International Journal of Grid and Distributed Computing
Vol. 10, No. 8 (2017), pp.21-36
http://dx.doi.org/10.14257/ijgdc.2017.10.8.03

Dynamic Task Scheduling in Cloud Computing Based on the Availability Level of Resources

Elhossiny Ibrahim[1], Nirmeen A. El-Bahnasawy[2] and Fatma A. Omara[3]

[1,2]Computer Science & Eng. Dept., Faculty of Electronic Eng. Menouf 32952, Egypt
[3]Faculty of Computers & information, Cairo University, Egypt
[1]elhossiny.ibrahim@el-eng.menofia.edu.eg, [2]nermeen.abd @el-eng.menofia.edu.eg, [3]f.omara@fci-cu.edu.eg

Abstract

Cloud Computing is a business model that based on "pay as you go" principle. It is used to provide the IT services to the user in the flexible and dynamic manner with minimal management effort. The most important feature of the Cloud Computing is the ability to dynamically schedule the application on the best resource according to the load. According to the work in this paper, a task schedule on the Cloud environment has been proposed. The principle of the proposed algorithm is 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 processing power, cost, and the number of running tasks to know its fitness to receive the incoming task, then choosing the best one to the incoming task. To evaluate the performance of the proposed algorithm, a comparative study has been done between this proposed algorithm, Round Ribbon (RR) algorithm, and Minimum Completion Time (MCT) algorithm. The experimental results show that the proposed algorithm outperforms the RR, and MCT algorithms by reducing make-span and the cost of the running tasks.

Keywords*: Cloud Computing; Dynamic Task scheduling; Service Request; Service Provider; Cloud Pricing*

1. Introduction

Virtualization is considered the core of the Cloud Computing which hides the heterogeneity of the resources by introducing its services in the form of shared pool of configurable resources called Virtual Machines (VMs) [1].The main principle of the Cloud Computing is to deliver different IT services to the customers which meet their requirements [2].

In order to service a user request, the service provider needs either to pay new hardware resources or to rent it from the resource provider. According to the Cloud Computing, the service provider hires resources from the resource provider and introduces the required resources in the form of Virtual Machine (VM) instances which meet the consumers' requirements. The resource provider is responsible for allocating the VMs on the physical server. When a user submits a task to the Cloud, it passes through the following steps to be scheduled on the VMs (see Figure 1):

- Resource discovering and filtering- Datacenter Broker discovers the resources present in the Cloud and collects status information related to them.
- Resource selection - Target resource is selected based on certain parameters of task and resource. This is considered deciding stage.
- Task submission -Task is submitted to the selected resource.

ISSN: 2005-4262 IJGDC

The load balance is considered one of the main principles of the optimized scheduling [3].

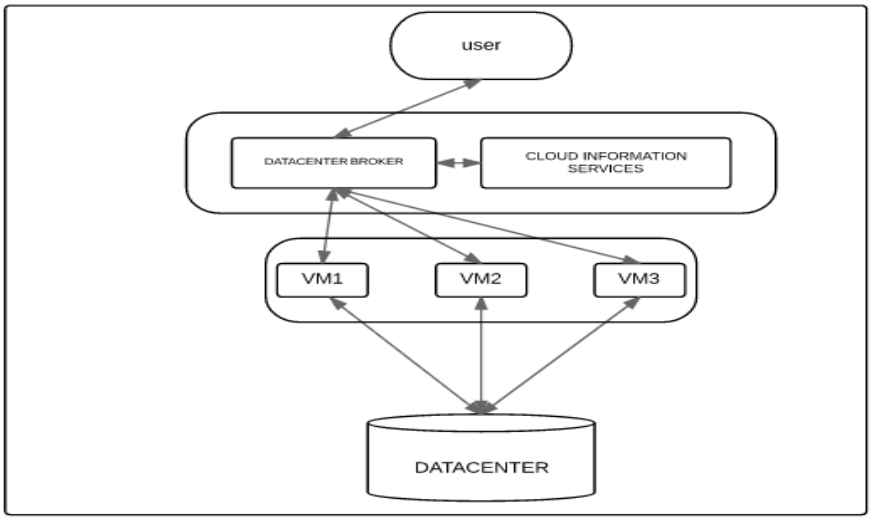

Figure 1. Scheduling in the Cloud[3]

There are two types of task scheduling in the Cloud Computing; static task scheduling, and dynamic task scheduling. Static scheduling allows for pre-fetching the required data and pipelining all the scheduling parameters at the beginning. Static scheduling imposes less runtime overhead. In case of dynamic scheduling, the information of the job components/task is not known beforehand. Thus, the execution time of the task may not be known and the allocation of tasks is done on the fly [4]. The proposed algorithm is focused on how dynamically schedule the new incoming tasks during the execution of other tasks by choosing the best available resource (*i.e.*, VM). This will be done by calculating the availability level of all VMs and allocating the incoming task to the most available one.

The remainder of the paper is organized as follows. In Section 2, the related work is illustrated followed by the fundamental of the proposed task-scheduling algorithm in Section 3. In Section 4, the performance evaluation of the proposed algorithm via Cloudsim simulator and the comparative study are discussed. Finally, in Section 5, conclude the contributions and point out the future work.

2. Related Work

A greedy task scheduling based on the priority undertaken for its computation has been existed [5]. According to this algorithm, the deadline-based tasks are prioritized based on task deadline. The tasks with shorter deadline are executed first. So, the tasks are sorted in ascending order according to its priority, and the highest priority tasks are executed first. Greedy algorithm is suitable for dynamic heterogeneous resource environment connected to the scheduler through homogeneous communication environment [6]. R. K. Sharma, *et al.*, [3] have proposed a dynamic optimization algorithm for task scheduling in Cloud Computing with resource utilization. The experimental results satisfy the requirements of both the users and the service providers through efficient schedule and priority reassignment. Liang Ma1, *et al.*, [7] have proposed a dynamic task scheduling in Cloud Computing based on greedy strategy. The experimental results show that the algorithm can dynamically allocate VMs to reach load balance rate and rapidly finish executing tasks by greedy strategy.

On the other hand, Round-Robin (RR) is a simple scheduling algorithm, based on time membership between jobs in circular queue without precedence, so, it is simple and easy to implement. But, each node is selected with a time slice in which it has to execute its task.

This algorithm simply allocates the job in round robin manner. The benefit of this algorithm is that no work has to wait for another one to be completed as on FCFS basis [8]. The main drawback of the RR algorithm is that the load on different machines has not considered. This drawback has been overcome in the proposed algorithm. According to the MCT algorithm, tasks are assigned to VMs such that the incoming task is scheduled to the VM with the highest speed. Therefore, the MCT algorithm focuses only on the speed of VMs [9].

3. The Proposed Task Scheduling Algorithm

According to the work in this paper, a new dynamic task-scheduling algorithm on the Cloud Computing environment has been proposed. The main idea of the proposed algorithm is how to allocate new coming tasks on the existed resources while some existed tasks are running. The proposed algorithm is based on our static task scheduling algorithm [10]. The principles of the proposed algorithm can be summarized in the following steps:-

1- Scheduling the existed tasks (current tasks) by calculating the total processing power of the available resources (*i.e.*, VMs), and the total requested processing power by the users' tasks, calculating the power factor of each VM (the ratio of its processing power to the total processing power of all VMs), then searching the users' tasks to find a task or a group of tasks that their processing power near to the power factor of each VM [10].
2- When a task is coming during the runtime (dynamic tasks) of the existed tasks, it is scheduled using the following steps:-

a) Calculate the power factor (PF) of each VMi which is equal to the ratio between the MIPS of the VMi and the MIPS of the worst VM (*i.e.*, the VM which contains only single CPU (core)) by using the following equation:

$$\boldsymbol{PF\_VMi} = \frac{\boldsymbol{MIPSi}}{\boldsymbol{MIPS\_s}} \quad (1)$$

Where, MIPS_s is the processing power of the worst VM.

b) Calculate the availability level (AVL) of all existed resources (*i.e.*, VMs) by using the following equation:

$$\boldsymbol{AVL\_VMi} = \frac{\boldsymbol{\#CPUs\_VMi}}{\boldsymbol{VMi\_cost}} * \boldsymbol{PF\_VMi} - \boldsymbol{\#\ running\ tasks\ on\ VMi} \quad (2)$$

Where **#CPUs_VMi** is the number of CPUs (cores) in the VM. For example, the number of cores for each VM and its price according to Amazon EC2 and Google price models are represented in Table [1] and Table [2] respectively [11][12].

By increasing **#CPUs_VMi** and the power ratio of the VM, the availability level of the VM is increased, but when the cost of the VMi and the number of running tasks on the VM increase, the availability level of the VM is decreased.

Table 1. Amazon EC2 Instance Types and Price [11]

Instance Type	VM_type1	VM_type2	VM_type3
CPU	4	7	20
Price($/hour)	0.34	0.5	.64

Table 2. Google Cloud Instance Types and Price [12]

Machine type	VCPUs	Price($/HOUR)
n1-highmem-2	2	$0.126
n1-highmem-4	4	$0.252
n1-highmem-8	8	$0.504
n1-highmem-16	16	$1.008
n1-highmem-32	32	$2.016

c) Sort the resources (VMs) in descending order according to their AVL.

d) Test AVL level of the VMs if it is acceptable according to QoS requirements (*e.g.*, execution time and cost).

e) If the suitable VM is existed, allocate the task to it. Otherwise, create a new VM, and allocate the task on it.

f) Calculate the execution time of the task on each VM by using the following equation:

$$\boldsymbol{Task(j)\_ET} = \frac{\boldsymbol{task(j)\_MI}}{\boldsymbol{VM(i)\_MIPS}} \quad (3)$$

g) Calculate the cost of each task by using the following equation[13]:

$$\text{Cloudlet (j)\_cost} = \text{VMi\_price} * \boldsymbol{Task(j)\_ET} \quad (4)$$

The pseudo code of the proposed algorithm is as follows:

Input:
Number n of current cloudlets (*i.e.*, tasks).
Number of dynamic cloudlets coming at random time during the execution of the existed cloudlets
Number of VMs (*i.e.*, resources).

Output:
Mapping Scheme for the requested existed tasks (cloudlets) on the available resources (VMs) and allocate the dynamic cloudlet to the best VM by considering the load on all VMs.

Firstly schedule the existed tasks[10]
Schedule the dynamic tasks coming during the runtime of the existed tasks.
Calculate the power factor of all VMs with respect to the worst one.

for i=1 to m do

calculate the power factor of each VM

$$\boldsymbol{PF\_VMi} = \frac{MIPSi}{MIPS\_s}$$

Calculate the availability level (AVL)of all existed resources (VMs)

$$\text{AVL\_VMi} = \frac{\#\text{CPUs\_VMi}}{\text{the cost of the VMi}} * \text{PF\_}VMi - \#\, running\ tasks\ on\ VMi$$

Sorting the resources (VMs) in descending order according to their AVL.

if AVL is satisfied Then allocate the task

Else create new VM satisfy it.

Calculate the time and cost for the task execution.

4. Performance Evaluation

The proposed task scheduling algorithm is slightly different from the default dynamic task scheduling algorithms, which mainly based on queuing the incoming tasks for a slice of time, then scheduling them as if they were static task scheduling, and repeat this sequence in periodic manner. Generally, our proposed algorithm is considered semi-dynamic schedule, where the incoming tasks will be scheduled to execute during the runtime of the static tasks. So, there is no waiting list (*i.e.*, queue) for the incoming task, but live scheduling is done to define the proper available resource for receiving new task.

According to the proposed algorithm, our static task scheduling algorithm which has been existed is applied first [10]. If a new task is coming during the runtime of the scheduled tasks, it will be scheduled.

To evaluate the performance of the proposed algorithm, a comparative study has been done between the proposed algorithm, Round Ribbon (RR) algorithm, and Minimum Completion Time (MCT) algorithm by considering two performance parameters; total execution time, and the cost of scheduling.

4.1. Experimental Environment

The proposed task scheduling algorithm has been written by java programming language using eclipse program in Intel(R) Core(TM) 2 Duo CPU in 2.10 GHZ of processor and 4.00 GB of RAM, through the CloudSim simulator. On the other hands, CloudSim is an extensible simulation toolkit that enables modeling and simulation of Cloud Computing systems and application provisioning environments. The CloudSim toolkit supports the Clouds components such as data centers, virtual machines (VMs), and resource provisioning policies. On the other hand, the Cloudsim implements generic application provisioning techniques that can be extended easily with limited efforts [14].

According to the implementation using Cloudsim, the VMs are considered the cloud resources and Cloudlets as tasks/jobs. The make-span of different algorithms has been measured by considering varying the cloudlets while VMs are fixed, as well as, varying VMs. Cloudlets were generated from a standard formatted workload of a High performance computing center called HPC2N in Sweden as a benchmark. According to this benchmark, Each row in the workload represents a cloudlet where the first column is the id of the cloudlet, the fourth column is the length of the cloudlet (*i.e.*, the runtime value multiplied by the rating which is defined as 1 MI in CloudSim), and the eighth column represents the number of the requested processing elements [15].

The calculation of the cost has been done using price models of Amazon EC2, and Google Clouds (see Table[1] and Table[2]) [11], [12].

4.2. Experimental Results

Two parameters have been used to measure the performance of the proposed semi-dynamic task scheduling algorithms; execution time of the tasks, and the cost of execution. Execution time is the summation of execution time of all tasks, and the total cost is the summation of the cost of all tasks on the available resources.

The proposed algorithm has been implemented into two versions; Proposed_1 and Proposed_2. According to Proposed_1 version, creating new VM is not considered, while Proposed_2 version considers initiating new VM.

4.2.1. The Evaluation Results Using Proposed_1

Execution Time Results

The execution time of any cloudlet is calculated using equation (3). First, the execution time will be evaluated using Amazon EC2 price model [11], and then it is evaluated using Google price model [12].

Using Amazon EC2 Price Model:

The experimental results of our proposed algorithm, RR, and MCT algorithms are presented in Figures 2, and 3, using different VMs, as well as, different cloudlets (jobs/tasks). According to the experimental results in Figure 2 with considering 5 VMs, it is found that the performance of our proposed algorithm outperforms RR, and MCT algorithms with respect to the total execution time (*i.e.*, makespan) by 42.4 %, and 75.5 % respectively. The overall performance of the proposed algorithm (*i.e.*, average improvement) is 59.1 % relative to the RR, and MCT algorithms.

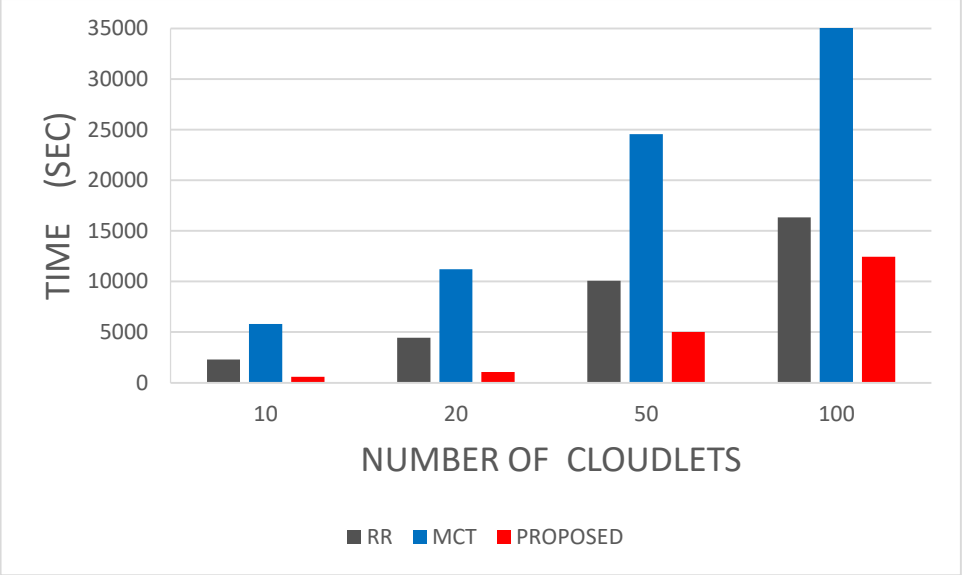

Figure 2. Total Execution Time by Considering 5 VMs (EC2)

According to the experimental results in Figure 3 with considering 10 VMs, it is found that the performance of our proposed algorithm outperforms RR, and MCT algorithms with respect to the total execution time (*i.e.*, makespan) by by 56.4 %, and 38.6 % respectively. The overall performance of the proposed algorithm (*i.e.*, average improvement) is 47.5 % relative to the RR, and MCT algorithms.

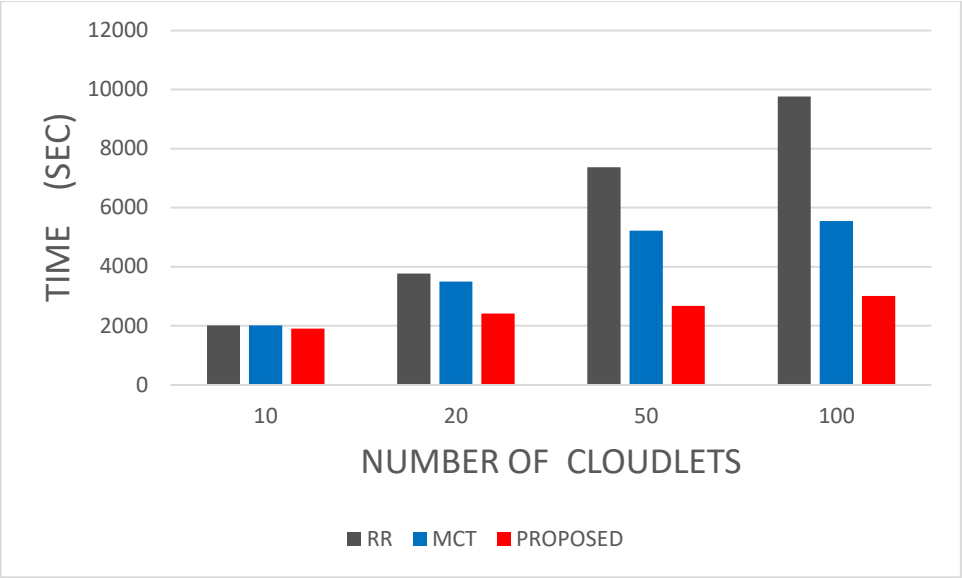

Figure 3. Total Execution Time by Considering 10 VMs (EC2)

Using Google Price Model:

The experimental results of our proposed algorithm, the RR algorithm, and MCT algorithm are presented in Figures 4, and 5, using different VMs, as well as, different cloudlets (jobs/tasks). According to the experimental results in Figure 4 with considering 5 VMs, it is found that the performance of our proposed algorithm outperforms RR, and MCT algorithms with respect to the total execution time (*i.e.*, makespan) by 82.7 %, and 84 % respectively. The overall performance of the proposed algorithm (*i.e.*, average improvement) is 83.3 % relative to the RR, and MCT algorithms.

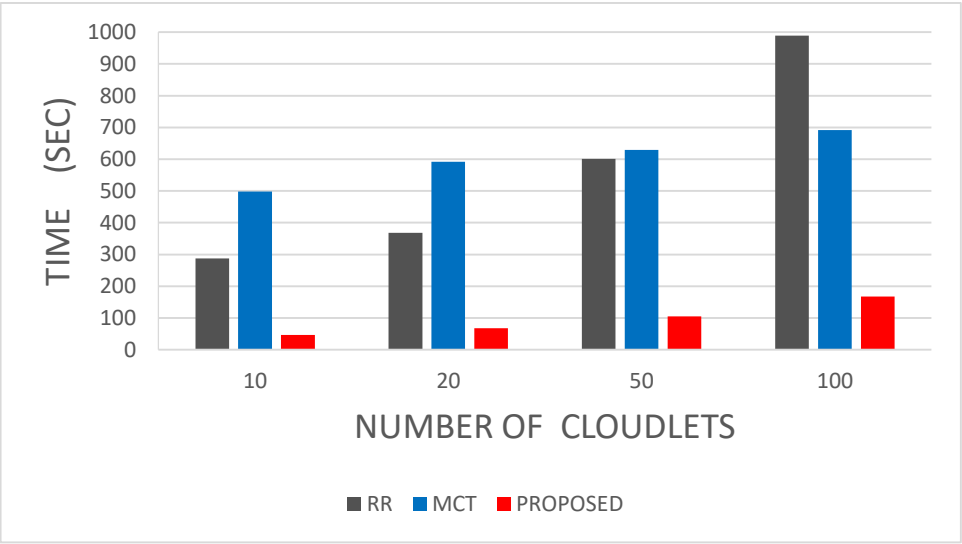

Figure 4. Total Execution Time by Considering 5 VMs (Google)

According to the experimental results in Figure 5 with considering 10 VMs, it is found that the performance of our proposed algorithm outperforms RR, and MCT algorithms with respect to the total execution time (*i.e.*, makespan) by 81.4 %, and 41.2 % respectively. The overall performance of the proposed algorithm (*i.e.*, average improvement) is 61.3 % relative to the RR, and MCT algorithms.

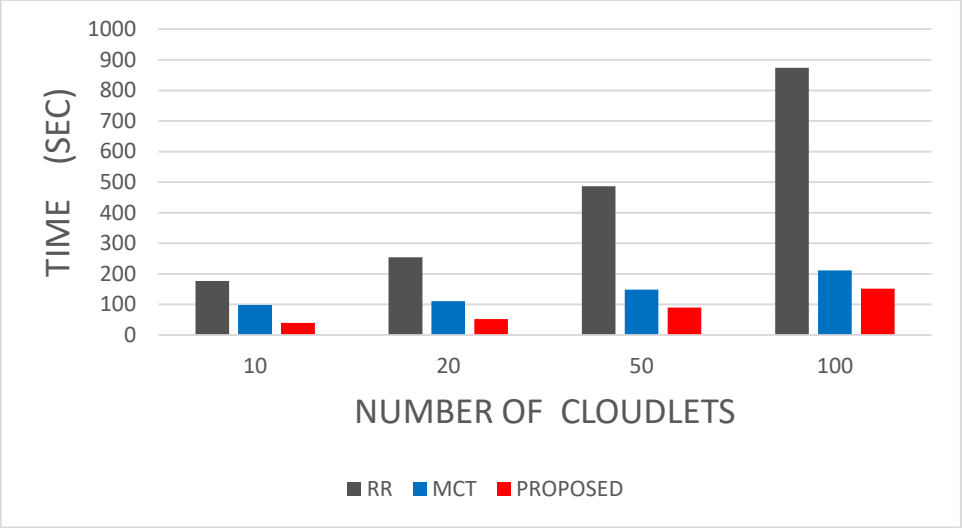

Figure 5. Total Execution Time by Considering 10 VMs (Google)

Generally, by calculating the average make_span of the proposed algorithm, RR, and MCT algorithms, it is found that the proposed algorithm outperforms RR, and MCT algorithms by

65.6 %, and 59.9 % respectively. And the overall make_span of the proposed algorithm is 62.7 % relative to RR, and MCT algorithms.

EXECUTION COST

The cost of executing a cloudlet on a VM is calculated using equation (4). First, the price will be evaluated using Amazon EC2 price model [11], and then it is evaluated using Google price model [12]. The used unit is cent (¢)= .01 dollar($).

Using EC2 Price Model:

According to the used VM_type to run a cloudlet, and the running time of this cloudlet, the cost of this cloudlet can be calculated [10].

According to the experimental results in Figure 6 with considering 5 VMs, it is found that the performance of the proposed algorithm outperforms RR, and MCT algorithms with respect to the total cost (¢) by 28.65 %, and 77.15 % respectively. The overall performance of the proposed algorithm (*i.e.*, average improvement) is 52.9 % relative to the RR, and MCT algorithms.

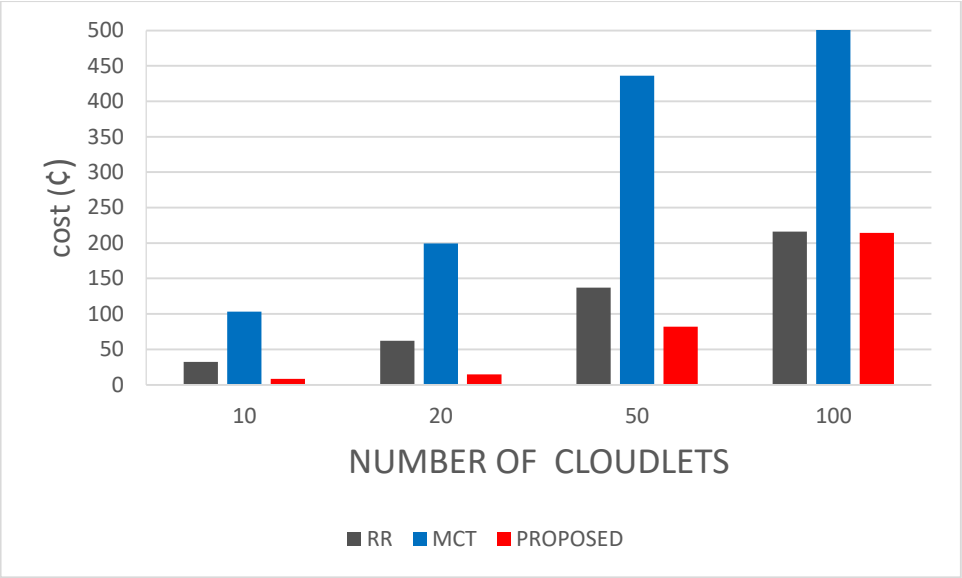

Figure 6. The Cost by Considering 5 VMs (EC2)

According to the experimental results in Figure 7 with considering 10 VMs, it is found that the performance of the proposed algorithm outperforms RR, and MCT algorithms with respect to the total cost(¢) by 60.7 %, and 64.4 % respectively. The overall performance of the proposed algorithm (*i.e.*, average improvement) is 62.55 % relative to the RR, and MCT algorithms.

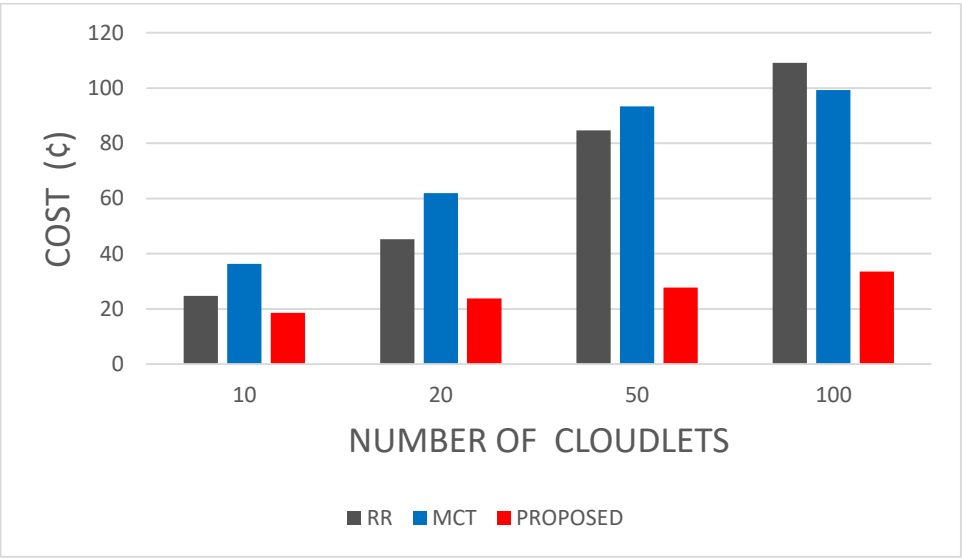

Figure 7. The Cost by Considering 10 VMs (EC2)

Using Google Price Model:

According to the experimental results in Figure 8 with considering 5 VMs, it is found that the performance of the proposed algorithm outperforms RR, and MCT algorithms with respect to the total cost(¢) by 48.6 %, and 86.87 % respectively. The overall performance of the proposed algorithm (*i.e.*, average improvement) is 67.74 % relative to the RR, and MCT algorithms.

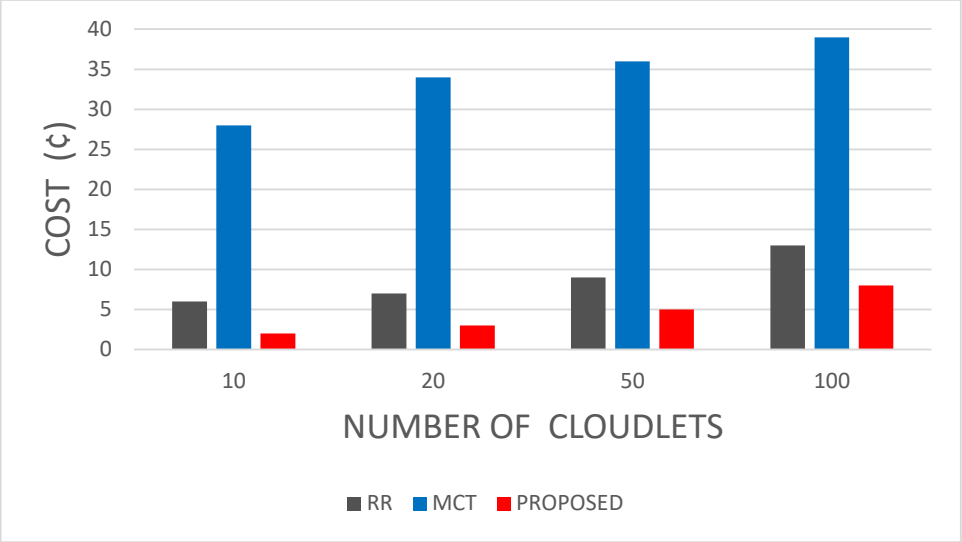

Figure 8. The Cost by Considering 5 VMs (Google)

According to the experimental results in Figure 9 with considering 10 VMs, it is found that the performance of the proposed algorithm outperforms RR, and MCT algorithms with respect to the total cost (¢) by 0.2 %, and 47.14 % respectively. The overall performance of the proposed algorithm (*i.e.*, average improvement) is 23.67 % relative to the RR, and MCT algorithms.

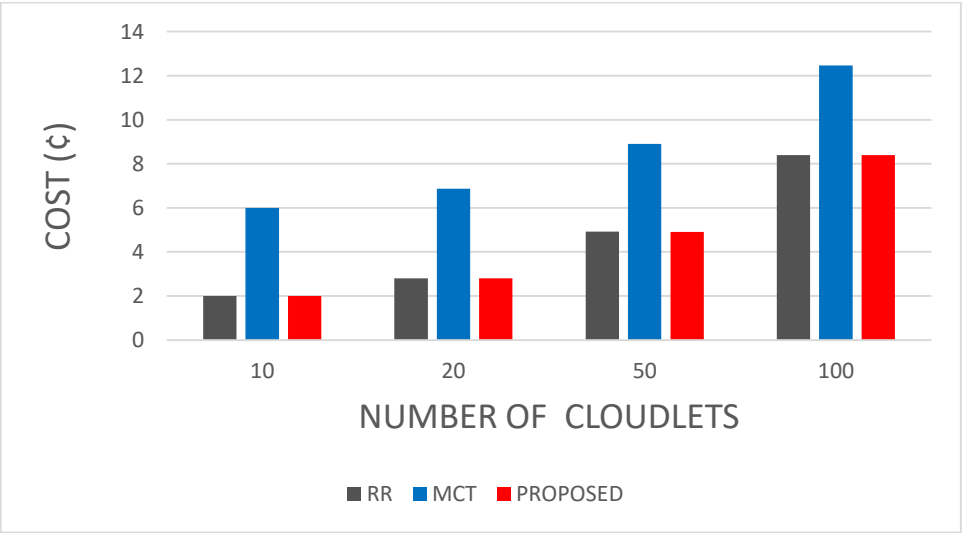

Figure 9. The Cost by Considering 10 VMs (Google)

Generally, by calculating the average cost of the proposed algorithm, RR, and MCT algorithms, it is found that the proposed algorithm outperforms RR, and MCT by 34.5 %, and 68.89 % respectively. And the overall cost of the proposed algorithm is minimize by 51.7 % relative to RR, and MCT algorithms.

4.1.1 The Evaluation Results Using Proposed_2

In this version, the ability of creating a new VM during the runtime will be considered.

The principle of Prposed_2 version is that if the availability level of the highest available VM is under the required QoS of the incoming task (*i.e.*, time and cost), a new VM will be created during the run time. This copes with the dynamic feature of the Cloud.

In the following section, the different between the two versions of the proposed algorithm (*i.e.*, creating or not creating new VM) and how it will affect the performance of the proposed algorithm will be investigated.

Execution Time

Figure 10 represents the implementation results Propose_1 (*i.e.*, no creation new VM) and Propose_2 (*i.e.*, considering creation new VM) with considering 5 VMs. According to the results, it is found that the performance of Proposed_2 algorithm outperforms Proposed_1 algorithm with respect to the total execution time by 92.26 %.

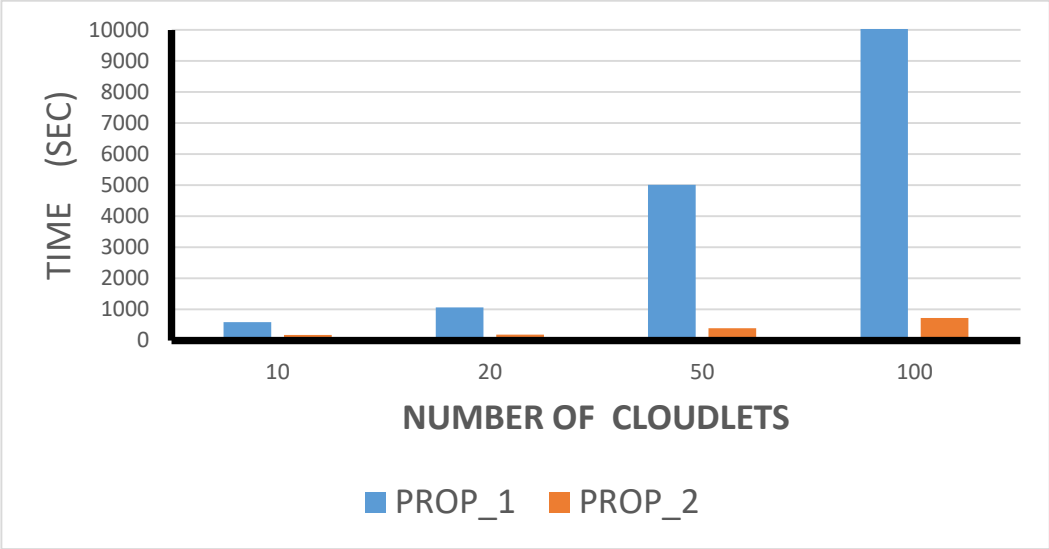

Figure 10. The Total Execution Time by Considering 5 VMs (EC2)

According to the experimental results in Figure 11 with considering 10 VMs, it is found that the performance of Proposed_2 version outperforms Proposed_1 version with respect to the total execution time by 81.77 %.

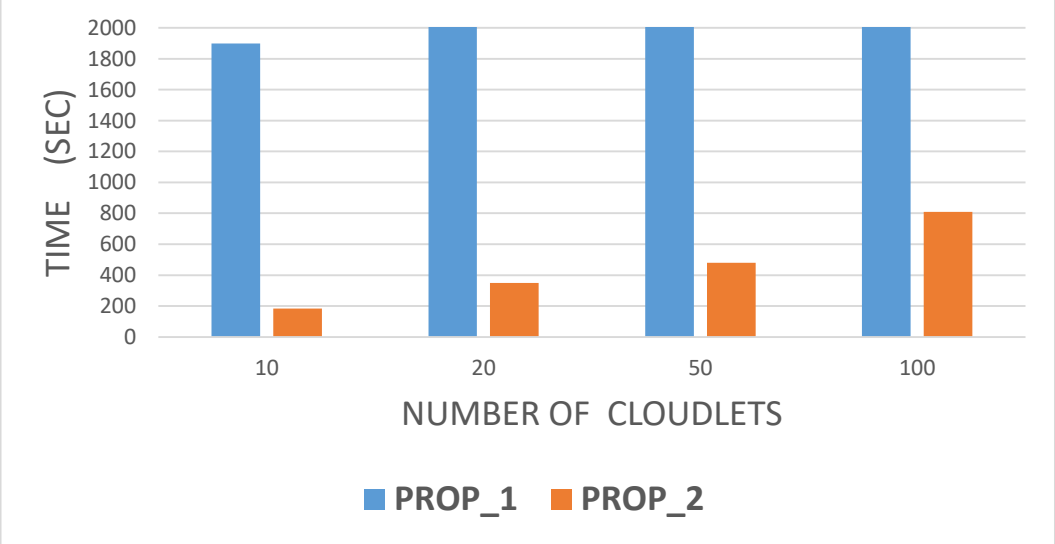

Figure 11. The Total Execution Time by Considering 10 VMs (EC2)

Using Google Price Model:

According to the results shown in Figure 12 with considering 5 VMs, it is found that the performance of Proposed_2 algorithm outperforms Proposed_1 algorithm with respect to the total execution time by 34.79 %.

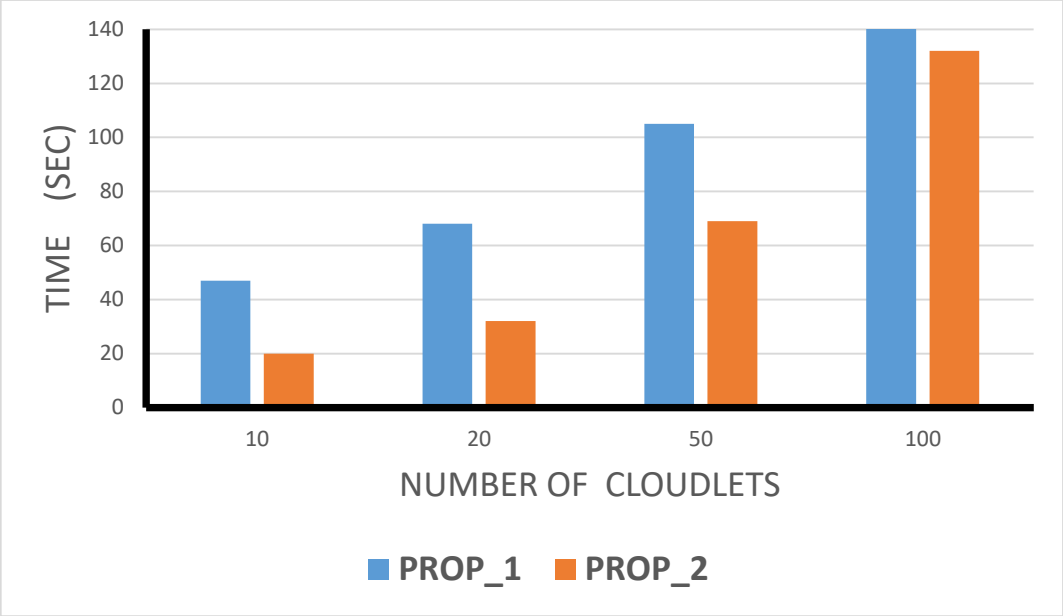

Figure 12. The Total Execution Time by Considering 5 VMs (Google)

According to the results shown in Figure 13 with considering 10 VMs, it is found that the performance of Proposed_2 algorithm outperforms Proposed_1 algorithm with respect to the total execution time by 20.74 %.

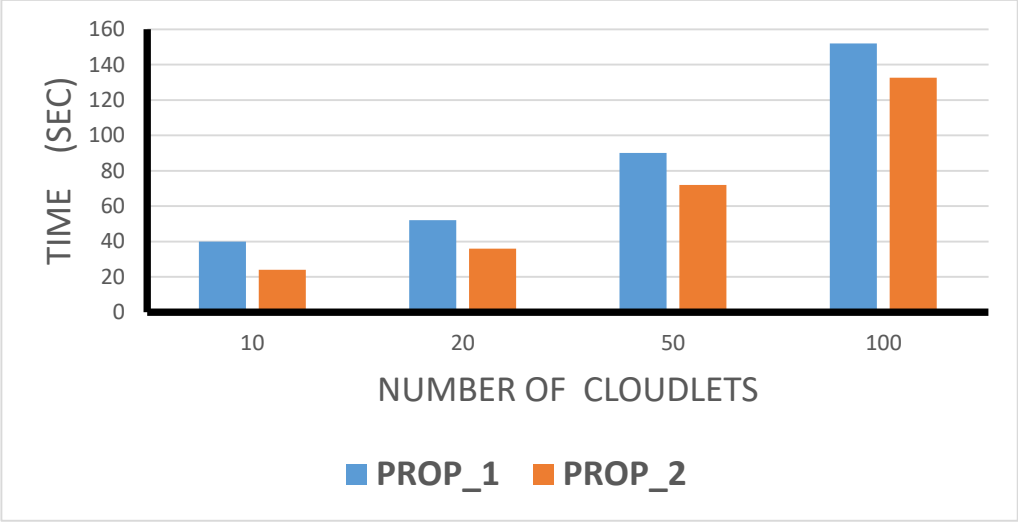

Figure 13. The Total Execution Time by Considering 10 VMs (Google)

Generally, by calculating the average of the total execution time of Proposed_2 version, and proposed_1 version of our proposed algorithm, it is found that Proposed_2 version outperforms Proposed_1 version by 57.39 %.

Execution Cost using Amazon EC2

According to the experimental results in Figure 15 with considering 5 VMs, it is found that the performance of Proposed_2 version outperforms Proposed_1 version with respect to the total cost by 98.99 %.

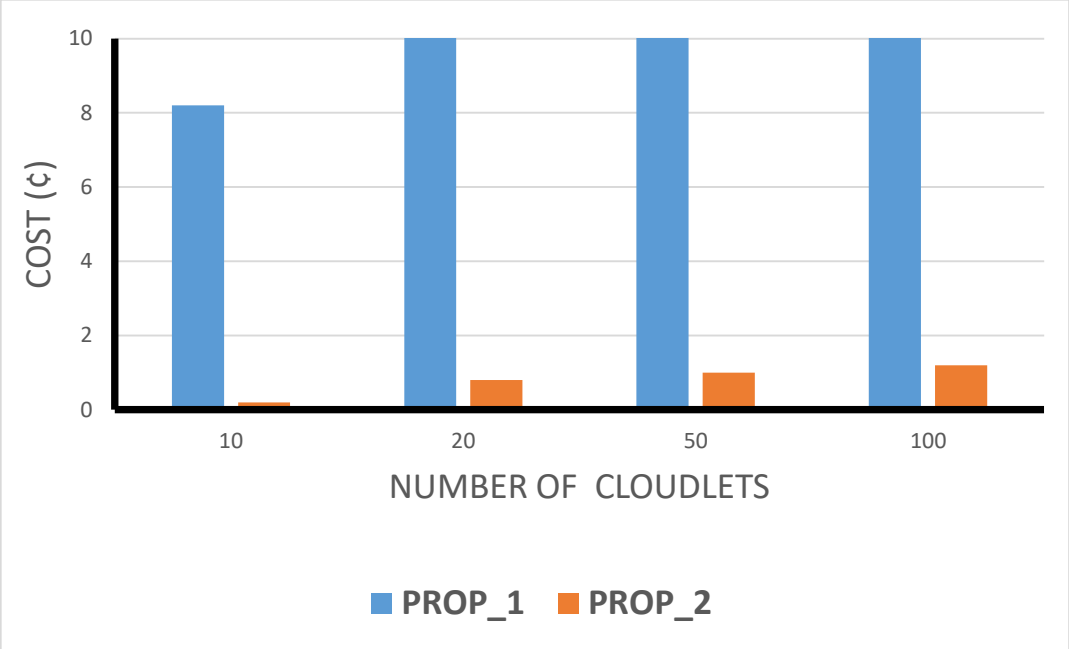

Figure 15. The cost by Considering 5 VMs (EC2)

According to the experimental results in Figure 16 with considering 10 VMs, it is found that the performance of Proposed_2 version outperforms proposed_1 version with respect to the total cost by 93.2 %.

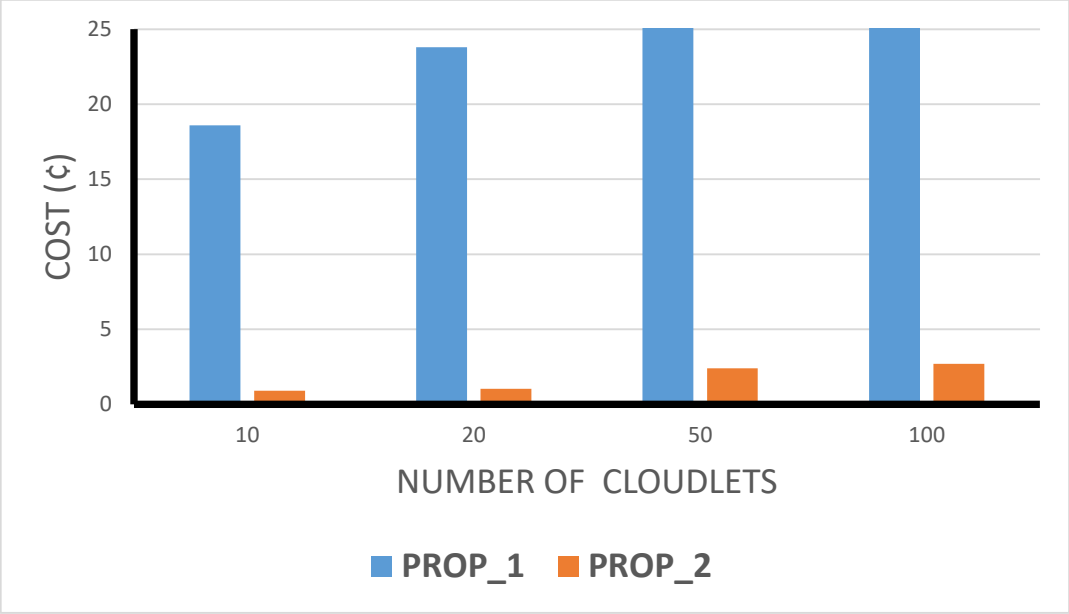

Figure 16. The Cost by Considering 10 VMs (EC2)

Using Google

According to the experimental results in Figure 17 with considering 5 VMs, it is found that the performance of Proposed_2 version outperforms proposed_1 version with respect to the total cost by 59.6 %.

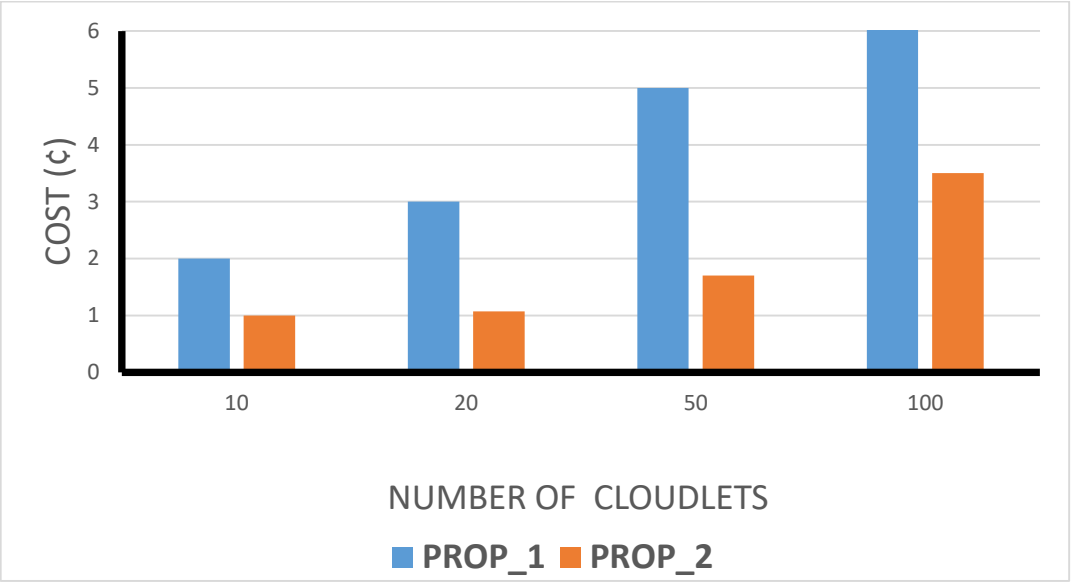

Figure 17. The Cost by Considering 5VM (Google)

According to the experimental results in Figure 18 with considering 10 VMs, it is found that the performance of Proposed_2 version outperforms proposed_1 version with respect to the total cost by 36.74 %.

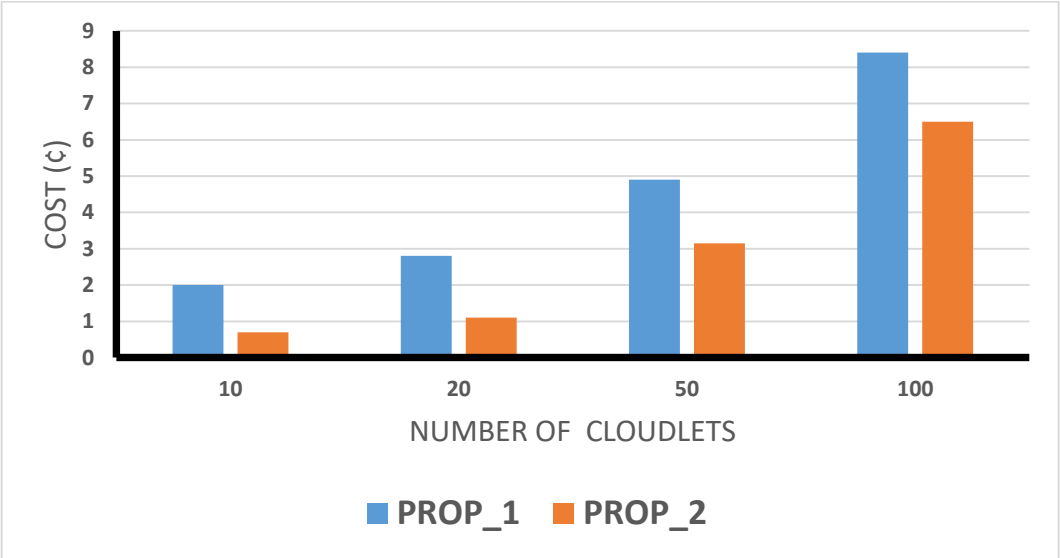

Figure 18. The Cost by Considering 10 VM (Google)

Generally, by calculating the average of the cost of Proposed_2 version, and the Proposed_1 version of our proposed algorithm, it is found that Proposed_2 version outperforms Proposed_1 version by 72.13 %.

5. Conclusion and Future Work

Task scheduling is one of the main issues in the Cloud Computing. Dynamic task scheduling is considered an important issue for saving the time and decreasing the cost. In this paper, a new semi-dynamic task scheduling algorithm has been proposed for the Cloud Computing environment. The main idea of the proposed algorithm is that continuous monitoring the availability level of the existed resources (VMs) according to its processing power, cost, the number of running tasks on it, and allocating the incoming task to the most available one. To evaluate the performance of the proposed algorithm, a comparative study has been done among the proposed algorithm, RR, and MCT algorithms with respect to the total execution time (*i.e.*, make-span), and the cost of execution using Amazon EC2, and Google pricing models. The proposed algorithm has been implemented into two versions; Propose_1 and Propose_2. According to Propose_1 version, creating new VM is not considered, but creating new VM is considered in Propose_2 version. The experimental results of Propose_1 version of our proposed algorithm prove the efficiency of the proposed algorithm with respect to the RR and MCT algorithms by minimizing make-span by 62.7 %. In addition, the cost is decreased by 51.7 %. According to Propose_2 version of our proposed algorithm, if the availability level of the highest available VM is under the required QoS of the incoming task (*i.e.*, time and cost), a new VM will be created during the run time. By comparing the Proposed_2 and Proposed_1 versions, the experimental results show that Proposed_2 version outperforms Proposed_1 with respect to total execution time and total cost by 57.39 %, and by 72.13 % respectively. The new proposed algorithm could be further extended by considering dependent tasks.

References

[1] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin and I. Stoica, "Above the clouds: A Berkeley view of cloud computing", Dept. Electr. Eng. Comput. Sci. Univ. California, Berkeley, Rep. UCB/EECS, vol. 28, no. 13, pp. **(2009)**.
[2] B. Xu, C. Zhao, E. Hu and B. Hu, "Job scheduling algorithm based on Berger model in cloud environment", Adv. Eng. Softw., vol. 42, no. 7, **(2011)**, pp. 419–425.
[3] R. K. Sharma and N. Sharma, "A Dynamic optimization algorithm for task scheduling in cloud computing with resource utilization", Int. J. Sci. Eng. Technol., no. 2, **(2013)**, pp. 1062-1068.
[4] R. Kannan, R. U. Rasool, H. Jin and S. R. Balasundaram, "Managing and Processing Big Data in Cloud Computing", vol. i, 1AD.
[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to algorithms second edition." The MIT Press, **(2001)**.
[6] S. Singh and K. Kant, "Greedy grid scheduling algorithm in dynamic job submission environment", Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference, **(2011)**, pp. 933-936.
[7] L. Ma, Y. Lu, F. Zhang and S. Sun, "Dynamic task scheduling in cloud computing based on greedy strategy", International Conference on Trustworthy Computing and Services, **(2012)**, pp. 156-162.
[8] J. Maniyar Bhumi and others, "Review On Round Robin Algorithm For Task Scheduling In Cloud Computing", Journal of Emerging Technologies and Innovative Research, vol. 2, no. 3, **(2015)** March.
[9] F. Dong and S. G. Akl, "Scheduling Algorithms for Grid Computing : State of the Art and Open Problems", Components, vol. 202, no. 4, **(2006)**, pp. 1-55.
[10] E. Ibrahim, N. A. El-Bahnasawy and F. A. Omara, "Job Scheduling based on Harmonization between the Requested and Available Processing Power in The Cloud Computing Environment", International Journal of Comput. Appl., vol. 125, no. 13, **(2015)**.
[11] "Amazon EC2", [Online]. Available: http://aws.amazon.com/ec2/. [Accessed: 01-Jan-2014].
[12] "Google,". [Online]. Available: https://cloud.google.com/compute/pricing. [Accessed: 11-Jan-2016].
[13] R. Sahal and F. A. Omara, "Effective virtual machine configuration for cloud environment", 2014 9th Int. Conf. Informatics Syst. INFOS 2014, pp. PDC15–PDC20, **(2015)**.
[14] R. Buyya, R. Ranjan and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities", High Performance Computing & Simulation, 2009. HPCS'09. International Conference, **(2009)**, pp. 1-11.
[15] "The HPC2N Seth log," [accessed 2015]. [Online]. Available: Available: http://goo.gl/wrxAK. [accessed 2015].

