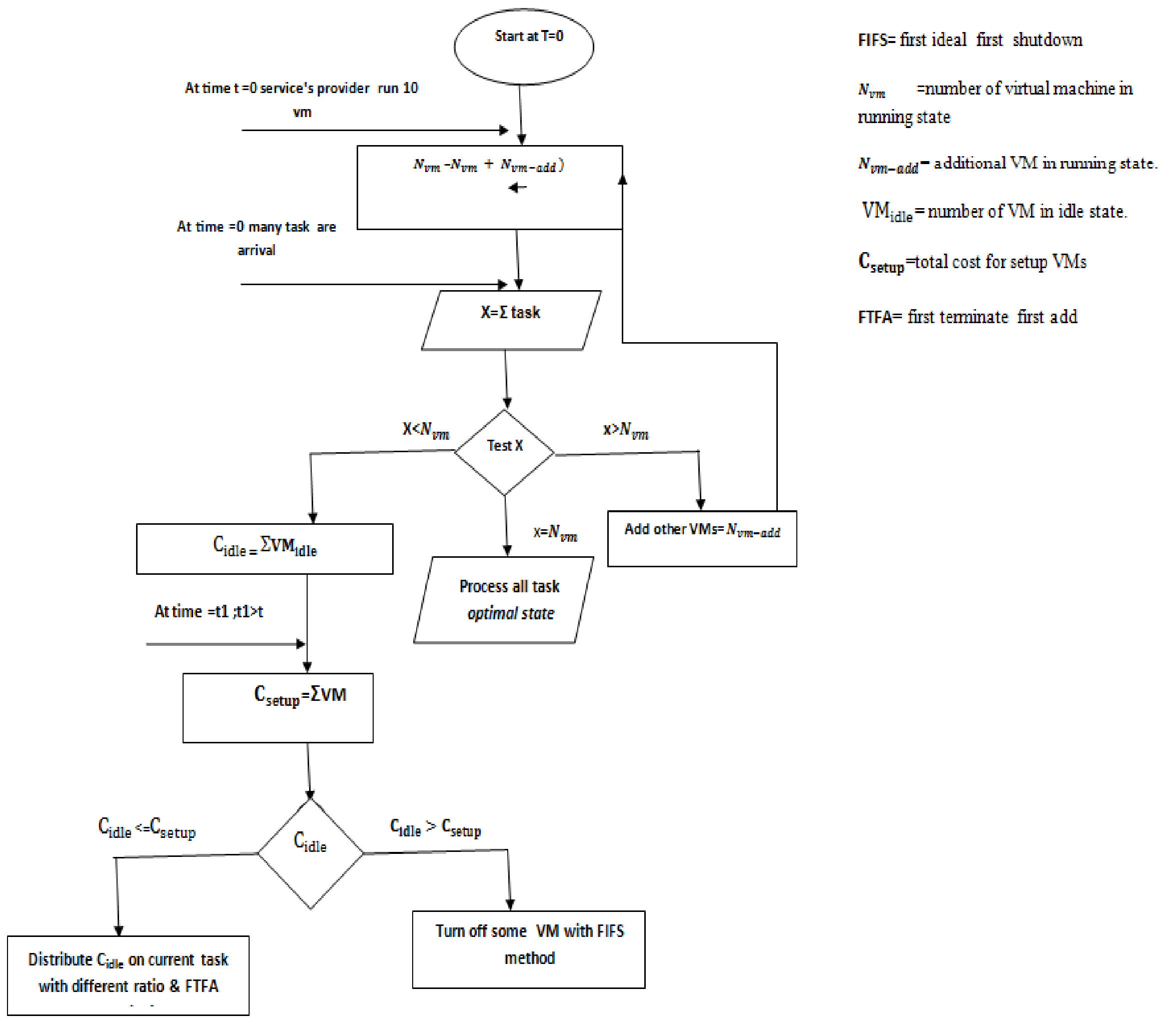

Start at T=0
At time t =0 service's provider run 10 vm
N_{vm} -N_{vm} + N_{vm-add})
At time =0 many task are arrival
X=Σ task
X<N_{vm}
Test X
x>N_{vm}
x=N_{vm}
Add other VMs=N_{vm-add}
C_{idle} = ΣVM$_{idle}$
Process all task optimal state
At time =t1 ;t1>t
C_{setup}=ΣVM
C_{idle} <=C_{setup}
C_{idle}
C_{idle} > C_{setup}
Distribute C_{idle} on current task with different ratio & FTFA
Turn off some VM with FIFS method
FIFS= first ideal first shutdown
N_{vm} =number of virtual machine in running state
N_{vm-add}= additional VM in running state.
VM$_{idle}$= number of VM in idle state.
C_{setup}=total cost for setup VMs
FTFA= first terminate first add

Figure1: Algorithm Diagram Model

4. STUDY RESULTS

The performance of an algorithm will be described by a competitive analysis where α is the cost ratio of the algorithm to the setup cost ($\alpha = \frac{C_{opt}}{C_{setup}}$) and δ ($\delta = \frac{D_{opt}}{D_{setup}}$) is the delay ratio:

1- In optimal case (I.e. $N_{task} = N_{vm}$ through time duration [0, t]) α=0 and the delay ratio δ=0

2- In the second case (when $N_{task} > N_{vm}$ through time duration [0, t]) α=0 for all jobs and additional tasks (cost activation of additional VMs is free in the first time).

Delay ratio:

$$\delta = \begin{cases} 1 & \textit{for addional task} \\ 0 & \textit{otherwise} \end{cases} \quad (4)$$

3- In third case (when $N_{task} < N_{vm}$ through time duration [0, t]):

$$\begin{cases} 0 \leq \alpha \leq 1 & for\ longest\ task\ duration. \\ 0 \leq \alpha \leq \frac{2}{3} & for\ middle\ task. \\ 0 \leq \alpha \leq \frac{1}{3} & for\ shortest\ task. \end{cases} \quad (5)$$

Delay ratio δ=0 (I.e. no arrival additional tasks to activation additional VMs).

As shown by figure 2, when the number of the tasks is equal to the number of the VMs ($N_{vm} = N_{task}$) through time duration [0,t]. That is, as far as the VM finishes processing a task, another task comes, and the VM will not turn off nor activate again. There is no delay for any neither task nor setup cost because running the VM occurs for the first time. This case is called "optimal". Notice the curves of cost and delay do not appear. In figure 3, when ($N_{task} > N_{vm}$) during the period [0, t], this case is considered rare because the problem occurs when the opposite happens. That is, services providers run a number of VMs for the additional tasks. The delay occurs because it waits for the VM to become ready as being closed earlier. However, these tasks without setup cost because these VMs occur for the first time, as shown in p15, p16, p18, p19 and p20. Notice the curves of VMs, delay appears only in the additional tasks

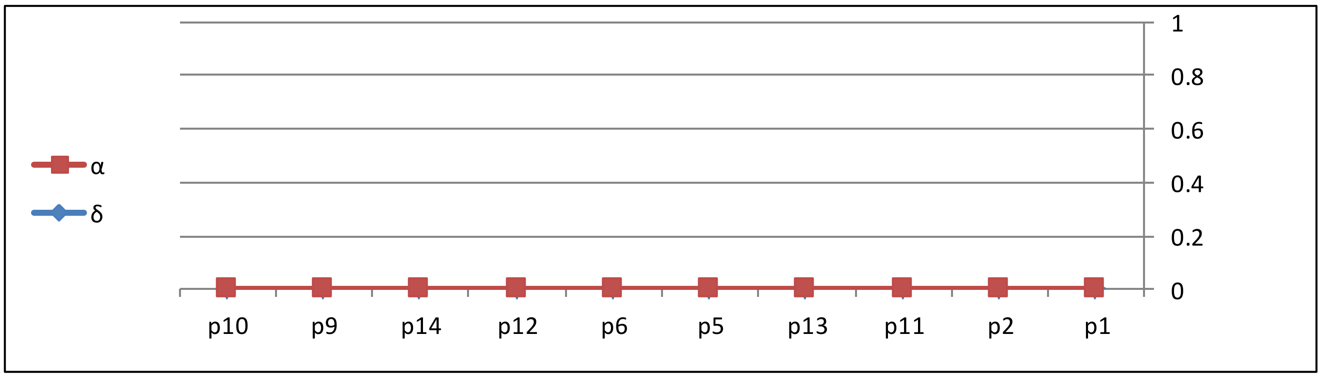


Figure 2: Optimal Case

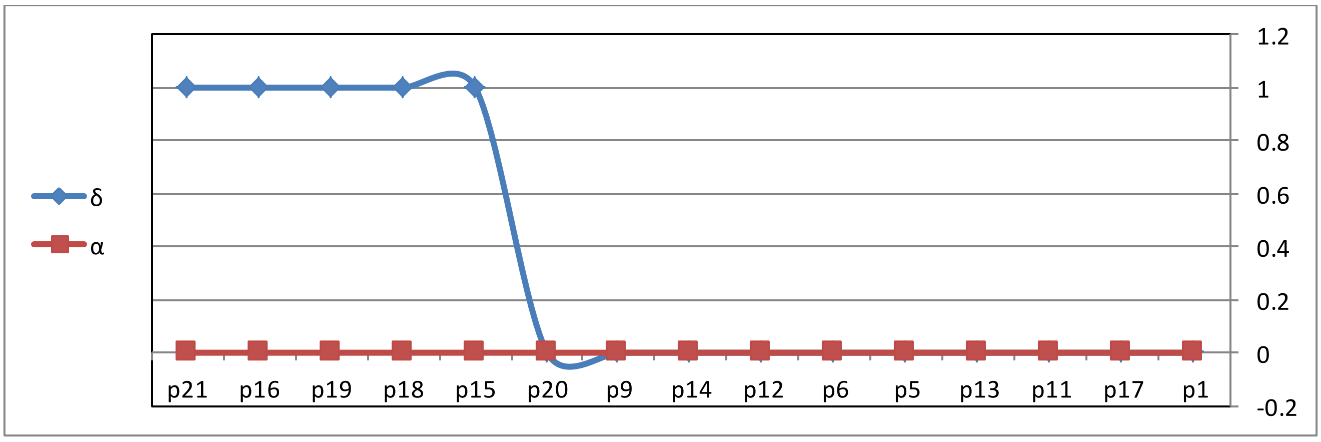


Figure 3: Delay with Additional Tasks

In figure 4, when ($N_{task} < N_{vm}$). (I.e. some VMs are idle), so, instead of turning off such VMs, the researchers should count its remaining cost when they are turning on. This cost gets distributed over all tasks during that period.

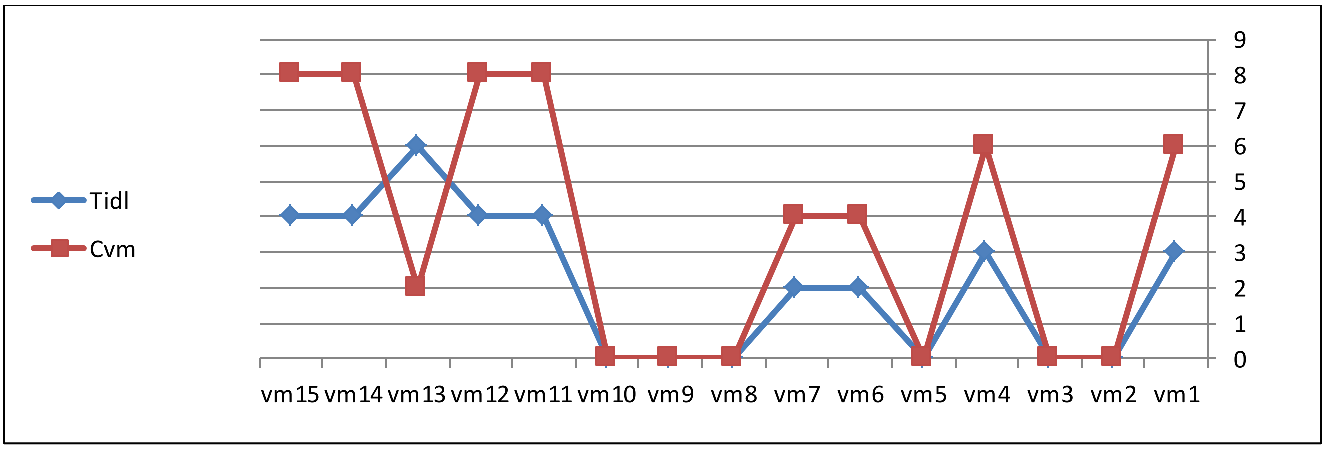


Figure 4: Cost Distribution

In table 1, it is noticed that the unit of time t=10, the total of idle cost VMs from VM1 to VM15 equals to 0.11. Thus, it shut down some VMs. It is noticed that, in the critical case, the long task will render a setup cost of one VM. In case of others, every task will render a setup cost less than that of one single VM such as VM1, VM4, VM5, VM6 etc. until the total of idle cost becomes less than the cost of activating all VMs again. Moreover, in all cases, there is no delay resulted from the time of VM's setup.

Table 1: Shutdown Some VMS

N-vm	N-task	Ta	Ts	VM_{idle}	C_{idle}	δ	α
vm1				5	0.01	0	
vm2	p2	4	8	0	0	0	0.001
vm3	p3	2	8	0	0	0	0.001
vm4				5	0.01	0	
vm5				2	0.004	0	
vm6				4	0.008	0	
vm7				4	0.008	0	
vm8				1	0.002	0	
vm9				2	0.004	0	
vm10	p10	4	6	0	0	0	0.0009
vm11				6	0.012	0	0
vm12				6	0.012	0	0
vm13				8	0.016	0	0
vm14				6	0.012	0	0
vm15				6	0.012	0	0

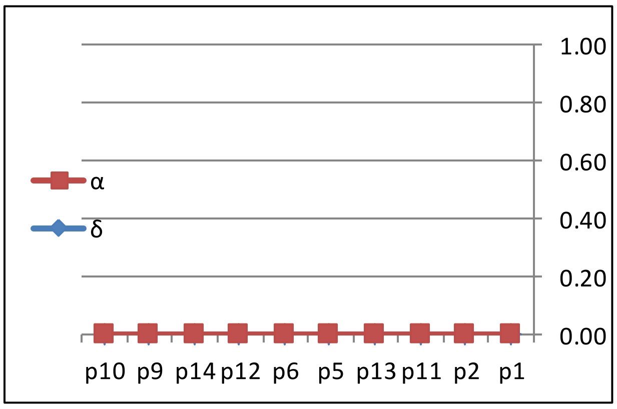
In all cases, it is supposed that the cost of one single time unit of the idle equals the double cost of setup unit, and the activation cost of VM is counted but the turning off is free.

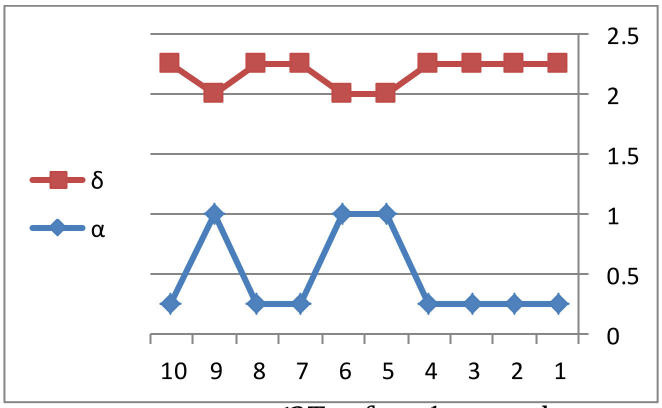
5. COMPARISON OF RESULTS

When our data is uses in others algorithms the δ and α appear as the following :

In the researchers' algorithm | In others

Case 1: $N_{task} = N_{vm}$



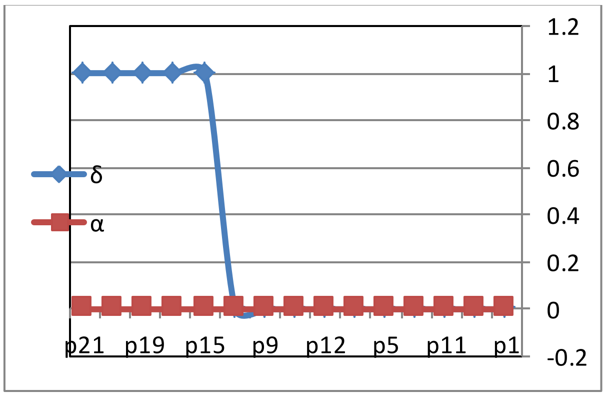


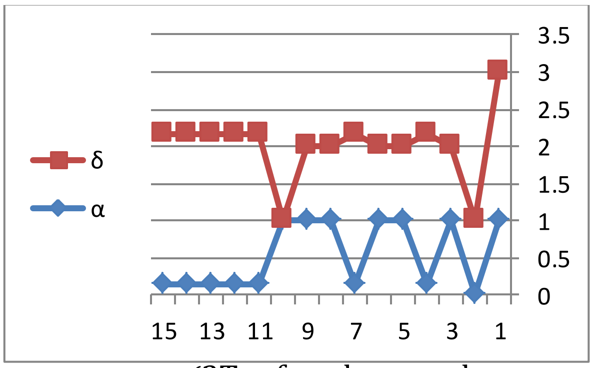
Ratio of delay (δ)=0
Ratio of setup cost (α)=0

$$\text{Ratio of delay}(\delta) = \begin{cases} 2Ts & \text{for short task} \\ Ts & \text{for lareg task} \end{cases}$$

$$\text{Ratio of Setup cost}(\alpha) = \begin{cases} 0 < \alpha < 1 & \textit{forshort task} \\ 1 & \text{for lareg task} \end{cases}$$

Case2: $N_{task} > N_{vm}$



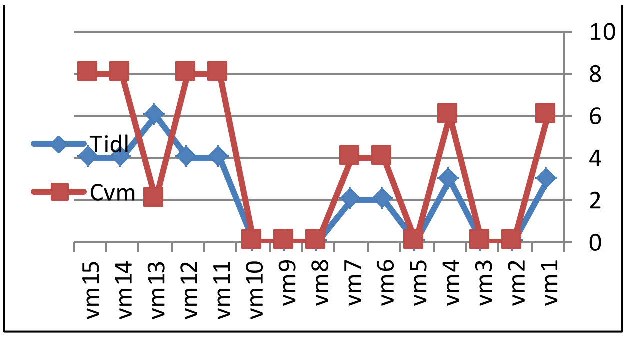


Delay $=T_s$ only with additional task.
No setup cost for all task.

$$\text{Delay}(\delta) = \begin{cases} 2Ts & \text{for short task} \\ Ts & \text{for large task} \end{cases}$$

$$\text{Ratio of Setup cost}(\alpha) = \begin{cases} 0 < \alpha < 1 & \textit{for short task} \\ 1 & \text{for large task} \end{cases}$$

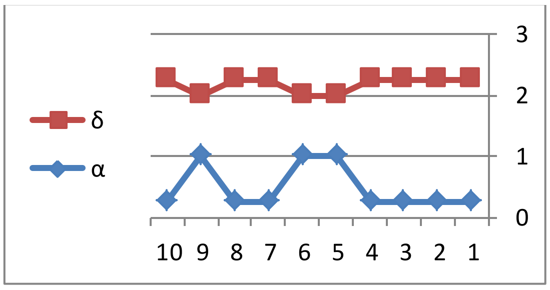
Case 3: $N_{task} < N_{vm}$



Delay =0

Idle Cost (Cvm) =

$$\begin{cases} 0 \le Cvm \le 1 \quad for\ longest\ task \\ 0 \le Cvm \le \frac{2}{3} \quad for\ middle\ task. \\ 0 \le Cvm \le \frac{1}{3} \quad for\ shortest\ task. \end{cases}$$



$$\text{Delay}(\delta) = \begin{cases} 2\text{Ts} \quad \text{for short task} \\ \text{Ts} \quad \text{for large task} \end{cases}$$

No idle cost but Ratio of Setup cost(α) =

$$\begin{cases} 0 < \alpha < 1 \, for\ short\ task \\ 1 \quad \text{for large task} \end{cases}$$

Figure 5: Comparison of Results

6. CONCLUSION AND FUTURE WORK

In this paper, the researchers devised a new scheduling algorithm for cloud environment. The proposed algorithm is a generalization of clairvoyant algorithm with known duration of tasks. the researchers' algorithm is limited the delay occurs by activation and shutdown VMs. Also it distributed the idle cost of VMs over all task in a fair manner. The simulation results show that the researchers proposed algorithm out performs in terms of setup delay while producing the setup cost as good as produced by clairvoyant algorithm. In future, the researchers intend to improve this work by finding out the optimal schedule plan to decrease the idle cost of VMs.

REFERENCES

1. Gahlawat, M., Sharma, P.: Analysis and Performance Assessment of CPU Scheduling Algorithms in Cloud using Cloud Sim. International Journal of Applied Information Systems (IJAIS), Foundation of Computer Science FCS, New York, USA, 5-8 (2013).

2. Hayes, B. : Cloud computing. Communications of the ACM, 51(7): 9-11, 2008.

3. Karger, D., Stein, C., Wein, J.: Scheduling algorithms: Algorithms and Theory of Computation Handbook (1997).

4. Pruhs, K., Sgall, J., Torng, E.: Online scheduling.: pp. 115- 124 (2003).

5. Padmavathi, M., MahabbobBasha, S.,Pothapragada, S. : A Survey on Scheduling Algorithms in Cloud Computing : IOSR Journal of Computer Engineering (IOSR-JCE), 16(4), 27-32 (2014).

6. Sgall, J.: On-line scheduling. In: Developments from a June 1996 seminar on Online algorithms: the state of the art, pp 196- 231(1998).

7. Azar, Y., Ben-Aroya, N., Devanur, N.: Cloud Scheduling with Setup Cost. In proc. SPAA'13, June 23–25, Canda (2013).

8. Ingole, A., Chavan, S., Pawde, U.: An optimized algorithm for task scheduling based on activity based costing in cloud computing. In Proc. 2011 (NCICT) published in International Journal of Computer Applications. (IJCA). 34-37 (2011).

9. Selvarani, S., SudhaSadhasivam, G.: Improved cost-based algorithm for task scheduling in cloud computing. In Proc. International Symposium on Cloud and Services Computing, IEEE, 1-5(2010).

10. Verma, A., Kaushal, S.: Cost-Time Efficient Scheduling Plan for Executing Workflows in the Cloud. Journal of Grid Computing. Springer Science, 493-506 (2015).

11. Choudhary, M., Peddoju, S.: A Dynamic Optimization Algorithm for Task Scheduling in Cloud Environment. International Journal of Engineering Research and Applications (IJERA). 2 (3), 2564-2568 (2012).

12. Chawla, Y., Bhonsle, M.: Dynamically optimized cost based task scheduling in Cloud Computing: (ITETTCS). 2 (3), 38-42 (2013).

13. Anand, P., Goswami, P.: Cost Based Algorithm Used In CloudSim. IJEDR. 2(3), 3112- 3116 (2014).