Journal of Engineering Science and Technology
Vol. 13, No. 8 (2018) 2299 - 2317

AN ENHANCED SCHEDULING APPROACH WITH CLOUDLET MIGRATIONS FOR RESOURCE INTENSIVE APPLICATIONS

NEELAM PANWAR[1,*], SARITA NEGI[2], MANMOHAN S. RAUTHAN[1], MAYANK AGGARWAL[3], PRAGYA JAIN[4]

[1]School of Engineering & Technology, HNB Garhwal University
Srinagar Garhwal Uttarakhand, India
[2]Computer Science and Engineering, Uttarakhand Technical University,
Dehradun, Uttarakahd,India
[3]GurukulKangri University, Haridwar, Uttarakhand, India
[4]Computer Services Centre, IIT Delhi, India
*Corresponding Author: neelam.panwar001@gmail.com

Abstract

Cloud computing is one of the most advanced technologies to present computerized generation. Scheduling plays a major role in it. The connectivity of Virtual Machines (VM) to schedule the assigned tasks (cloudlet) is a most attractive field to research. This paper introduces a confined Cloudlet Migration based scheduling algorithm using Enhanced-First Come First Serve (CM-eFCFS). The objective of this work is to minimize the makespan, cost and to optimize the resource utilization. The proposed work has been simulated in the CloudSim toolkit package. The results have been compared with pre-existing scheduling algorithms with same experimental configuration. Important parameters like execution time, completion time, cost, makespan and utilization of resources are compared to measure the performance of the proposed algorithm. Extensive simulation results prove that introduced work has better results than existing approaches. 99.8% resource utilization has been achieved by CM-eFCFS. Plotted graphs and calculated values show that the proposed algorithm is very effective for cloudlet scheduling.

Keywords: Cloud computing, Cloudlet, Cloudlet migration, Resource utilization, Virtual machine.

1. Introduction

Cloud computing technology is emerging as standard advanced computer technology where the network functioning knowledge is no longer needed for the user. This computing paradigm contributes delivering application based services and resources over the web-shared pool as per the user demand. The cloud computing characteristics include on-demand services, broad network access, resources pooling, measured service, reliability, etc. The user utilizes different cloud service models, i.e., Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IaaS especially provides the storage resources (e.g., Amazon Elastic Compute Cloud, GoGrid, Nimbula, etc.) whereas PaaS allows users to develop the application in the cloud (e.g., Google App Engine, Github, Gigaspaces, etc.) and lastly, SaaS enhances user services by providing complete software application (e.g., Google App, Facebook, Linkedin, Slideshare, etc.). To deploy these cloud service, cloud computing categorizes deployment models into Public, Private, Hybrid and Community cloud. Each deployment model handover the cloud services to multi-user, single-user and specific-user. Figure 1 cites the cloud computing architecture with cloud services and deployment model. The major components of the cloud are Application, Client, Infrastructure, Platform, Service, Storage and Processing Power.

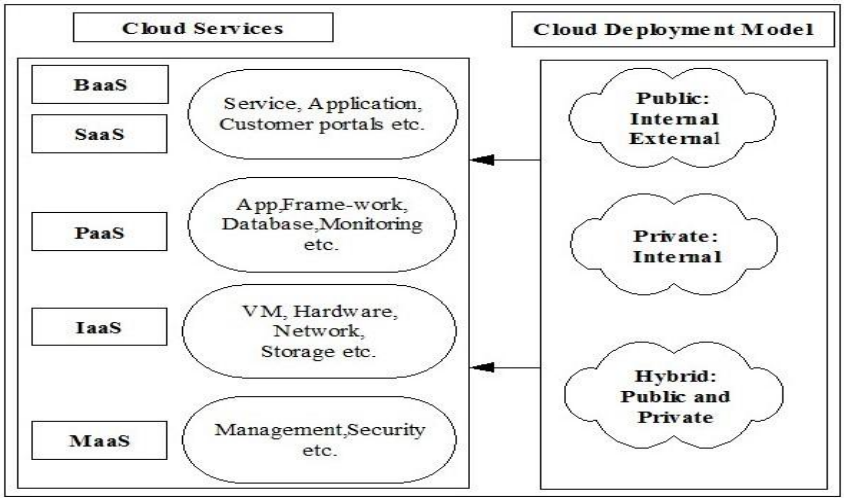

Fig. 1. Cloud computing service model.

The popularity of cloud computing enhances by taking care of better resource utilization, average cost, load balancing factor, completion time, execution time, average power consumption, network latency, bandwidth, average energy consumption, makespan and response time, etc. Apart from the various outperform services delivered by cloud technology, Virtualization; Load Balancing; Fault Tolerance; Security and Scheduling (VM and Cloudlets) are the major issues to be concerned. There are two main areas to be scheduled, i.e., allocation of the processor to VM, which is done by VM scheduler and submission of cloudlet to VM, which is done by cloudlet scheduler. Scheduling of cloudlets is one of the major challenges in cloud technology to achieve highly efficient computations among the machines [1-3].

The procedure of searching the needed resources is the same as the procedure of searching the various VMs, as the needed resources together form VMs. Users send requests to the datacenter to execute their cloudlets. A cloudlet may involve entering and processing data, accessing software or some storage functions. Cloudlets are submitted to the cloudlet scheduler. It is the responsibility of cloudlet scheduler to submit cloudlet to appropriate VM. Then the mapping between cloudlet and required resources is performed. The cloudlet must execute and a reply is conveyed to the user. Cloudlets can either be independent, dependent or the combination of both. The selections of types of Cloudlets are vital to introduce an efficient Cloudlet scheduler [4]. Research on aperiodic and sporadic types of Cloudlets in a cloud environment has been introduced [5].

Cloudlet scheduling in a cloud environment can be defined as a process of choosing appropriate resources offered for performing cloudlets execution. These resources are used to assign cloudlets to the VMs in order to minimize makespan and improve resource utilization. Different scheduling algorithms are proposed in [3, 6-13]. These algorithms consider different parameters including makespan (finish time), priority vector, comparison matrix, load balancing, cost and bandwidth. In scheduling algorithm, sorting of cloudlets is performed by assigning priority to each cloudlet where priorities are assigned on the basis of numerous factors such as cost and deadline. Cloudlets are selected on the basis of their priorities and then allocated to the offered resources and VMs, which satisfy objective function defined by cloudlets [14].

In this work, a new cloudlet scheduling method is introduced and its performance has been compared with some pre-existing scheduling algorithms (FCFS, SJF, RR, and Min-Min). This paper demonstrates the cloudlet scheduling by using a method called cloudlet migration. Some cloudlet migration and VM migration techniques are introduced in [15-18]. The results clearly prove that the proposed algorithm gives better result in terms of execution time, processing cost and utilization of resources. The primary objectives of this paper are as follows:

- Formulation of a method to schedule the cloudlets in a cloud environment to reduce the makespan (overall time of execution), cost (processing) and increase the resource utilization.
- Design of CM-eFCFS algorithm (Enhanced- First Come First Serve with Cloudlet Migration) for cloudlet scheduling based on the cloudlet migration method.

The entire work can be described as the development and implementation of a new scheduling algorithm CM-e FCFS, which comprises of a concept called "Cloudlet Migration". Previously proposed scheduling algorithms were totally based on the type of cloudlets assigned to the VMs but this work focuses on the cloudlet migration. Cloudlets are assigned to each VM and the VM, which executes its assigned cloudlets early, can lead to the next partially executed cloudlet to be migrated, which is assigned to the slower VM in a pre-emptive way.

This paper is systematized as follows. Section 1 introduces the cloud computing model. Section 2 explores definitions and literature review related to the proposed work. Section 3 provides the system model for the proposed algorithm. Section 4 provides research implementation of designed algorithms. Simulation results and their comparison are elaborated in section 5. In section 6, we conclude our contributions with future scopes to the work.

2. Definition and Literature Review

In cloud, scheduling plays a significant role to increase reliability, efficiency and flexibility of the system. In a decade, there has been an increasing interest in cloudlet scheduling over cloud computing. It has been analysed that the scheduling algorithms can be categorized on the basis of objective functions. Objective functions can be either application-centric or resource centric. In application-centric, an algorithm beneficial for the user is designed to reduce the execution span and cost. In the case of resource-centric, an algorithm beneficial for service provider designed to maximize resource utilization and profit. Several cloud-based task scheduling schemes have been proposed [19] and such related schemes are reviewed in this section. The performances of each algorithm measured with the specified metrics to describe the processing unit in the resource system are as follows:

- **Makespan**: The complete time that passes from the beginning of the scheduling process to the end. To maintain the efficiency of the algorithm makespan should always as low as possible.
- **Cost**: From the user's perspective the computing cost should be less. Service providers spend a huge amount of money in network, storage and compute, hence the effective scheduling algorithm must take care of the cost.
- **Execution time**: The precise time required to execute cloudlet is known as the execution time. An appropriate scheduling algorithm is aimed to diminish the execution time.
- **Completion time**: The time in which the entire execution of the cloudlet is completed is known as completion time. It comprises the delay caused by the cloud environment and the total execution time. Many scheduling algorithms aim to reduce completion time for the better performance.
- **Resource utilization:** The average utilization of resources is defined as the percentage of the total time for which the processor is performing some operation and is not idle.
- **Delay time:** The time duration by which the execution of a cloudlet is delayed due to cloudlet migration.

From the decades, the basic task scheduling algorithms: First Come First Serve (FCFS), Shortest Job First (SJF) and Round Robin (RR) have been focused to enhance the advancement of scheduling. In this paper, the proposed scheduling algorithm is compared with these basic algorithms to find out the benefit of the proposed method. FCFS works on the First Come First Serve principle, where the cloudlet that arrives first will be assigned to VM first. Although, the algorithm is simple to implement but not appropriate for heavy (lengthy) cloudlets as it leads to higher makespan. SJF takes up the cloudlet with the shortest length first and assigns it to VM.SJF leads to the higher waiting time for long cloudlets if short cloudlets keep coming. RR focuses on assigning cloudlets to each VM equally. Using this algorithm, the Scheduler allocates one cloudlet to a VM in a cyclic manner. RR works on time slice manner so it uses Time-Shared Cloudlet Scheduler. The introduced work has overcome the problems and shortcomings of these basic algorithms.

Tawfeek et al. [3] and Dorigo and Blum [20] introduced an Ant Colony Optimization based algorithm (ACO) that exhibits food searching behaviour of ant colonies. An army of ants uses pheromone chemical to communicate while

searching for food. The ACO algorithm initializes pheromone, chooses VM for next cloudlet and updates pheromone. The scheduling problem symbolized using a graph $G = (N, E)$, where N denotes VMs and cloudlets and E denotes the connections between the number of cloudlets and VMs. It was assumed that cloudlets were independent of each other, i.e., the execution of one cloudlet has no effect on the execution of other cloudlets. Also, cloudlets were considered non-pre-emptive and non-interruptible.

Zaa et al. [16] introduced a lightweight task migration technique where migration is performed between the mobile device and cloud node. In this process, mobile application uses some resources from the cloud. The inputs are provided by a mobile device as a remote control. Two different cloud servers are used. When the input is provided to the mobile device, it looks for resources to execute a particular task. If resources are available, the task is executed on the same cloud, if the resources are not available then it migrates that task to another cloud for further execution resulting into load balancing.

Lin et al. [8] introduced Bandwidth Aware Divisible Task Scheduling (BATS). Under the bounded multi-port model [9], a nonlinear programming model was introduced. The model obtains enhanced allocation scheme to define an appropriate set of cloudlets allocated to each resource. On the ground of the enhanced scheme of allocation, BATS algorithm was proposed. The algorithm allocates the suitable number of cloudlets to all resources according to their CPU capacity, memory, network bandwidth and space.

Santhosh and Manjaiah [21] worked to introduce Improved Max-Min based algorithm. In the algorithm, an exceptional variation to the improved max-min [22] algorithm was performed, where a task is selected whose CPU time is larger than the average execution time. The task allocated to the resource, which completes the task in minimum time. The algorithm overcomes some limitations of Max-Min. The foremost drawback of the Max-min was that the execution of the smaller jobs was delayed and postponed indefinitely, which has been taken care of the proposed algorithm.

Chawla and Bhonsle [23] proposed dynamically optimized cost based task-scheduling algorithms, which were based on the priority of user tasks. Greedy scheduling from user's perspective results in wastage of resources whereas from service provider's perspective may result in dissatisfaction of user on QoS constraints. The proposed algorithm adds up the two necessities, the first one, cost-based task scheduling which was advantageous to the user and next dynamically improved resource allocation approach to the service provider. The computation/communication ratio and utilization of accessible resources are well optimized through an assemblage of the user tasks prior resource allocation.

Chen et al. [24] proposed User-Priority Guided Load-Balanced Min-Min Algorithm (PA-LBIMM). The algorithm was introduced to provide satisfactory utilization of resources provided by the cloud services. User priority was not considered by Yu and Yu [10]. The biggest disadvantage of [10] was it did not consider load balancing. An improved load balanced algorithm was further introduced which is grounded on the Min-Min algorithm and increases the utilization of resources by shifting some load from heavy loaded resources to resources having zero load, i.e., LBIMM, but it does not care about the user priority necessity of cloudlets. For this reason, the concept of user priority was considered

in User-priority aware load balance improved Min-Min (PA-LBIMM*)* algorithm, so that user's demand could be fulfilled more completely and satisfactorily.

Priyadarsini and Arockiam [25] introduced the Min-Min Algorithm, in which the least possible completion time is evaluated for all cloudlets on each resource. The cloudlet with least possible completion time is picked and assigned to a resource that provides it. The recently mapped cloudlet is detached from cloudlet-list and this procedure is repeated until entire cloudlet list is mapped to resources. Min-Min is an unsophisticated and speedy algorithm, which accomplished good performance. Min-Min schedules "best case", cloudlets first producing good schedules, but allocating small cloudlet first is its disadvantage. This Min-Min approach has been modified in this paper by using enhanced FCFS together with it.

Awad et al. [26] introduced a mathematical model, which is based on Load Balancing Mutation a Particle Swarm Optimization (LBMPSO) based scheduling and allocation. The scheduling takes care of makespan, execution time, transmission time, round trip time, reliability, transmission cost and load balancing between VMand cloudlets. LBMPSO considers the resource availability and reschedules cloudlets that failure to allocate hence, can play a major role in accomplishing reliability.

Hassan Ali et al. [27] worked to introduce the Grouped Tasks Scheduling (GTS) algorithm that applies QoS to schedule the cloudlets in the cloud-computing environment in order to fulfil user's requirement. GTS algorithm categorizes cloudlets into five groups; each group has cloudlets having similar characteristics (cloudlet type, length of cloudlet, latency and user type). By adding cloudlets to the accurate group, it starts scheduling these cloudlets into available resources. Scheduling is performed in two stages: in the first stage, it is decided which group will be scheduled first. Which cloudlet inside the chosen group will be scheduled, is decided in the second stage and depends on the cloudlet size.

3. System Model and Proposed Work

The system model comprises a set of VMs=(VM_1, VM_2,.VM_j) in each host of a datacentre. A number of cloudlets=*(*C_1, C_2, C_i*)* are assigned to each VM respectively to perform the execution of cloudlets. Each VM runs on its own resources in parallel and independently. Figure 2 depicts the system architectural model of the simulation. The simulation work has been performed under CloudSimtool, which includes various cloud environment configurations such as Datacenter Characteristics: architecture, operating system, virtual machine monitor (VMM), host-list, time zone, CPU cost, cost per memory, cost per storage, and cost per bandwidth. Host Characteristics: RAM, bandwidth, storage, number of processing elements, MIPS, and VM scheduler (space-shared and time-shared). VM Characteristics: RAM, bandwidth, storage, number of processing elements, MIPS, and cloudlet scheduler (space-shared and time-shared). Cloudlet Characteristics: Length, input file size, output file size, number of required processing elements, utilization model. Datacenter-Broker Methods: VM creation, Cloudlet submission to VM, and VM destruction.

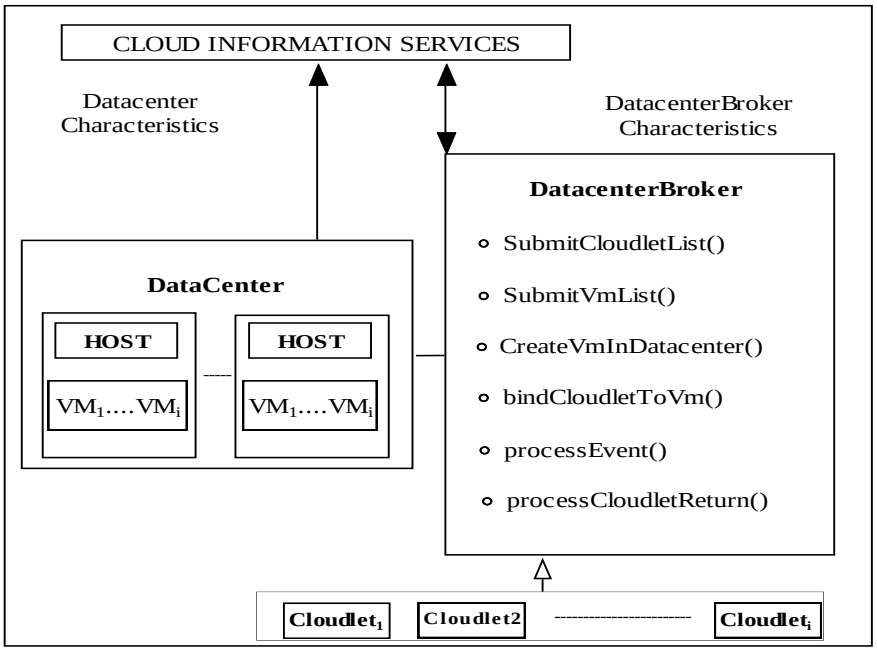

Fig. 2. System architecture.

3.1. Algorithm description for proposed approach

Cloudlets are assigned to VMs in the same order as they arrive in the system. *ECT* of first cloudlet is calculated for each VM and that cloudlet is assigned to VM, which gives minimum *ECT*. This procedure is repeated for rest of the cloudlets. Figure 3 shows the basic working mechanism of the algorithm, which is explained as:

The ECT_{ij} is calculated using $RT_{j,}$ which is the clock time of the system (CLK_{sys}=*0.1* ms*)* and EET_{ij}. Equations (1) to (3) formulate the RT_j, EET_{ij} and ECT_{ij} respectively [25].

$$RT_j = CLK_{sys} \tag{1}$$

$$EET_{ij} = {L_{ci}}\Big/{\sum_{k=1}^{PE_j} MIPS_k} \tag{2}$$

$$ECT_{ij} = RT_j + EET_{ij} \tag{3}$$

where j=1, 2, 3,…and i=1, 2, 3,…. symbolized the set of VMs and number of cloudlets respectively. The calculated values of EET_{ij} with four VMs shown in Fig. 4 where each cloudlet with subsequent length ranges from 2000 to 5000 (Million Instructions) is presented. Figure 5 shows the EET_{ij} with eight number of VMs. These outputs are used to compute ECT_{ij} in order to achieve $ERCT_{i,\ which}$ is calculated using Eq. (4).

$$ERCT_i = Min\left[ECT_{i1}, ECT_{i2}, ECT_{i3} \ldots \ldots ECT_{ij}\right] \tag{4}$$

Now, C_i is assigned to VM_{ie} and hence ready time of VM_{ie} is updated using Eq. (5).

$$RT_{VM_{ie}} = ERCT_i \quad (5)$$

This procedure is repeated until all the cloudlets are mapped to VMs, then all VMs start processing cloudlets. The VM, which processes its assigned cloudlets early, can make any partially executed or unprocessed cloudlet to be migrated, which is assigned to the slower VM in a pre-emptive way.

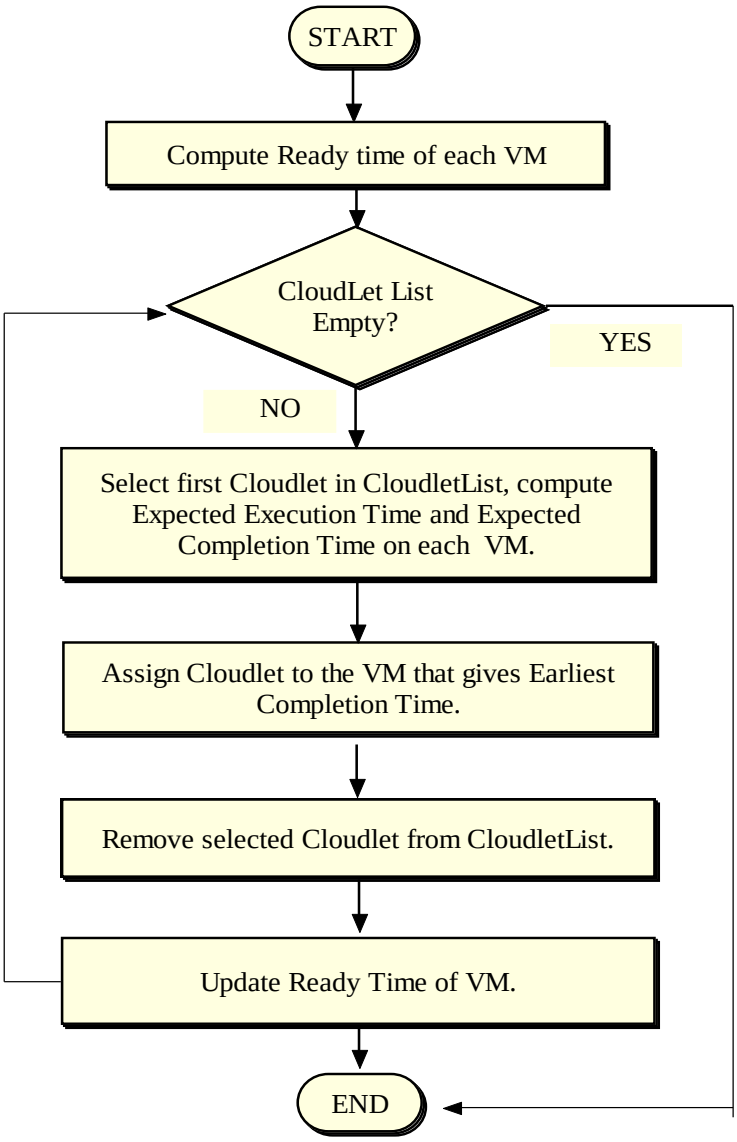

Fig. 3. Flow chart of CM-eFCFS.

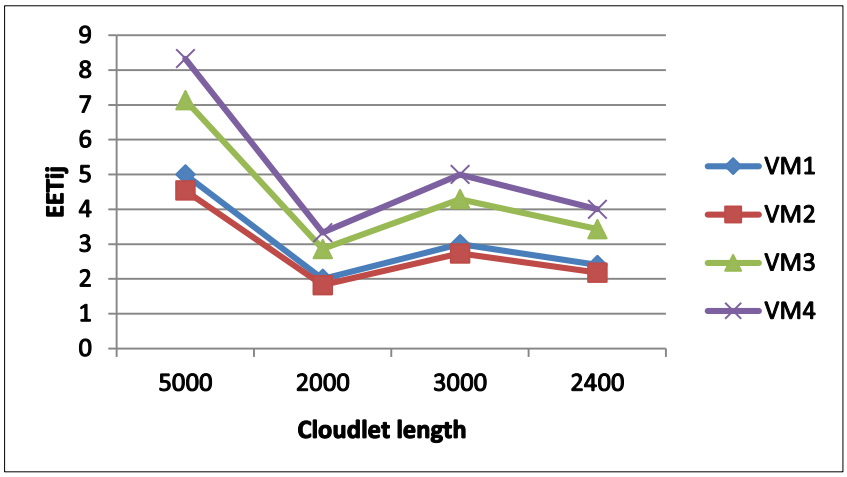

Fig. 4. Calculated expected execution time for 4 VMs.

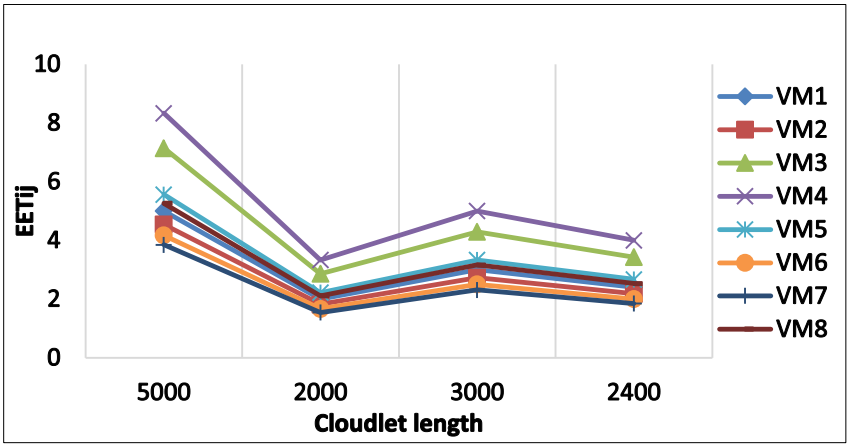

Fig. 5. Calculated expected execution time for 8 VMs.

3.2. Cloudlet migration

The cloudlets are assigned to the number of VMs. The VM has the highest capacity (*MIPS* of processing element) [14] executes its assigned cloudlets early, and hence a cloudlet which is partially executed or waiting for execution on any slower VM can be migrated to the faster VM in a pre-emptive way. The Expected completion time of cloudlet after migration can be evaluated by Eq. (6).

$$ECT_m = RT_j + {RL_m}/{\sum_{k=1}^{PE_j} MIPS_k} \quad (6)$$

where CLK_{sys} is the current time of the system, RL_m is the remaining length to be executed of the cloudlet, which has to be migrated. Figure 6 shows the cloudlet

migration from lowest processing VM to the highest processing VM. The algorithm of CM-eFCFS and cloudlet migration is showing in below section.

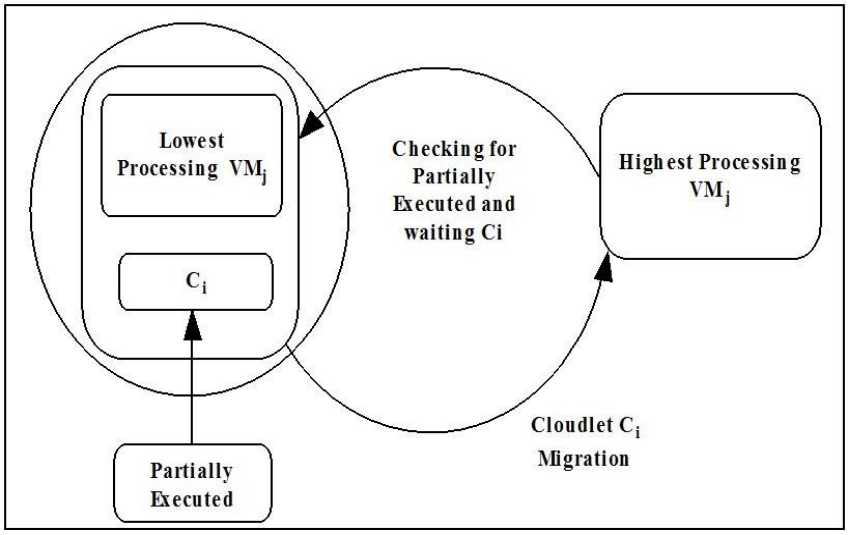

Fig. 6. Cloudlet migration.

3.3. Makespan

The complete time that elapsed from the starting of the scheduling process until the end is called makespan and is calculated as:

$$Makespan = \max(CT_j) \qquad j \in VMs \tag{7}$$

The j^{th} VM that has maximum completion time consider as makespan.

3.4. Delay time of migrated cloudlet

The delay time can be evaluated as:

$$Delay\ time = \frac{Size_i}{BW_j} = t2 - t1 \tag{8}$$

where t_2 is the time when cloudlet starts on the destination VM and t_1 is the time when cloudlet execution is stopped on slower VM, $Size_i$ is the remaining length of C_i to be executed and BW_j is the bandwidth of VM_j. The delay time of each migrated cloudlet for four VMs and six cloudlets as shown in Table 1. The table depicts that cloudlets 0,1,2,3 and 4 does not perform any migration hence no delay. But in case of cloudlet 5, $Size_i$=339.52 and BW_j =1000. Hence using Eq. (8), the delay calculated 0.339 ms.

3.5. Processing cost

The processing cost is the cost of processing in the current resource. To minimize the processing cost is a major problem in cloud computing environment. The overall processing cost of an algorithm can be evaluated as:

$$Processing\ Cost = \sum_{j=1}^{m} Cost * EVM_j \tag{9}$$

where m symbolizes the number of VMs.

Table 1. Calculated delay time of migrated cloudlet.

Cloudlet Id	Length	Source VM Id	Destination VM Id	*Size*	BW_j	Delay (ms)
0	2000	1	No migration	-	-	No
1	2000	0	No migration	-	-	No
2	2000	2	No migration	-	-	No
3	1980	3	No migration	-	-	No
4	1960	1	No migration	-	-	No
5	1940	0	VM 1	339.52	1000	0.339

ALGORITHM 1: CM-eFCFS Scheduler

(1) Identify the Ready Time of VMs, when cloudlets are submitted to broker for mapping.
   (a) For *j* from 0 by 1 to get Vms Created List. Size-1 do {Set Ready Time of Vm [*j*] = CloudSim.Clock}

(2) Identify the Earliest Completion time of each cloudlet by calculating the Expected Execution Time and Expected Completion Time of cloudlets on each VM. Assign cloudlet to VM which gives Earliest Completion time.
For *i* from 0 by 1 to get CloudletList. *Size*-1 do {
For *j* from 1 by 1 to get Vms Created List. *Size*-1 do {
   (a) Set Expected Execution Time of C_i on VM_j=Length of C_i/ capacity of VM_j
   (b) Set Expected Completion Timeo f C_i on VM_j= Ready Time of Vm [*j*] + Expected Execution Time of C_i on VM_j
   (c) To find Earliest Completion Time of C_i:
   Set Earliest Completion Time of C_i = 0
   If (*j* equals 1)
   Set Earliest Completion Time of C_i = Expected Completion Time of C_i on VM_1
   End If
   If (Expected Completion Time of C_i on VM_j <Earliest Completion Time of C_i)
   Set Earliest Completion Time of C_i= Expected Completion Time of C_i on VM_j
   Set vmid = *j*
   End If }
   Set vm = get Vms created list.get (vmid)
   Set Cloudlet.set VmId (vm.getId)
   Send Now(get Vms to datacenters Map.get (vm.getId), CloudSim Tags. CLOUDLET_SUBMIT, Cloudlet)
   //Assign Cloudlet to the VM that gives Earliest Completion Time for that cloudle
   Increase cloudlets Submitted List by 1
   Update Ready Time of Vm [vmid] = Earliest Completion Time of C_i }

3. Remove all the assigned Cloudlets from the cloudletList.

ALGORITHM 2: Cloudlet Migration

Identify the VM which has highest capacity.
Set highest $Capacity_{vm}$ = get Vm List. get(0).get Mips
(a) Set vm Index = 0
(b) For vm Number from 1 by 1 to total number of VMs do {
If (get VmList.get (vmNumber).get Mips greater than highest Capacity vm)
Set highest $Capacity_{vm}$ = get Vm List.get (vmNumber).getMips
Set vm Index = j
End If }
(c) Set Vm fastest VM = get Vm List.get(VmIndex)
(Identify, if VM with highest capacity has executed all the cloudlets assigned to it.
If (fastest VM.get Cloudlet Scheduler.running cloudlets = 0)
Find Cloudlet to be migrated and store related data in an array.
Set cloudlet to migrate = 0
(a) For Cloudlet cloudlet in get Cloudlet Submitted List do {
If (cloudlet. Get Cloudlet Status = (QUEUED or INEXEC) and cloudlet to migrate equals 0)
Set old VmId = cloudlet.get Vm Id
End If }
(b) Set cloudlet To Migrate = cloudlet.get Cloudlet Id
(c) Set User Id = cloudlet.get User Id
(d) Set intarr [] = new int [5]
(e) Set arr [0] = cloudlet to migrate
(f) Set arr [1] = UserId
(g) Set arr [2] = old VmId
(h) Set arr [3] = fastest VM getId
(i) Set arr [4] = datacentre Id
Migrate Cloudlet to new VM.
Send now (datacenterId, CloudSim Tags.CLOUDLET_MOVE, arr)
Start cloudlet execution on new VM with highest capacity.

3.6. Resource utilization

One of the major challenges in a cloud environment is to achieve high resource utilization. It is the percentage of total time for which the processor performs some operation and is not idle. The average utilization of an algorithm can be evaluated as [28]:

$$Average\ Utilization = \sum_{j \in VMs} CT_j \Big/ Makespan * Number\ of\ VMs \quad (10)$$

4. Implementation

The implementation is sub-categorized into two sections. The first section shows the configuration details of the simulation environment, whereas the second section describes the obtained results using CloudSim simulator.

CloudSim simulation environment

The CloudSim [29] toolkit has been used to simulate the proposed scheduling algorithm. The overview of cloud setup configuration for four and eight number of VMs is shown in Table 2.

Table 2. Cloud setup configuration details.

Tool configuration	No. of VM=4	No. of VM=8
System architecture:	X86	X86
Operating system:	Linux	Linux
VMM:	Xen	Xen
Host description		
RAM:	2048 (MB)	4096(MB)
Storage:	1000000	1000000
Bandwidth:	10000	10000
No. of PE:	4	8
VM description		
RAM:	512	512
Size (amount of storage):	10000 (MB)	10000
MIPS:	1000	1000
	1100	1100
	700	700
	600	600
	-	900
	-	1200
	-	1300
	-	950

5. Results and Discussion

This section elaborates the comparison of CM-eFCFS with other scheduling algorithms. The comparative analysis has been done with four pre-existing cloudlet-scheduling algorithms under the same cloud computing system configuration. The four pre-existing scheduling algorithms are FCFS, SJF, RR and Min-Min. For the purpose of analysing the performance of the proposed algorithm, the work is simulated for two different scenarios considering three vital parameters, i.e., Makespan, Resource Utilization and Cost.

5.1. Comparison of makespan

In the first scenario, four VMs are taken, whereas in the second scenario, eight VMs are taken and comparisons are performed for 600 to 1000 number of cloudlets. The makespan is calculated using Eq. (7). Figure 7 illustrates the makespan of algorithms having four VMs with a different range of cloudlet length. It can be analysed that with an increased set of cloudlets CM-eFCFS achieves less makespan. Comparison of akespan using eight VMs is shown in Fig. 8 and it is clearly depicted that CM-eFCFS is more efficient than other algorithms.

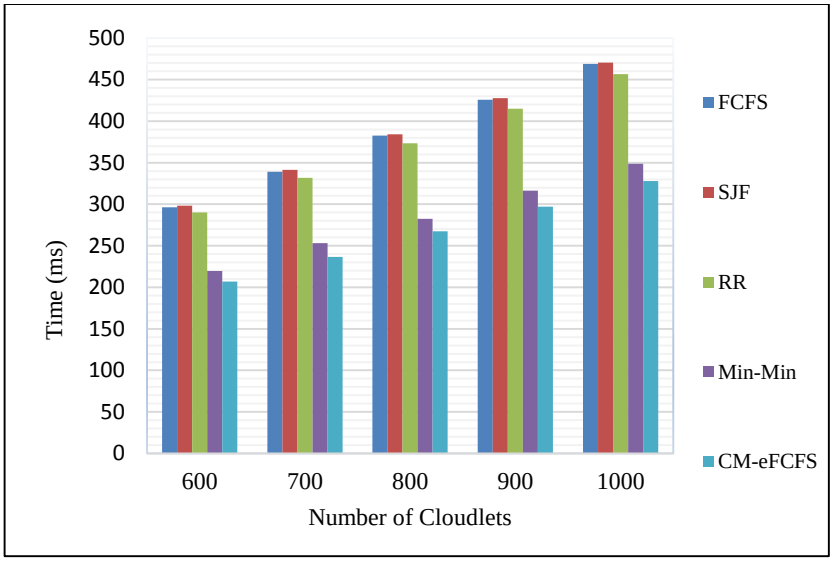

Fig. 7. Calculated makespan for four number of VMs by algorithms.

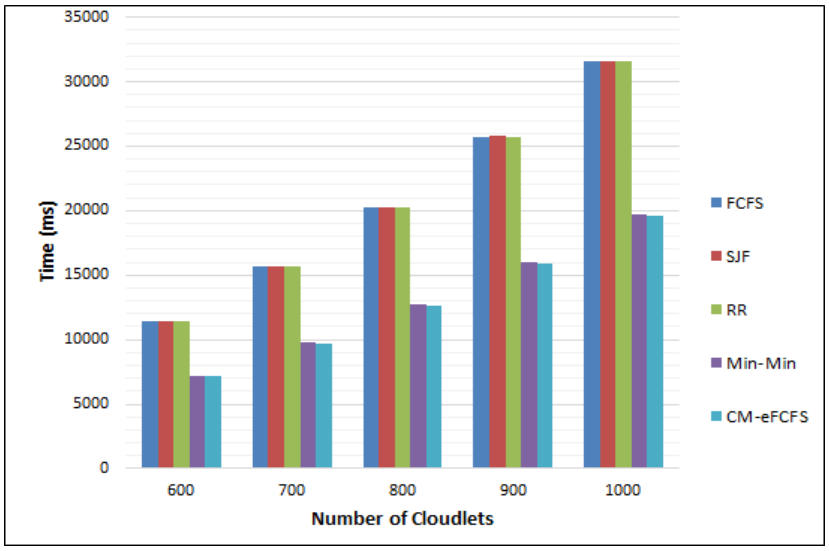

Fig. 8. Calculated makespan for eight number of VMs by algorithms.

5.2. Comparison of processing cost

Figure 9 illustrates the processing cost of algorithms with four number of VMs, which is derived from Eq. (9) where the cost of using the current resource is taken as 0.2 per ms. Figure 10 shows the processing cost for eight number of VMs. The cost of processing increases linearly with respect to an increase in the number of cloudlets. It can be observed that CM-eFCFS achieves less processing cost than the FCFS, SJF, RR and Min-Min.

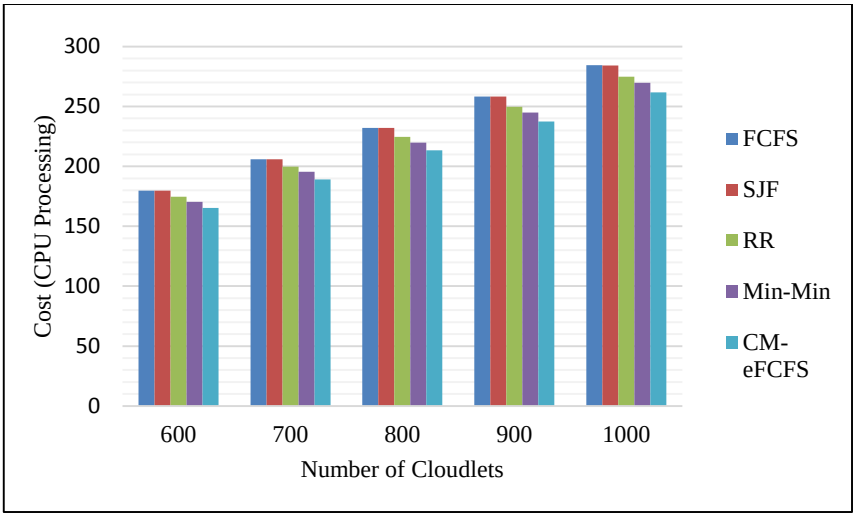

Fig. 9. Calculated processing cost for four numbers of VMs by algorithms.

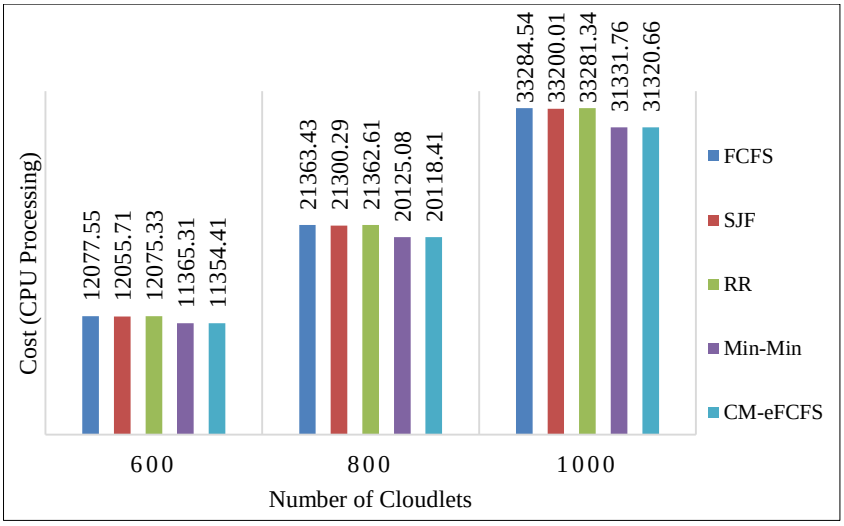

Fig. 10. Calculated processing cost for eight numbers of VMs by algorithms.

5.3. Comparison of resource utilization

The utilization of resources must be maximum so that a maximum number of users can be served in a better way. The average utilization of resources is calculated using Eq. (10).Here we compare resource utilization performance of the CM-eFCFS algorithm with Min-Min, RR, SJF and FCFS algorithm. High resource utilization reduces wastage of cloud resources. This metric also shows the efficiency of the system to utilize the allocation of cloud resources such as CPU, memory, and bandwidth. Figure 11 shows the comparative analysis of resource utilization with respect to time. The figure depicts that resource wastage in the CM-eFCFS method is low, i.e., resource utilization is high due

to efficient cloudlet assignment process. Table 3 shows the calculated percentage of the duration for which the resources are busy. The table shows that the resources in CM-eFCFS are less ideal, i.e., resource utilization is high.

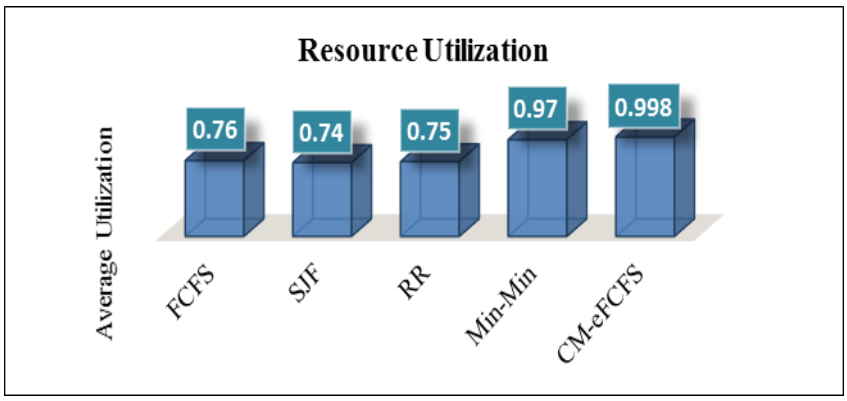

Fig. 11. Illustration of calculated resource utilization by algorithms.

Hence, these simulation results show that CM-eFCFS is better than other algorithms. Since load balancing, cloudlet assignment and cloudlet migration processes consider resources such as CPU, bandwidth and memory, it is obvious that CM-eFCFS method achieves better resource utilization in order to minimize resource wastage.

Table 3. Calculated values for resource utilization by algorithms.

	FCFS	SJF	RR	MIN-MIN	CM-eFCFS
CT_0 (ms)	287.61	280.76	275.54	331.15	327.82
CT_1 (ms)	261.84	258.29	250.03	326.13	327.93
CT_2(ms)	404.51	400.49	392.1	342.86	327.16
CT_3 (ms)	469.09	460.52	456.58	348.69	326.28
Resource utilization (%)	**75.84**	**74.38**	**75.2**	**96.7**	**99.8**

6. Conclusion

This paper proposed and formulated a new approach of cloudlet scheduling. Cloudlets are arranged and assigned to VM depending on their expected completion time. Migration is performed from one VM to another in case a better VM is available. The calculation of expected completion time and cloudlet migration gave a new algorithm, which is known as CM-eFCFS. The results have shown remarkable improvement in Execution time, Execution cost and Resource utilization over the traditional scheduling algorithms. The results make CM-eFCFS as one of the finest algorithms to be used in cloud computing. The resource utilization achieves 99.8 %, which makes the introduced approach most suited to cloud computing. In future, we wish to extend introduced work by making the approach run on a live machine built by Libvirt and Qemu and also compare CM-eFCFS with other existing approaches.

Nomenclatures

C_i	i^{th} Cloudlet
Cost	The processing costing current resource
CT_j	Completion time of VM_j
ECT_{ij}	Expected completion time of C_i on VM_j
ECT_m	Expected completion time of cloudlet after migration to fastest VM
EET_{ij}	Expected execution time of C_i on VM_j
$ERCT_i$	Earliest completion time of C_i
EVM_j	Time taken by VM_j to execute all assigned cloudlets
EVM_j	Time taken by VM_j to execute all assigned cloudlets
L_{ci}	Total length of C_i
L_{ci}	Total length of C_i
$MIPS_j$	Processing speed or capacity of PE_j (Million Instructions Per Second)
PE_j	Number of processing elements allocated to VM_j
RL_m	Remaining length to be executed of cloudlet
RT_j	Ready time of VM_j
VM_{ie}	VM that gives *ERCTi*
VM_j	j^{th} VM

Abbreviations

FCFS	First Come First Serve
SJF	Shortest Job First
RR	Round Robin
VM	Virtual Machine

References

1. Li, Q.; Hao, Q.; Xiao, L.; and Li, Z. (2009). Adaptive management of virtualized resources in cloud computing using feedback control. *Proceedings of the IEEE 1st International Conference on Information Science and Engineering*. Nanjing, China, 99-102.
2. Parikh, K.; Hawanna, N.; Haleema, P.K.; Jayasubalakshmi, R.; and Iyengar, S.N. (2015). Virtual machine allocation policy in cloud computing using cloudsim in Java. *International Journal of Grid and Distributed Computing*, 8(1), 145-158.
3. Tawfeek, M.; El-Sisi, A.; Keshk, A.; and Torkey, F. (2015). Cloud task scheduling based on ant colony optimization. *The International Arab Journal of Information Technology*, 12(2), 129-137.
4. Gao, K.; Wang, Q.; and Xi, L. (2014). Reduct algorithm based execution times prediction in knowledge discovery cloud computing environment. *The International Arab Journal of Information Technology*, 11(3), 268-275.
5. Pop, F.; Dobre, C.; Cristea, V.; and Bessis, N. (2013). Scheduling of sporadic tasks with deadline constrains in cloud environments. *Proceedings of the IEEE*

27th International Conference on Advanced Information Networking and Applications (AINA). Barcelona, Spain, 764-771.

6. Guo, L.; Zhao, S.; Shen, S.; and Jiang, C. (2012). Task scheduling optimization in cloud computing based on heuristic algorithm. *Journal of Networks*, 7(3), 547-553.
7. Yuan, H.; Bi, J.; Tan, W.; Zhou, M.; Li, B.H.; and Li, J. (2017). TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds. *IEEE Transactions on Cybernetics*, 47(11), 3658-3668.
8. Lin, W.; Liang, C.; Wang, J.Z.; and Buyya, R. (2014). Bandwidth-aware divisible task scheduling for cloud computing. *Journal of Software: Practice and Experience*, 44(2), 163-174.
9. Hong, B.; and Prasanna, V.K. (2004). Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. *Proceedings of the IEEE 18th International Symposium on Parallel and Distributed Processing*. Santa Fe, United States of America, 52-60.
10. Yu, X.; and Yu, X. (2009). A new grid computation-based min-min algorithm. *Proceedings of the IEEE 6th International Conference on Fuzzy Systems and Knowledge Discovery*. Tianjin, China, 43-45.
11. Chen, W.; Xie, G.; Li, R.; Bai, Y.; Fan, C.; and Li, K. (2017). Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems. *Future Generation Computer Systems*, 74(C), 1-11.
12. Mosleh, M.A.S.; Radhamani, G.; Hazber, M.A.G.; and Hasan, S.H. (2016). Adaptive cost-based task scheduling in cloud environment. Article ID. 8239239. *Scientific Programming*, 1-9.
13. Li, Y.; Chen, M.; Dai, W.; and Qiu, M. (2017). Energy optimization with dynamic task scheduling mobile cloud computing. *IEEE Systems Journal*, 11(1), 96-105.
14. Radulescu, A.; and van Gemund, A.J.C. (2000). Fast and effective task scheduling in heterogeneous systems. *Proceedings of the 9th Heterogeneous Computing Workshop (HCW 2000)*. Cancun, Mexico, 229-238.
15. Tsakalozos, K.; Verroios, V.; Roussopoulos, M.; and Delis, A. (2017). Live VM migration under time-constraints in share-nothing IaaS-clouds. *IEEE Transactions on Parallel and Distributed Systems*, 28(8), 2285-2298.
16. Zaa, M.; Gabhane, J.P.; and Dehankar, A.V. (2014). Task based migration using mobile cloud computing. *International Journal of Advances in Computer Science and Cloud Computing*, 2(2), 98-101.
17. Xie, R.; Wen, Y.; Jia, X.; and Xie, H. (2015). Supporting seamless virtual machine migration via named data networking in cloud data center. *IEEE Transactions on Parallel and Distributed Systems*, 26(12), 3485-3497.
18. Kansal, N.J.; and Chana, I. (2016). Energy-aware virtual machine migration for cloud computing - A firefly optimization approach. *Journal of Grid Computing*, 14(2), 327-345.
19. Panwar, N.; and Rauthan, M.S. (2017). Analysis of various task scheduling algorithms in cloud environment: Review. *Proceedings of the 7th*

International Conference on Cloud Computing, Data Science & Engineering. Noida, India, 255-261.

20. Dorigo, M.; and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344 (2-3), 243-278.
21. Santhosh, B.; and Manjaiah, D.H. (2014). An improved task scheduling algorithm based on max-min for cloud computing. *International Journal of Innovative Research in Computer and Communication Engineering*, 2(2), 84-88.
22. Elzeki, O.M.; Reshad, M.Z.; and Elsoud, M.A. (2012). Improved max-min algorithm in cloud computing. *International Journal of Computer Applications*, 50(12), 22-27.
23. Chawla, Y.; and Bhonsle, M. (2013). Dynamically optimized cost based task scheduling in cloud computing. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 2(3), 38-42.
24. Chen, H.; Wang, F.; Helian, N.; and Akanmu, G. (2013). User-priority guided min-min scheduling algorithm for load balancing in cloud computing. *Proceedings of the IEEE National Conference on Parallel Computing Technologies (PARCOMPTECH)*. Bangalore, India, 1-8.
25. Priyadarsini, R.J.; and Arockiam, L. (2014). Performance evaluation of min-min and max-min algorithms for job scheduling in federated cloud. *International Journal of Computer Applications*, 99(18), 47-54.
26. Awad, A.I.; El-Hefnawy, N.A.; and Abdel_kader, H.M. (2015). Enhanced particle swarm optimization for task scheduling in cloud computing environments. *Procedia Computer Science*, 65, 920-929.
27. Hassan Ali, H.G.E.D.; Saroit, I.A.; and Kotb, A.M. (2017). Grouped tasks scheduling algorithm based on QoS in cloud computing network. *Egyptian Informatics Journal*, 18(1), 11-19.
28. Devi, D.C.; and Uthariaraj, V.R. (2016). Load balancing in cloud computing environment using improved weighted round robin algorithm for non-preemptive dependent tasks. Article ID 3896065. *The Scientific World Journal*, 14 pages.
29. Buyya, R.; Ranjan, R.; and Calheiros, R.N. (2009). Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. *Proceedings of the International Conference on High Performance Computing and Simulation*. Leipzig, Germany, 1-11.