Future Computing and Informatics Journal

Volume 3
Issue 2 *Vol 3,Issue 2,2018*

Article 26

2018

# A new online scheduling approach for enhancing QOS in cloud

Aida A. Nasr
*Faculty of Electronic Engineering, Menofia University, Department of Computer Science and Engineering, Menouf 32952, Egypt*, aida.nasr2009@gmail.com

Nirmeen A. El-Bahnasawy
*Faculty of Electronic Engineering, Menofia University, Department of Computer Science and Engineering, Menouf 32952, Egypt*, nermeen.abd@el-eng.menofia.edu.eg

Gamal Attiya
*Faculty of Electronic Engineering, Menofia University, Department of Computer Science and Engineering, Menouf 32952, Egypt*, gamal.attiya@yahoo.com

Ayman El-Sayed
*Faculty of Electronic Engineering, Menofia University, Department of Computer Science and Engineering, Menouf 32952, Egypt*, ayman.elsayed@el-eng.menofia.edu.eg

Available online at www.sciencedirect.com

ScienceDirect

Future Computing and Informatics Journal 3 (2018) 424–435
http://www.journals.elsevier.com/future-computing-and-informatics-journal/

# A new online scheduling approach for enhancing *QOS* in cloud

Aida A. Nasr*, Nirmeen A. El-Bahnasawy, Gamal Attiya, Ayman El-Sayed

*Faculty of Electronic Engineering, Menofia University, Department of Computer Science and Engineering, Menouf 32952, Egypt*

Received 2 February 2018; accepted 8 November 2018
Available online 16 November 2018

**Abstract**

Quality-of-Services (*QoS*) is one of the most important requirements of cloud users. So, cloud providers continuously try to enhance cloud management tools to guarantee the required QoS and provide users the services with high quality. One of the most important management tools which play a vital role in enhancing *QoS* is scheduling. Scheduling is the process of assigning users' tasks into available Virtual Machines (VMs). This paper presents a new task scheduling approach, called Online Potential Finish Time (*OPFT*), to enhance the cloud data-center broker, which is responsible for the scheduling process, and solve the QoS issue. The main idea of the new approach is inspired from the idea of passing vehicles through the highways. Whenever the width of the road increases, the number of passing vehicles increases. We apply this idea to assign different users' tasks into the available VMs. The number of tasks that are allocated to a VM is in proportion to the processing power of this VM. Whenever the VM capacity increases, the number of tasks that are assigned into this VM increases. The proposed *OPFT* approach is evaluated using the *CloudSim* simulator considering real tasks and real cost model. The experimental results indicate that the proposed *OPFT* algorithm is more efficient than the *FCFS*, *RR*, *Min-Min*, and *MCT* algorithms in terms of schedule length, cost, balance degree, response time and resource utilization.


*Keywords:* Cloud computing; Task scheduling; *QOS*; Load balancing

## 1. Introduction

Cloud computing is another worldview of shared resources. It has enormous pools of resources that are accessible to clients for utilizing as pay-per-use on-demand over the Internet [1]. The cloud-computing environment provides three main service models: Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS). In all cases, clients can utilize what they need on the cloud and pay just for what they utilize. Services suppliers, similar to Amazon, Google, and Microsoft, allow their clients to assign, get to, and deal with a gathering of Virtual Machines (*VM*s) that keep running inside the server farms and just charge them for the time of utilizing the machines [2].

These days, cloud computing is turning into a proficient paradigm that gives high-performance computing resources over the Internet to execute large-scale complex applications. However, one of the key issues that debase the cloud computing performance is resource allocation. Resource allocation is characterized by the presence of limited number of virtual machines that should be allocated to execute several tasks. Subsequently, management of cloud resources is important especially when numerous tasks are submitted at the same time to the cloud computing [3].

Numerous researchers have presented distinctive methods to solve resource allocation. The primary objective of some of existing methods is to decrease the makespan. Yet, they overlook critical imperatives like, time complexity, response

* Corresponding author.
*E-mail addresses:* aida.nasr2009@gmail.com (A.A. Nasr), nermeen.abd@el-eng.menofia.edu.eg (N.A. El-Bahnasawy), gamal.attiya@yahoo.com (G. Attiya), ayman.elsayed@el-eng.menofia.edu.eg (A. El-Sayed).

Peer review under responsibility of Faculty of Computers and Information Technology, Future University in Egypt.

https://doi.org/10.1016/j.fcij.2018.11.005

time, resource utilization, and sometimes they don't take into account the cost. This metrics evaluate the Quality of Service *QOS* of cloud system. The QOS is defined as the required benefit desires as consented to be given by the service supplier to users in light of their needs [4]. Users need to execute their applications in low time and with low cost. The problem of allocating resources is a part of *QOS* management issue.

This article presents a new resource allocation algorithm for cloud computing environment to solve QOS management issue. It considers many terms like, makespan, cost, resource utilization, and time complexity. The *OPFT* enhances the overall performance of cloud computing by taking into account both the tasks requirements and resources availability, and improving both total execution cost and resource utilization with low time complexity. It also considers minimizing the response time, where it checks continuously, if new tasks submitted or not to take them in consideration.

The remainder of this article is organized as follows. Section 2 introduces the cloud computing and the task scheduling problem. Section 3 presents the related work while the proposed *OPFT* is developed in section 4. Section 5 discusses the simulation results while section 6 presents the conclusion of this research work.

## 2. Cloud computing and task scheduling problem

There are main components [5] of task scheduling in cloud computing system, client, cloud information system, Datacenter Broker, and *VM*s. Fig. 1 discusses the relationship of these components.

**Client component**: Client is responsible for submitting his/her task(s) to supplier and wait until it is executed. The submitted tasks determine the requirement resources.

**Information system**: All tasks information stores in cloud information system. This component is very important. It provides the necessary information of tasks arrived into cloud computing environment for execution. Information such as task length, arrival time, resources information, number of submitted tasks, and other information are stored in the information system.

**Datacenter broker:** The main component in task scheduling model is datacenter broker. It is the backbone of scheduling process. Datacenter broker include the scheduler (by default FCFS scheduler), which is responsible for scheduling the tasks. It determines the execution order of each task.

**Virtual machines:** *VM*s is the component that executes client tasks and returns the results. It is the critical component, because the numbers of them are less than the number of submitted tasks. *VM*s component called also the available resources.

In this article, we focus on the IaaS model. Our model looks like Amazon EC2 model. IaaS provides virtual computing, storage, and network resources without returning to the physical resources complexity. Cloud computing technology uses the virtualization technology to provide all resources in virtual form [6]. The users will rent high-performance computing resources in the form of *VMs* to execute their applications. Each *VM* has different configuration such as CPU

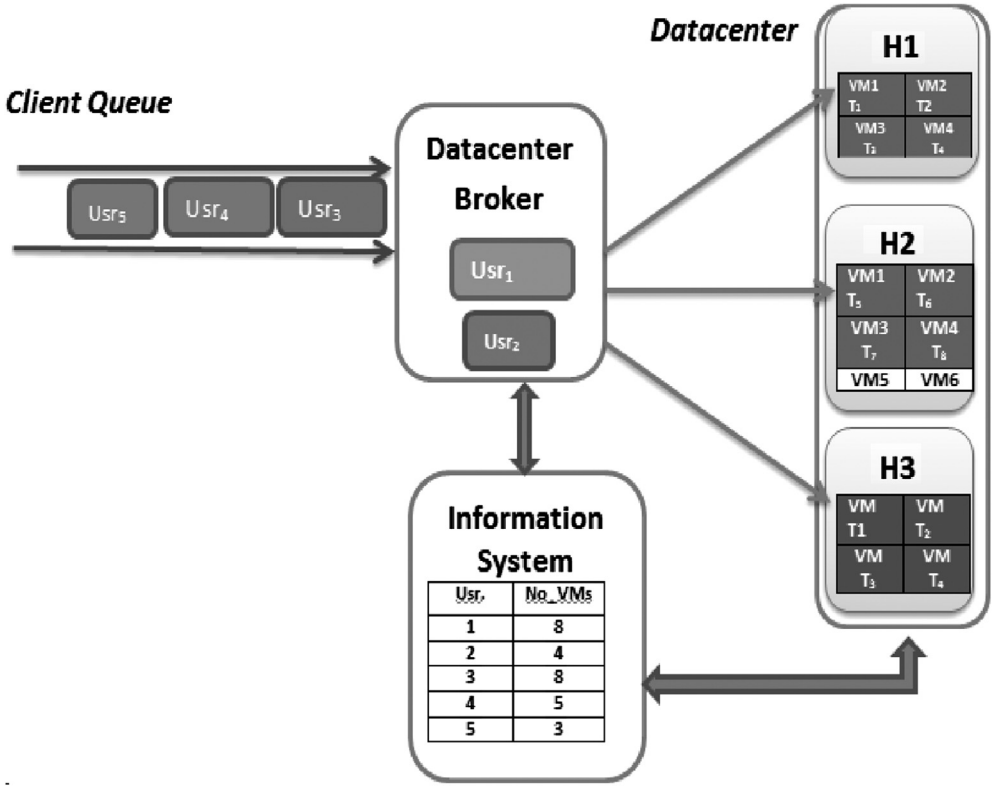


Fig. 1. Components of scheduling in cloud computing.

speed, number of CPU cores, memory size, cost per time spent, and so on.

Resource allocation algorithm is a method to allocate the available resources to execute the submitted tasks. Each algorithm should consider some of conditions to allocate the tasks in a correct way. These conditions [7] are:

1. Any task should be assigned into exactly one *VM*.
2. Ensure that a task assigned on only one a *VM* which is running.
3. Total processing requirements of all tasks, in Million Instructions (MI), assigned into a virtual resource should not exceed the available processing capacity of that virtual resource.
4. Total memory requirements of all tasks assigned into a virtual resource should not exceed the maximum memory available with that a virtual resource.

## 3. Related work

Two types of algorithms are developed for task scheduling; heuristic algorithms and meta-heuristic algorithms. Heuristic algorithms depend extremely on the predictions to achieve a near optimal solution. These algorithms have low time complexity, but they often provide high schedule length [8]. In opposition to heuristic-based, the meta-heuristic strategies search the solution space in a direct manner and create proficient outcomes on broad domain problems, yet these techniques often have high time complexity [9]. In this article we focus on the heuristic algorithms type. Many heuristic algorithms are developed to be used to achieve the near optimal solution. Here, we will discuss some of them:

***OPFT Algorithm***

***Input***: *Arrived Tasks*
***Output***: *Scheduling Solution*

1. *Prepare information about available resources and tasks requirements*
2. *Sort the arrived tasks in descending order list*
3. *Calculate* $Total_{Req}$ *and* $Total_{Avail}$
4. *Calculate PFT*
5. *Set* $\alpha$=1.00001
6. *For i=0 to n-1*
7. *Select task* $T_i$ *from ordered List*
8. ***For*** *j=0 to m-1*
9. ***If***$((TFT(VM_j) + FT(T_i, VM_j) <= PFT)$
10. *Assign selected* $T_i$ *to* $VM_j$
11. $TFT(VM_j) += FT(T_i, VM_j)$
12. *Break*
13. ***Else if*** $(TFT(VMj)+FT(Ti, VMj))<= \mu*PFT)$
14. *Assign selected task to* $VM_j$
15. $TFT(VM_j) += FT(T_i, VM_j)$
16. *Break*
17. ***Else***
18. $\mu=\mu+0.00001$
19. *Continue*
20. ***End if***
21. ***End for***
22. 
23. ***If there are new submitted tasks go to step(1)***

Fig. 2. Online Potential Finish Time scheduling algorithm (*OPFT*).

Table 1
Tasks length with millions of instructions (MI).

| Task | Length |
|---|---|
| $T_0$ | 1000 |
| $T_1$ | 1500 |
| $T_2$ | 2000 |
| $T_3$ | 2500 |
| $T_4$ | 3000 |
| $T_5$ | 3500 |
| $T_6$ | 4000 |
| $T_7$ | 4500 |
| $T_8$ | 4700 |
| $T_9$ | 5000 |
| **$Total_{req}$** | **31,700** |

**First Come First Serve (FCFT)** [10]: *FCFS* is the first method that is used in cloud computing systems [11]. In *FCFS* method, all tasks are combined in queue and wait until the resources are available. Once they become available, the tasks are assigned to them based on arrival time. The *FCFS* is simple method to implement in cloud computing, but it does not take into account any criteria for scheduling the tasks to *VMs* [11]. Therefore, the total time resulting from this way is very high and the balancing degree is very small.

**Round-Robin (RR)** [12]: It uses the same steps of *FCFS* for scheduling some tasks, but RR scheduler mostly utilizes time-sharing, presenting each task a time slot or quantum, and stopping the task if it is not completed. RR is very useful for load balancing, but it gives a high makespan.

**Min-Min algorithm** [13]: it is a heuristic method used for task scheduling [12–14]. Min-min algorithm computes minimum completion time of each task overall *VM*s. Then, it finds the task with minimum completion time and assigns it to *VM* that gives this completion time. The algorithm iterates until all tasks are scheduled. Min-min algorithm without improving has a high makespan and it doesn't consider the system load balancing, because it assigns smaller tasks in faster *VM*s.

**Minimum Completion Time MCT** [11]**:** algorithm assigns each process in arbitrary order to the *VM* that has the minimum completion time.

Table 2
Illustration of scheduling solution.

| VM No. | Tasks | $TFT(VM_j)$ by Sec. |
|---|---|---|
| $VM_0$ | $T_9$, $T_8$, $T_7$, $T_1$ | (5000 + 4700+4500 + 2000)/550 = **29** |
| $VM_1$ | $T_6$, $T_5$, $T_0$ | (4000 + 3500+1000)/250 = **34** |
| $VM_2$ | $T_4$, $T_3$, $T_2$ | (3000 + 2500+1500)/200 = **35** |

Table 3
Amazon EC2 pricing model.

| Type | vCPU | ECU | Memory (Giga) | Cost ($) |
|---|---|---|---|---|
| t2.small | 1 | Variable | 2 | $0.023 per Hour |
| t2.large | 2 | Variable | 8 | $0.0928 per Hour |
| m4.large | 2 | 6.5 | 8 | $0.1 per Hour |
| c4.large | 2 | 8 | 3.75 | $0.1 per Hour |
| r3.large | 2 | 6.5 | 15 | $0.166 per Hour |

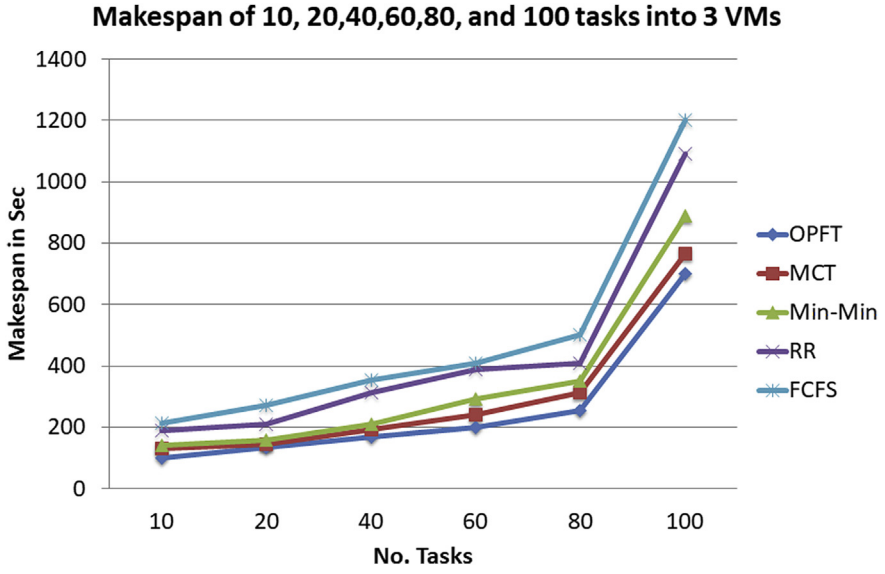


Fig. 3. Schedule length results.

## 4. Online Potential Finish Time (*OPFT*) algorithm

This section presents a new *OPFT* task scheduling algorithm to solve the task-scheduling problem in cloud computing. The main goal is to enhance the overall system performance through improving some parameters like, total execution cost, achieving task requirements without violating system resources and *VM* utilization with low time complexity.

### *4.1. Proposed OPFT*

The main idea of the proposed algorithm is taken from the idea of the vehicles traffic in highways. Whenever the width of the road increases, the number of vehicles, in the traffic, increases. We use this idea to assign tasks into the available VMs. The highest number of tasks should be scheduled into the VM that has highest capacity. We can do this by calculating the Potential Finish Time (PFT) (see Eq. 3 below) of the arrived tasks and schedule the tasks according to this value. That is, give each VM the best fit group of tasks to achieve approximately finish time equal to PFT, although each VM executes a different number of tasks from the other VMs.

The OPFT consists of two stages: **a preparing stage** and **a selection stage**.

1. **Preparing stage:** Firstly, the datacenter broker prepares all information about resource availability and task requirements. Secondly, the algorithm sorts the arrived tasks into descending order. Thirdly, the algorithm calculates the

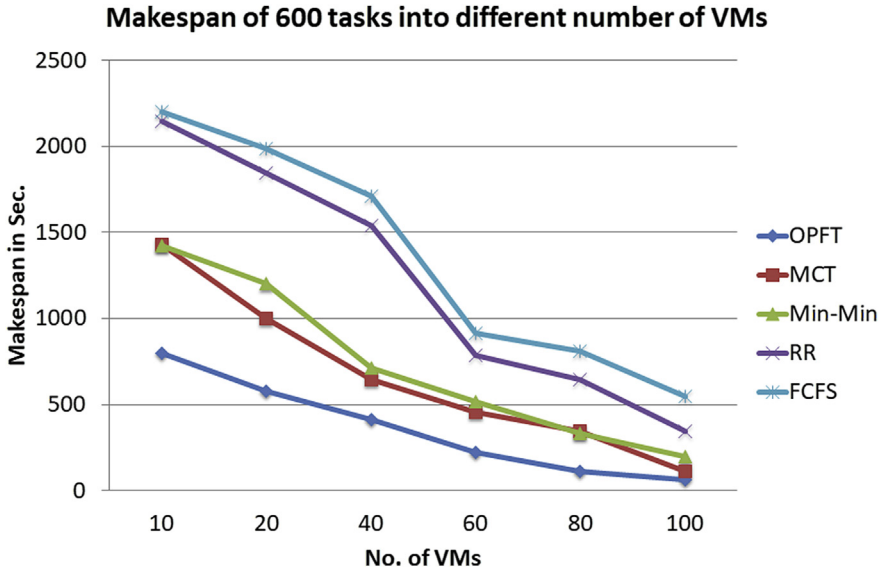


Fig. 4. Schedule length results of 600 tasks.

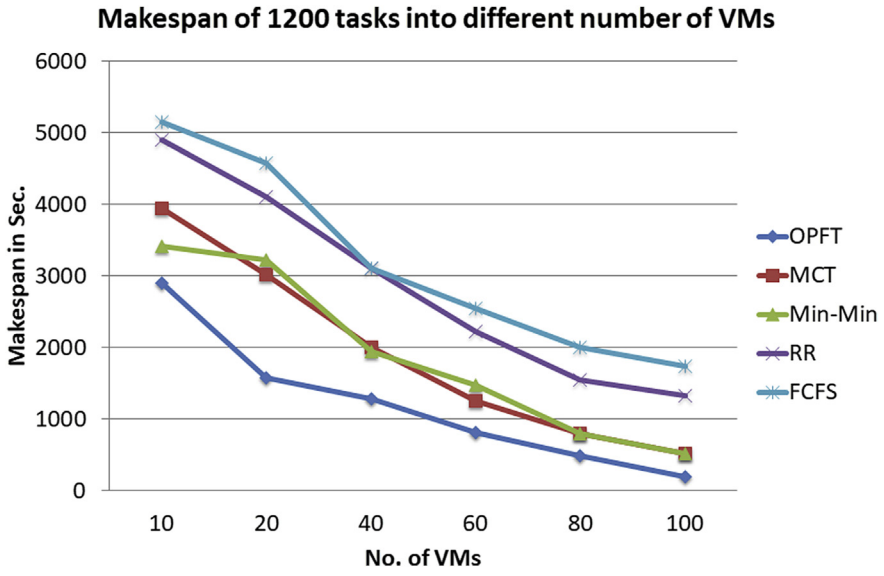


Fig. 5. Schedule length results of 1200 tasks.

Potential Finish Time *PFT* of the scheduling model. It can compute the *PFT* using Equations (1)–(3). It calculates $Total_{Req}$ and $Total_{Avail}$ values, where $Total_{Req}$ is the total requirement processing power of the arrived tasks and $Total_{Avail}$ is the total processing of the available *VMs*, and *Mean* is the average of all tasks length.

$$Total_{\mathrm{Req}} = \sum_{j=0}^{m} MIPS_j \tag{1}$$

$$Total_{Avail} \sum_{i=0}^{n} MI_i \tag{2}$$

$$PFT = \left(Total_{req} + Mean\right) / Total_{Avail} \tag{3}$$

2. **Selection stage:** in this stage, The *OPFT* algorithm assigns tasks into *VM* according to the *PFT* value. By another way, each $VM_j$ should take number of tasks where the total finish time $TFT(VM_j)$ of these tasks is approximately equal to the *PFT* value. Where the $TFT\ (VM_j)$ is the summation of finish time *FT* of all tasks that are assigned to $VM_j$. We use *PFT* as a reference for all VMs to detect the time that each VM should be spent in processing.

We cannot assign the exact group of tasks with $Total_{Req}$ that gives the same *PFT* value for each *VM*. So, we use $\alpha$ value as the control parameter to monitor the selected tasks that are

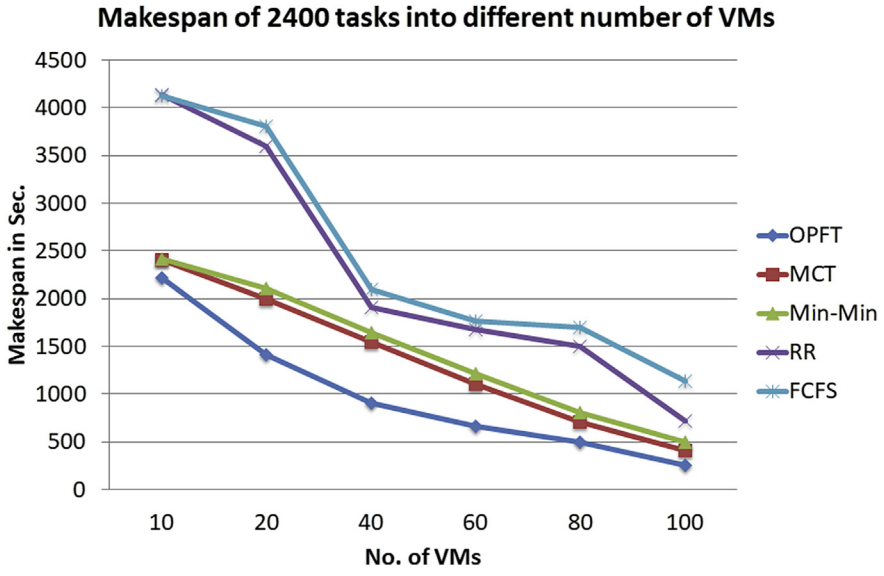


Fig. 6. Schedule length results of 2400 tasks.

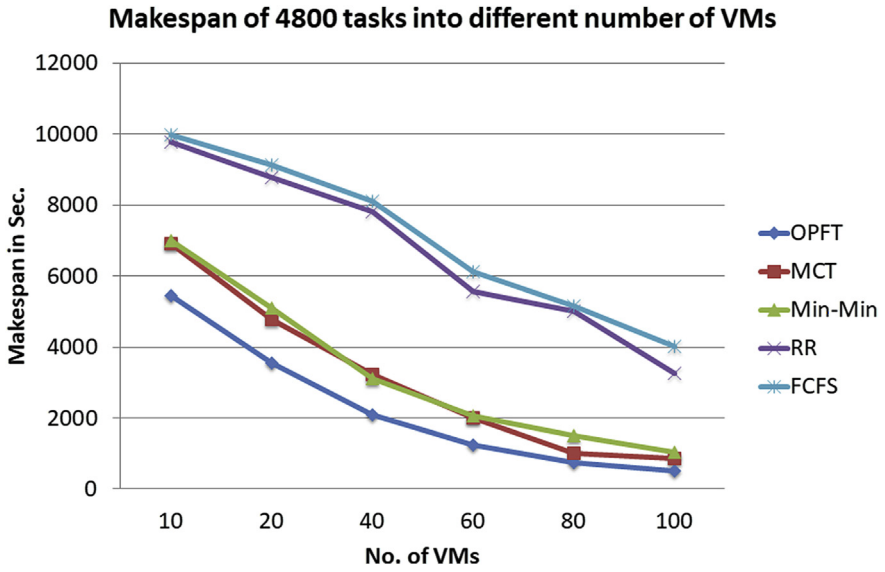


Fig. 7. Schedule length results of 4800 tasks.

assigned to each *VM*. $\alpha$ value can eliminate the amount of gab between *PFT* and the actual *TFT* of each *VM*. We start with small value ($\alpha = 1.00001$) and add another value $=0.00001$ for each iteration. This gives good results.

### 4.2. Time complexity

The time complexity of the proposed *OPFT* algorithm is analyzed according to the algorithm that is shown in Fig. 2. The time complexity is the summation of time complexity of *stage 1* and *stage 2*. In the stage 1, the *OPFT* sorts the arrived tasks by using the insertion sort with time complexity $O(n \log n)$. After that it calculates *TFT* for each *VM* with time complexity $O(m)$. Finally, it assigns each task to the fit *VM* with time complexity $O(n)$. So we can say that the total time complexity is $O(n\ (\log n + 1) + m)$ or $O(n \log n + m)$. If the number of VMs is small, We can rewrite it as $O(n)$.

### 4.3. Case study

To understand the working sequence of the proposed *OPFT*, let's take an example to assign ten independent tasks, shown in Table 1, into three *VM*s. The $VM_0$ has 550 *MIPS*, $VM_1$ has 250 *MIPS*, and $VM_2$ has 200 *MIPS*. So the $total_{Avial} = 1000$ *MIPS*. The proposed *OPFT* schedules the tasks into the available three *VMs* by applying two stages.

**In the first stage**: the OPFT collects information about tasks requirements and resources availability. Indeed, it sorts

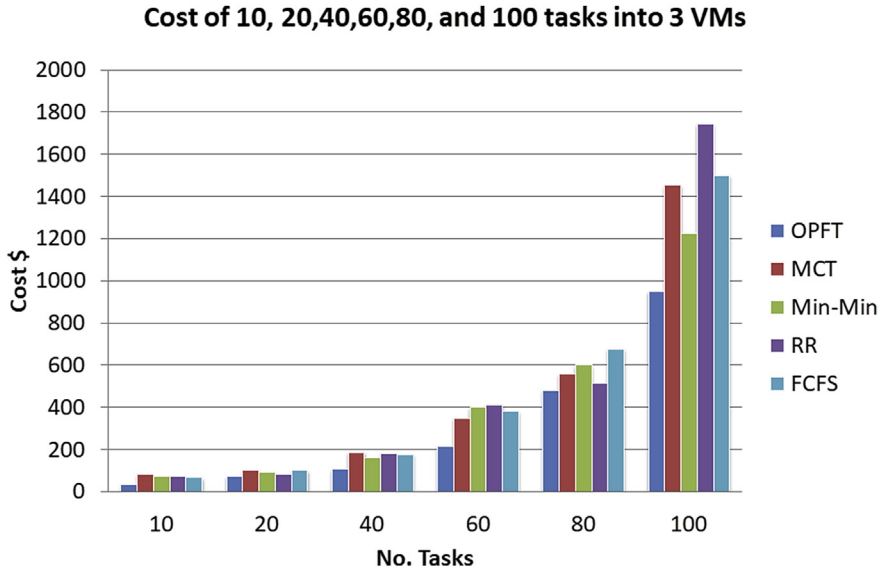


Fig. 8. The cost results of 10, 20, 40, 60, 80, and 100 tasks into 3 *VMs*.

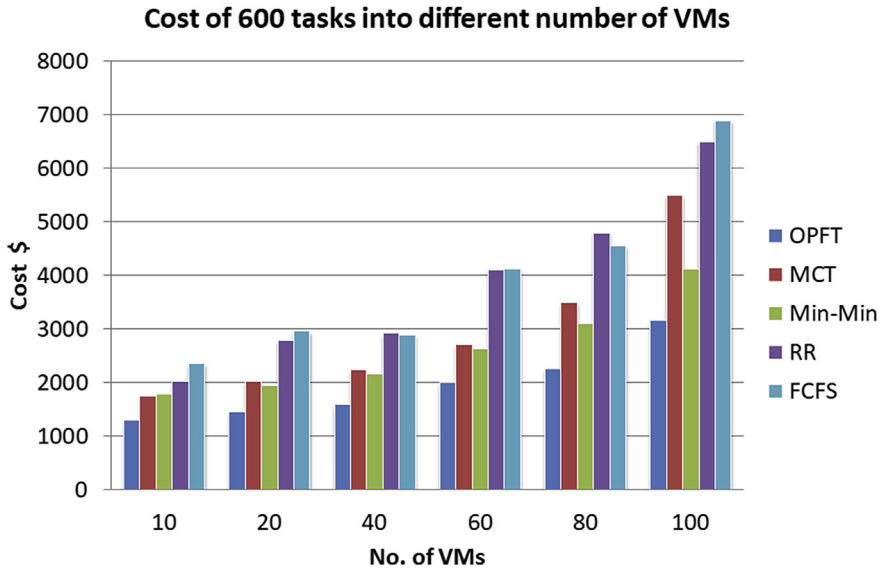


Fig. 9. The cost results of 600 tasks.

the tasks in descending order according to their lengths (i.e. number of MI). After that it computes the *PFT* value according to Equation (3). *PFT* = (*31,700* + *3170*)/ *1000* = *34.87 s.*

In the second stage: the algorithm assigns the ordered tasks according to the *PFT* value. Table 2 shows the scheduling solution.

## 5. Simulation results

This section presents performance evaluation of the proposed *OPFT*. In this evaluation, the well-known *CloudSim* [14] is used to simulate the cloud-computing environment. The simulation environment is a 64-bit windows 7 operating system installed in laptop core i5 with 8 GB RAM.

To evaluate the performance of the new algorithm, a list of tasks is generated by using a standard formatted workload of a high-performance computing center called *HPC2N* in Sweden as a benchmark [15]. In addition, a list of *VM*s is generated according to Amazon EC2 model shown in Table 3. We compare the new algorithm against *FCFS, RR, Min-Min, and MCT* algorithms as the most famous algorithms which are used as schedulers in cloud computing.

We have used some of metrics to measure the performance of our paper and compare it against the others. These metrics are a schedule length, cost, and balancing degree. Also we

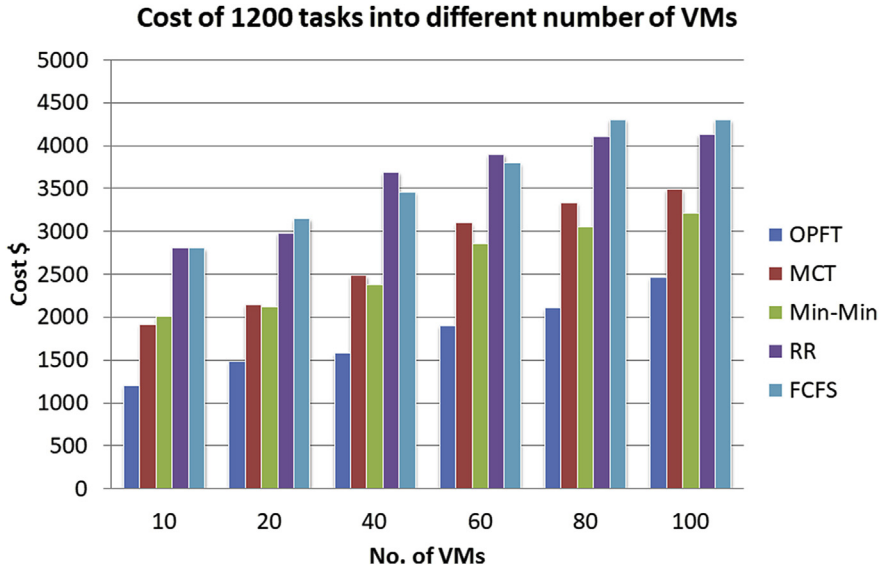


Fig. 10. The cost results of 1200 tasks.

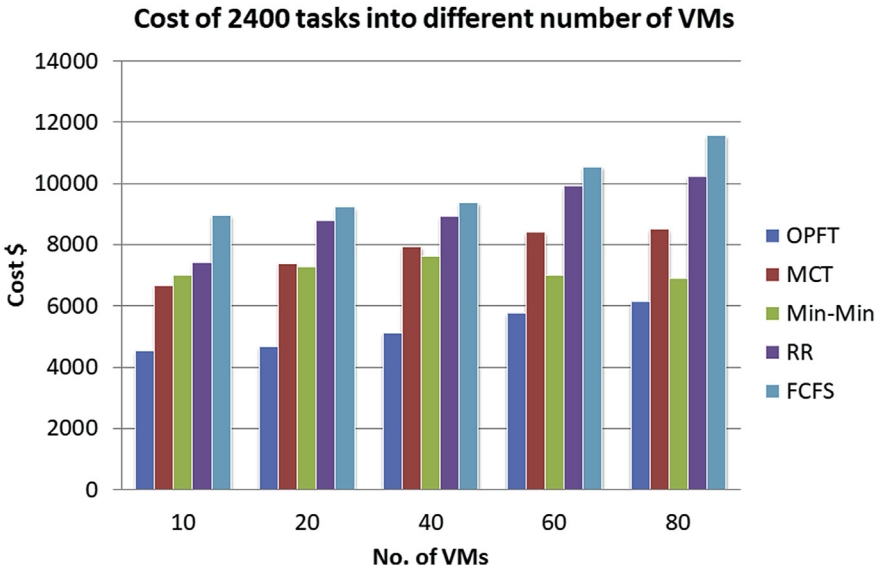


Fig. 11. The cost results of 2400 tasks.

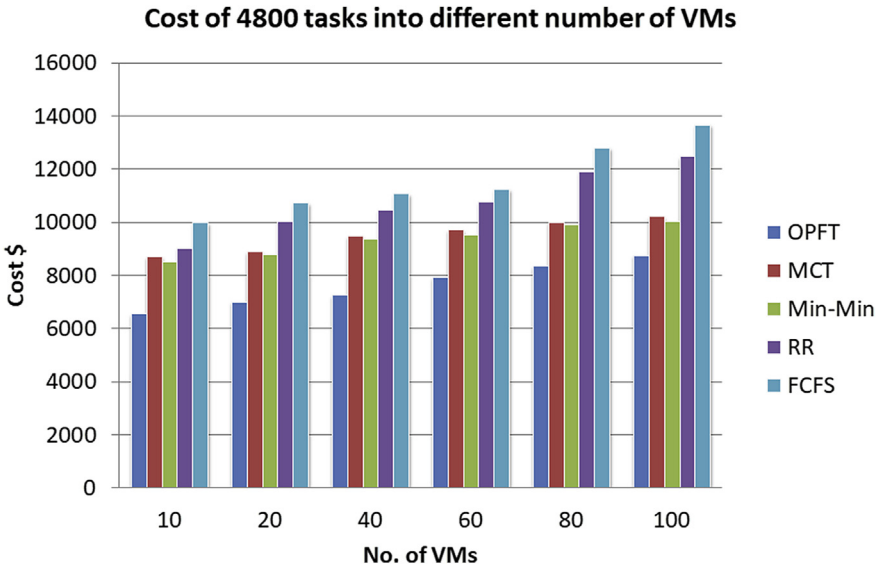


Fig. 12. The cost results of 4800 tasks.

show how the new algorithm can improve in resource utilization, response time and time complexity.

### 5.1. Schedule length (SL)

The *SL* or makespan is the maximum execution time at the most loaded virtual machine. It is the most important metric which should be used to show the efficient of any new algorithm. Figs. 3–7 show the schedule length results of different group of tasks which are assigned into different *VMs*. The results show that the new algorithm is better than the other algorithms. It gives low *SL*. As may be seen below, *OPFT* algorithm can manage any number of *VMs* to achieve the best solution. Let's take an example to show that. With number of 100 tasks, the new algorithm gets $SL = 700$ sec., whereas *MCT* and *Min-min* gives more 780 Sec., and *FCFS* and *RR* algorithms achieve SL is more than 1000 S these results are taken at 3 *VMs*. Another example, let's take 600 tasks at 10 *VMs*. We find that *OPFT* algorithm can finish the tasks execution and save more than 16 minutes than the other algorithms. Finally if we take another case at 4800 tasks and 40 *VMs*, we will find that *OPFT* algorithm can save more than 15 minutes than the other algorithms. In summary, we can say that our algorithm can achieve the solution with a lowest schedule length.

### 5.2. Cost

Amazon EC2 presents four different instances to the end user. Each instance has special way to pay. These instances are On-Demand, Reserved Instances, Spot Instances, and dedicated hosts. We choose On-Demand model to evaluate our algorithm, because it is famous and has many advantages for

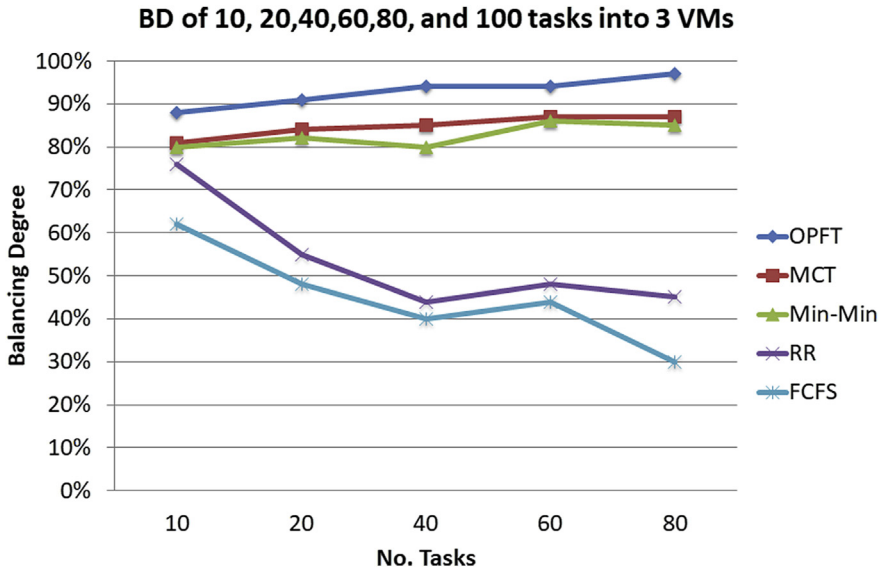


Fig. 13. The balancing degree results of 10, 20, 40, 60, 80, and 100 tasks at 3 *VMs*.

users. See Table 3, where we select some of different *VM*s with different characteristics [16].

Figs. 8–12 show the cost results of different tasks on different *VMs*. From the figures, we note that the *OPFT* algorithm achieves solutions with very low cost than the other algorithms.

In summary, we notice that our algorithm gives inexpensive solutions.

### 5.3. Balance degree (BD)

The optimal solution of the task scheduling problem may be achieved by applying an optimization method such as exhaustive search algorithm or branch-and-bound. However, the drawbacks of applying such methods are that they have very large time complexity and it is very difficult to be used in case of large number of tasks. Therefore, most of the research works trend to use heuristic methods, which achieve near optimal solution. The optimal solution ($S_{opt}$) may be defined as the best solution that achieves the lowest makespan. According to [3], the system can achieve the lowest makespan (i.e. optimal solution) if and only if the next conditions are met:

1. Each task is assigned to distinct *VM*.
2. Each task starts execution as soon as possible.

The *BD* is the degree of balancing the workload on the available *VMs* after scheduling. The *BD* may be calculated as follows.

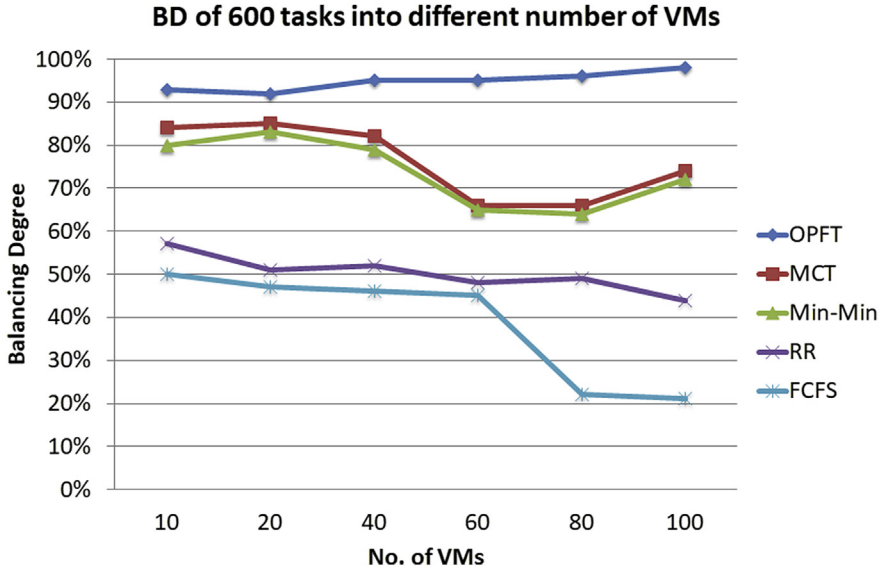


Fig. 14. The balancing degree results of 600 tasks.

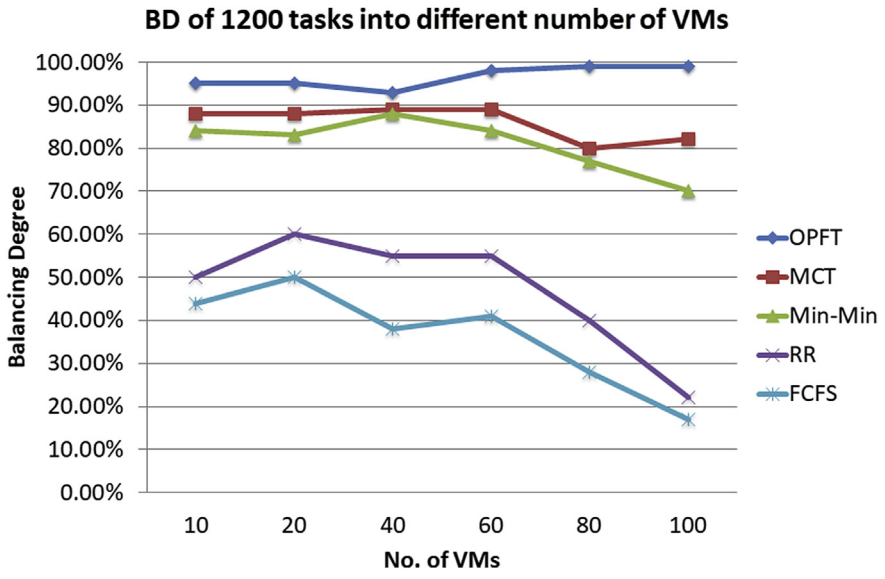


Fig. 15. The balancing degree results 1200 tasks.

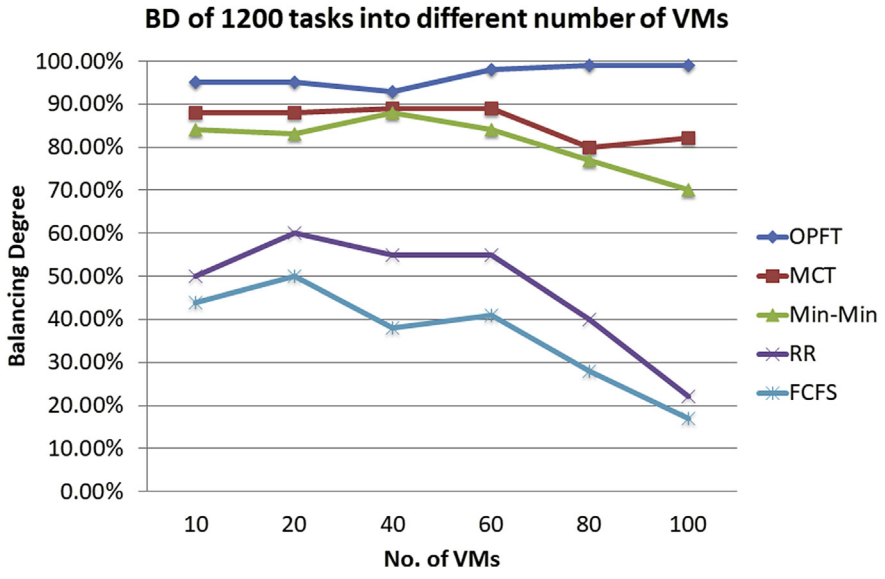


Fig. 16. The balancing degree results of 2400 tasks.

$$BD = SL(S_{opt})/SL(S_{fin}) \quad (4)$$

where, $S_{opt}$ is the optimal solution and $S_{fin}$ is the final solution obtained from the applied algorithm. In this article, $S_{opt}$ is assumed as the ideal solution, where $S_{opt}$ is computed as the summation of total *MI* of all the tasks over the total *MIPS* of the available *VMs*. That is, $S_{op}t = Total_{Req}/Total_{vaile.}$ From Equation (4), the algorithm with high balancing degree achieves the near optimal solution.

Figs. 13–17 show the results of *BD* for different tasks and different *VMs*. From the figures we see that the new algorithm is better than the other algorithms, because it gives the lowest *SL*. In the all cases, our algorithm has higher *BD* than the others. It is always close from the optimal solution and gives a degree more than 88%.

Authors in Ref. [17] define load balancing as "A technique to spread work between two or more computers, in order to get optimal resource utilization, maximize throughput, and minimize response time has high load balancing degree". So, we can say that our algorithm not only achieves the highest *BD* value, but also it minimizes the response time and utilizes the available resources in a good way.

### 5.4. Summary

From the above figures, we note that the new algorithm can improve in the cloud computing performance. It gives the near optimal solutions at different number of tasks and different number of VMs. The algorithm can achieve these solutions under load balancing consideration. Thus, it eliminates the response time and increases the resource utilization. The

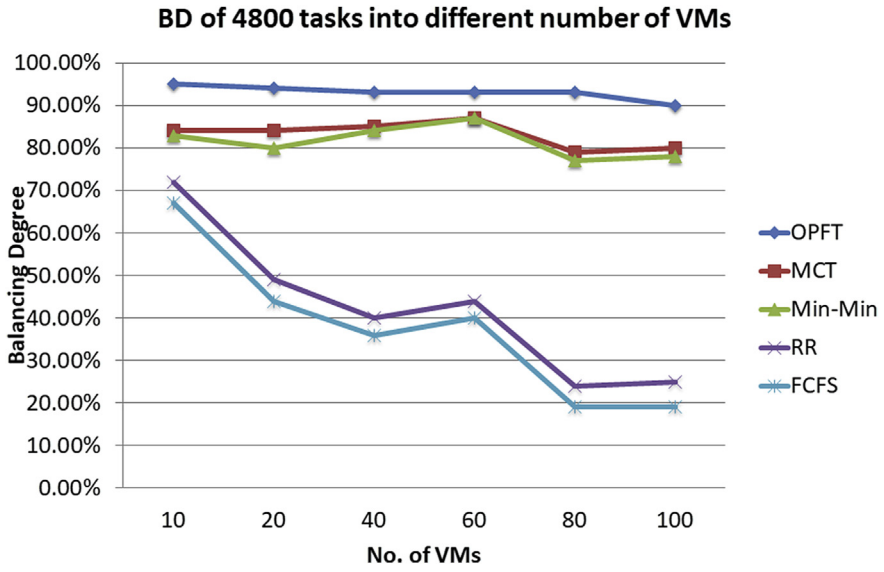


Fig. 17. The balancing degree results 4800 tasks.

Table 4
Summary of comparison between *OPFT* algorithm and other algorithms.

| Algorithm | OPFT | FCFS | RR | MCT | Min-Min |
|---|---|---|---|---|---|
| Time complexity | O(n) | O(1) | O(1) | O(mn) | O(mn) |
| Schedule length | Low | V. High | V. High | V. High | Medium |
| Cost | Low | V. High | V. High | V. High | High |
| Balance degree | V. High | V. low | V. low | Medium | Medium |
| Resource utilization | V. High | V. low | V. low | Medium | Medium |

algorithm also considers any new submitted tasks. It acts as a static algorithm for submitted group of tasks and as a dynamic algorithm after completing the scheduling of this group. It always checks if a new group of tasks are submitted or not. Table 4 shows comparison between the new algorithm and *FCFS, RR, Min-Min, and MCT* algorithms. From the table we see that *OPFT* algorithm is better than the others in terms of SL, BD, cost, resource utilization. However, FCFS and RR are better than it in time complexity. But if we compare the new algorithm against *FCFS and RR* with considering the total time of execution, we will find that the new algorithm is better than them, where the total time of execution is the submission of *SL* plus Running time (i.e. running time present time complexity in seconds).

## 6. Conclusion

In this article, a new *OPFT* for task scheduling in cloud computing is proposed. The main idea of the new algorithm is inspired from the vehicles traffic in the highways, where the width of the way has an important effect in the vehicles speed and their arrival time. Thus we can allocate a submitted group of tasks by using this rule, where the number of tasks that are allocated to specific virtual machine proportion to the power processing of this virtual machine. It applies this in two stages; the preparing stage and the selection stage. From the results, we find that the new algorithm can achieve the best solution at low running time. It improves *QOS* for cloud system through considering some terms like execution time, cost, response time, load balancing degree, and resource utilization. The results of our algorithm are always close from the optimal solution value. It always gives balancing degree higher than 88%. As a future work, we will use the algorithm to develop meta-heuristic algorithms. It will be used as an initial stage for some of meta-heuristic algorithms like Genetic Algorithm *GA*, Simulated Annealing *SA*, and ant Colony Optimization algorithm *ACO*. This is because; it gives low schedule length and low time complexity. This makes it one of good choices to be initial stage and give good solutions.

## References

[1] Shawish A, Salama M. Cloud computing: paradigms and technologies. Springer-Verlag Berlin Heidelberg; 2014.
[2] Hugos M, Hulitzky D. Business in the cloud: whatever business needs to know about cloud computing. Hoboken, New Jersey: John Wiley Sons, Inc.; 2011.
[3] Mei L, Chan WK, Tse TH. A tale of clouds: paradigm comparisons and some thoughts on research issues. Proceedings of the APSCC. 2008. p. 464–9.
[4] Mathew T, Sekaran K, Jose J. Study and analysis of various task scheduling algorithms in the cloud computing environment. International conference on advances in computing, communications, and informatics (ICACCI). 2014. p. 658–64.
[5] Nasr A, El-Bahnasawy N, Attiya G, El-Sayed A. Using the TSP Solution Strategy for Cloudlet Scheduling in Cloud Computing. Journal of Network and Systems Management 2018:1–22.
[6] Alworafi MA, Dhari A, El-Booz SA, Nasr AA, Arpitha A, Mallappa S. Enhanced Task Scheduling in Cloud Computing Based on Hybrid Approach. Singapore: In Data Analytics and Learning (pp. 11-25). Springer; 2019.
[7] Sinnen O. Task scheduling for parallel systems. Wiley series on parallel and distributed computing. Wiley-Interscience; 2007.
[8] Nasr A, EL-Bahnasawy N, El-Sayed A. A new duplication task scheduling algorithm in heterogeneous distributed computing systems. Bull Electr Eng Inf (BEEI) September 2016;5(No. 3):373–82.
[9] Kimpan W, Kruekaew B. Heuristic task scheduling with artificial bee Colony algorithm for virtual machines. The 8th international conference on soft computing and intelligent systems (SCIS) and 17th international symposium on advanced intelligent systems (ISIS). 2016.

[10] Hememalini M. Review on grid task scheduling in distributed heterogeneous environment. Int J Comput Appl 2012;40(2):24–30.
[11] Pavithra B, Ranjana R. A comparative study on performance of energy efficient load balancing techniques in cloud. International conference on wireless communications, signal processing and networking (WiSPNET). 2016. p. 1192–6.
[12] Chaturvedi P, Sharma S. A provision for workflow scheduling using round Robin algorithm in cloud. Int J Adv Res Comput Sci May-June 2017;8(5).
[13] Chen H, Wang F, Helian N, Akanmu G. User-priority guided min-min scheduling algorithm for load balancing in cloud computing. National conference on parallel computing technologies (PARCOMPTECH). 2013. p. 1–8.
[14] Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: practice and experience. Wiley; 2010.
[15] Feitelson DG, Tsafrir D, Krakov D. Experience with using the parallel workloads archive. J Parallel Distr Comput 2014;74(10):2967–82.
[16] Golden B. Amazon web services for dummies. John Wiley & Sons; 2013.
[17] Abdulmohson A, Pelluri S, Sirandas. Energy efficient load balancing of virtual machines in cloud environments. Int J Cloud Comput Supercomput 2015;2(1):23–38.

**Aida A. Nasr** is a PHD candidate in the Dept. of Computer Science and Engineering, Faculty of Electronic Engineering, Menofia University, Egypt. She is lecturer in The Higher Institute of Engineering and Technology (THIET), Egypt. She received M.SC in 2015 from the Faculty of electronic Engineering, Menofia University, Egypt. She graduated from Faculty of electronic Engineering, Menofia University, Egypt in 2010.

**Nirmeen A. El-Bahnasawy,** she is Lecturer at Computer Science and Engineering Department – Faculty of Electronic Engineering – Menofia University – Egypt. She received PhD. degree in computer science and engineering. "Task Scheduling in High Performance Heterogeneous Computing Systems" Dept. of Computer Science and Engineering, Faculty of Electronic Engineering, Menofia University, EGYPT in 2013. She gained M.Sc. degree in computer science and engineering. "Task dispatching for parallel vector Processors" Dept. of Computer Science and Engineering, Faculty of Electronic Engineering, Menofia University, EGYPT in 2003. Her main research interests include distributed computing, task allocation and scheduling, computer Networks, load balancing, cloud and grid computing.

**Gamal Attiya** graduated in 1993 and obtained his M.Sc. degree in computer science and engineering from the Menofia University, Egypt, in 1999. He received PhD degree in computer engineering from the University of Marne-La-Vallée, Paris-France, in 2004. He is currently Associate Professor at the department of Computer Science and Engineering, Faculty of Electronic Engineering, Menofia University, Egypt. His main research interests include distributed computing, Task Allocation and Scheduling, Cloud Computing, Big Data Analysis, computer networks and protocols, congestion control, QoS, and multimedia networking.

**Ayman El-Sayed,** Professor and head of computer science and Engineering department, He received the BSc degree in computer science and engineering in 1994, the master's degree in computer networks in 2000 from the University of Menofia, Egypt, and the PhD degree in computer network in 2004 from "Institute National De Polytechnique De Grenoble" INPG, France. He is an associate professor in the Computer Science and Engineering Department, Faculty of Electronic Engineering, Menofia University, Egypt. He is specialized in soft computing, algorithms, data structure, and cloud computing and big data analysis. In addition, his interests include multicast routing protocols, application-level multicast techniques, multicast on both mobile network and mobile IP, and image processing techniques. In addition, there are other interesting topics such as bioinformatics, Biocomputing, and bio computer. He is an approved supervisor for M.Sc. and Ph.D. programs in various University. He has completed various project in government and private organization. He has published more than 100 research papers in international Journals and two books about OSPF protocol and multicast protocols. Currently, he is serving as an editorial board member in various international Journals and conferences. He is a senior member of the IEEE.