Arab J Sci Eng (2015) 40:1409–1425
DOI 10.1007/s13369-015-1626-9

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Development and Analysis of a New Cloudlet Allocation Strategy for QoS Improvement in Cloud

Sourav Banerjee · Mainak Adhikari · Sukhendu Kar · Utpal Biswas

Received: 15 July 2014 / Accepted: 1 March 2015 / Published online: 13 March 2015

Abstract Cloud computing has emerged as a dominant and transformational paradigm in Information technology domain over the last few years. It begins to affect a multitude of industries such as government, finance, telecommunications, and education. The Quality of Service (QoS) of a cloud service provider is an important research field which encompasses different critical issues such as efficient load balancing, response time optimization, completion time improvement, makespan improvement, and reduction in wastage of bandwidth, accountability of the overall system. This paper highlights a new cloudlet allocation policy with suitable load balancing technique that helps in distributing the cloudlets to the virtual machines (VMs) equally likely to their capacity which makes the system more active, alive, and balanced. This reduces the completion time of the cloudlet(s) as well as reduces the makespan of the VM(s) and the host(s) of a data center. Eventually, this proposed work improves the QoS. The experimental results obtained using CloudSim 3.0.3 toolkit extending few base classes are compared and analyzed with several existing allocation policies.

Keywords Cloud computing · Quality of Service (QoS) · Cloud service provider (CSP) · Virtual machine (VM) · Makespan · Data center (DC) · CloudSim 3.0.3 toolkit

S. Banerjee (✉)
Department of Computer Science and Engineering,
Kalyani Government Engineering College, Kalyani,
Nadia, India
e-mail: souravacademia@gmail.com

M. Adhikari · S. Kar
Department of Computer Science and Engineering,
IMPS College of Engineering and Technology,
Chandipur, Malda, India
e-mail: onlinemainak@yahoo.com

S. Kar
e-mail: haldia.sukh@gmail.com

U. Biswas
Department of Computer Science and Engineering,
University of Kalyani, Kalyani, Nadia, India
e-mail: utpal01in@yahoo.com

1 Introduction

Nowadays, distributed environments have evolved from shared community platform to utility-based models. The latest among these is cloud computing [1–3]. It enables the delivery of IT resources over the internet and follows an *On demand service model* where the users are charged based on their consumption. There are various types of cloud service available such as [4]—Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), Database as a Service (DaaS), and Identity as a Service (IdaaS). Moreover, they offer the flexibility, elasticity, and scalability to acquire or release resources varying configuration to the best suit of requirements of an application. This empowers the cloud users (CUs) [5] and gives the registered users more control over the resources. It also develops the innovative scheduling techniques so that the distributed resources are efficiently utilized.

Cloud computing [6,7] is a type of parallel and distributed system. It consists of collection of large-scale heterogeneous interconnected and virtualized [8,9] computers that are dynamically provisioned and presented as one or more unified computing resources established through negotiation between the service providers and customers. It really brings the concept of physical location independence to its true meaning. It provides the user with high end infrastructure even when the user is at a location where such infrastructure is impossible to setup. Although cloud computing promises to eliminate obstacles due to management of various computing resources and reduce the infrastructure cost; however, the realization of cloud computing based on mobile devices, namely mobile cloud is still in its early years [10].

Cloud computing is thus a business package where companies provide computation power, huge storage, and various other software services to users through a common interface without requiring the knowledge of location of the resources. The background activities in this domain such as virtual machine (VM) [9,11] allocation, load sharing, load balancing [12], process migration, and distributed shared memory access are completely abstracted from the user's purview. The end users or the customers can only access the cloud-based applications [3,4] as well as infrastructure through logging in to a cloud interface. The cloud service providers strive to give the same or better service and performance than the software programs which were installed locally on end-user machines. Binding or allocating cloudlet to VM in a heterogeneous cloud environment is a challenging issue. To make the environment more proficient, it requires an efficient allocation policy. There are many cloudlet allocation policies [11] in cloud computing which are available to allocate the cloudlets to the different resources or VMs in an optimal way. The cloudlet allocation policy plays a vital role to improve the overall system performance minimizing the completion time and makespan [13,14] of VMs as well as the host(s). Not only this, a proper allocation policy may lead to a good assignment of cloudlets to the suitable resources or VMs that may eventually lead to improve the Quality of Service of the overall system.

1.1 Our Contribution

In this paper, a new cloudlet allocation policy with a suitable load balancing technique has been proposed that will allocate a cloudlet from the sorted *cloudlet_list* [11] to a VM whose load capacity is maximum among all the VMs present in the *vm_list* [11]. Before allocating the next cloudlet to that VM, the remaining load capacity (RLC) of the VM will be calculated and compared with the maximum load capacity (MLC) of the other VMs in the *vm_list*. If the comparison returns greater RLC value than the MLC of other VMs, then that VM may continue with further cloudlet allocation until its RLC value becomes lesser than the MLC value of the other VMs. This process will continue until all the cloudlets in *cloudlet_list* will get assigned into the VMs. This proposed policy improves the makespan of the VMs as well as the host present in a data center (DC). Hence, the system utilization has been improved also. Various researches have been undertaken, based on scheduling techniques with various network scenarios and combinations of service classes [15]. The CloudSim 3.0.3 simulation toolkit [5,16] has been used in this work. The proposed policy has been simulated in this environment and compared with two other existing allocation policies. The first one is Round Robin Allocation (RRA) [11] Policy which has already been implemented primarily, and the second allocation policy was published in our previous paper [17] named conductance algorithm (CA). The major drawback of RRA and CA is larger makespan [14] of the VMs as well as the host present in the DC. The proposed cloudlet allocation policy provides better results than the aforementioned policies. A detail description is presented in Sect. 4.

1.2 Organization

The rest of the paper is organized as follows—Sect. 2 describes the Load Balancing in Cloud Environment. Sect. 3 describes different related works regarding allocation policies. The proposed work has been emphasized with algorithm and flow chart in Sect. 4. The experimental result of the proposed algorithm described in Sects. 5 and 6 presents the comparison result and analysis to illustrate the prominence of this proposed policy over some existing algorithms. Section 7 describes the threats to validity of the proposed work. Finally, Sect. 8 concludes and discusses future scope of the proposed work.

2 Load Balancing in Cloud Environment

The typical cloud computing environment involves a large number of geographically distributed data centers (DCs) [11] which can be interconnected and effectively utilized in order to achieve better performance not ordinarily attainable on a single DC. Each DC consists of number of hosts, and the resources of each host are accessed and utilized by one or more VMs allotted to that host according to their capacity. Each VM possesses an initial load which represents an amount of work to be performed and may have a different processing capacity.

To minimize the overall completion time of the system, the workload has to be distributed. This is why load balancing [18] is required.

The load balancing problem [18,19] is closely related to cloudlet allocation and scheduling techniques in cloud environment. It is concerned with all techniques, allowing an even distribution of workloads among the available VMs in the system. The main objective [20] of load balancing is to optimize the response time, completion time, network bandwidth, memory provisioning, makespan of the system etc.

The goals of the Load Balancing algorithms are [21] as follows:

i) To improve the performance substantially.
ii) To have a backup plan in case the system/ VM(s) failure.
iii) To maintain the system stability.
iv) To accommodate future modification in the system.
v) To improve the makespan of the system.

Types of load balancing algorithms [21,22] are:

a) Depending on the initiation of the process:
   - *Sender initiated* The load balancing algorithm is initiated by sender.
   - *Receiver initiated* The load balancing algorithm is initiated by receiver.
   - *Symmetric* It is combination of both sender initiated and receiver initiated.

b) Depending on the current state of the system:
   - *Static* It does not depend on the current state of the system. Prior knowledge of the system is required.
   - *Dynamic* Decision on load balancing is based on the current state of the system. No prior knowledge is required. So, it is better than the static approach.

   Load balancer can work in two ways [21,22]:
   - *Co-operative* The nodes work simultaneously in order to achieve the common goal of optimizing the overall response time.
   - *Non co-operative* The jobs run independently in order to improve the response time of local jobs.

2.1 Load Calculation

The load is mainly related to the number of requesting cloudlets and the bandwidth [19,23,24]. Load of the host in DC is calculated using a coefficient vector to indicate the final proportion of various impact parameters in the overall load. Parameters are defined as below [18,19]:
Load (CPU): CPU utilization
Load (MEM): Memory Usage
Load (PCR): Number of connection
Load (DIO): The size of disk occupation
Load_i = Host load of a DC occupied by application (cloudlet) is calculated according to the overall index. Then,

$$\text{Load}_i = \begin{vmatrix} r_1 & r_2 & r_3 & r_4 \end{vmatrix} \begin{vmatrix} \text{Load (CPU)} \\ \text{Load (MEM)} \\ \text{Load (PCR)} \\ \text{Load (DIO)} \end{vmatrix}$$

Coefficient S $= |r_1 \quad r_2 \quad r_3 \quad r_4|$ represents the importance of all indicator to load.

3 Related Work

3.1 CloudSim Toolkit

Several Grid simulators [25,26] such as GridSim, SimGrid, and GangSim are capable of modeling and simulating the grid application in a distributed environment but fails to support the infrastructure and application-level requirements arising from cloud computing paradigm [27]. The grid simulators are unable to isolate the multi-layer service (Software as a Service, Platform as a Service, Infrastructure as a Service) discrimination clearly as required by cloud computing paradigm. In particular, there is negligible support on existing grid simulation toolkits for modeling of virtualization-enabled resource and application management environment [28]. Cloud computing domain promises to deliver services on subscription basis in a pay-per-use model to Software as a Service providers. Therefore, cloud environment modeling and simulation toolkits must provide support for economic entities for enabling real-time trading of services between cloud users and cloud service providers. Cloudsim [8] is a cloud infrastructure modeling and simulation toolkit that supports real-time trading of services between customers and providers. An open source CloudSim framework [5] shown in Fig. 1 developed on GridSim toolkit [11] offers support for economic-driven resource management and cloudlet scheduling, bandwidth management, cost management etc. One of the prime aspects that makes cloud computing infrastructure [29] different from grid computing infrastructure is the massive deployment of virtualized infrastructure. The Cloudsim toolkit is customizable. So, it is advantageous for the researchers to implement their own policies in the domain of cloud by extending relevant classes of Cloudsim toolkit. Clouds contain an extra layer (the virtualization layer) that acts as an execution, management and hosting environment for application services which is not possible for grids. It provides users a series of extended entities and methods. In addition, it helps users to implement and analyze their own scheduling and allocation strategy at different levels including modification of module deployment techniques and conduct-related performance testing by expanding few interfaces. Present study aims at expanding CloudSim by utilizing the broker policy. The data center Broker policy is a decision making procedure through which an individual can make the best link between cloudlets and VMs. Few modules of CloudSim toolkit mentioned in Fig. 1 are relevant to our research as follows.

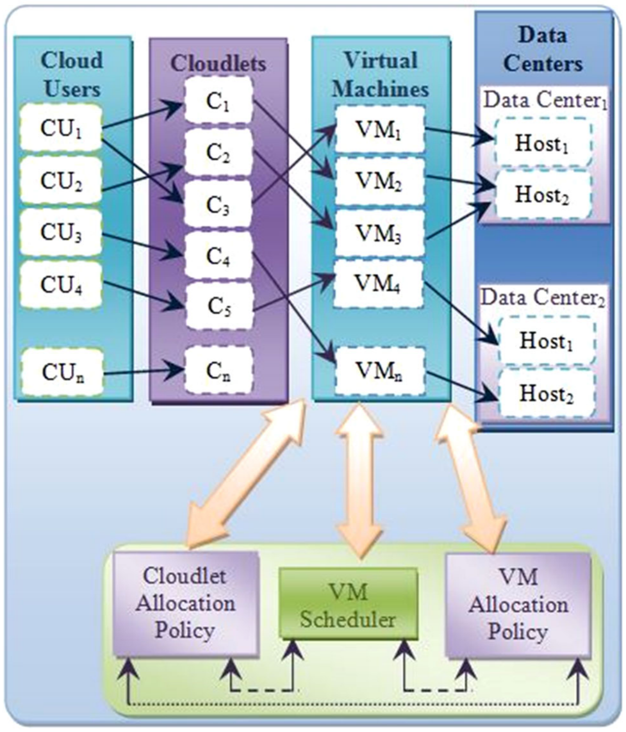

Fig. 1 Cloudsim Work Style

- *Cloud User (CU)* The CU is an end user of cloud environment. They avail the service provided by cloud data center.
- *Data center (DC)* Data center is composed of a set of hosts or physical nodes. It behaves like an IaaS provider.
- *Cloudlet* It is a bag of tasks. A cloudlet is a request sent from CU for processing [28] to the DC. It includes several properties such as cloudlet ID, cloudlet length, output file size, and number of PE (Processing Element) required. The parameters length and output file sizes of cloudlets should be greater than or equal to one [30].
- *Virtual machine (VM)* A virtual machine is a multi-user shared resource that provides heterogeneous service to all computer or network resources.
- *Host* Host is a physical node, located inside every data center which may run on various virtual machine(s). The host contains Processing Element(s) or Core(s). It can be considered as a bare metal also.

In Cloudsim 3.0.3 [23], there is a class named data center Broker which contains several policies associated with different research issues like resource management, cloudlet scheduling, cost calculation and optimization etc. The researchers may apply their own logic and policies by extending few base classes. Some of the relevant policies are described below:

- *VM Scheduler* This schedules the VM(s) to the Processing element or Core of a Host. It is required for allocating processing power to VMs [12,31]. VM scheduler is an abstract class implemented by a host component. Few policies (space-shared, time-shared) are in-built in Cloudsim 3.0.3. Researchers may apply their own policies by extending the base class.
- *VM Allocation Policy* VM Allocation Policy [12,31] is used to select available host in a data center which meets the memory, storage, and availability requirement for a VM deployment. Few policies (such as, vmallocation-simple) are in-built in Cloudsim 3.0.3. Researchers may apply their own policies by extending the base class.
- *Cloudlet Allocation Policy* Cloudlet allocation policy [8] is used to select available VM(s) in a system. It includes cloudlet scheduling mechanism which schedules the cloudlet(s) to the VM(s). Few policies [8] (Space-Shared, Time-Shared, etc) are in-built in Cloudsim 3.0.3 [23]. Researchers may apply their own policies by extending the base class.

3.2 Cloudlet Allocation Policy

The cloudlet allocation policy helps binding a cloudlet with a virtual machine (VM) and reduces the completion time of the cloudlets as well as reduces the makespan of the VMs and the host in the DC. Designing a good allocation policy is a distinguishable challenge in the cloud computing domain. The two basic existing cloudlet allocation policies are describe below:

- *Round Robin Allocation (RRA) Policy* [11] Round Robin Allocation Policy [31] allocates the cloudlet to first available VM. For example, consider there are four cloudlets (CL_1, CL_2, CL_3, CL_4) and two VMs (VM_1, VM_2) present in the system. Table 1 emphasizes the allocation fashion. According to this policy, cloudlet

Table 1 Cloudlet binding with VM

Cloudlet	Virtual machine (VM)
CL_1	VM_1
CL_2	VM_2
CL_3	VM_1
CL_4	VM_2

CL_1 allocated to VM_1, CL_2 allocated to VM_2 and CL_3 and CL_4 allocated to the VM_1 and VM_2, respectively.

- *Conductance algorithm (CA)* [17] It considers each VM as a pipe. It calculates the Conductance (processing power) as per Eq. (1) of each VM as the ratio of its capacity to the sum of the capacity of all the VMs present in a System.

$$\text{Conductance}_j = \text{MIPS}_j / \sum_{j=1}^{n} \text{MIPS}_j \qquad (1)$$

After the calculation of conductance, multiply the conductance of that particular VM with the length of the cloudlet list. To determine the strip length, Eq. (2) is used. It determines the number of cloudlets the VM can process.

$$\text{Striplength}_j = \text{Conductance}_j \times (\text{Length of the cloudlet list}) \qquad (2)$$

The existing policies do not consider the procedure for finding out the minimum makespan of the VM(s) as well as the host(s) in DC. To overcome this problem, we propose a new cloudlet allocation policy which improves the makespan of the VMs as well as the host which will be discussed in Sect. 4.

4 Proposed Work

This work highlights a new cloudlet allocation strategy with a suitable load balancing technique that will eventually improve the completion time of the cloudlets as well as the makespan [13,32] of the VMs and the host(s) present in the data center (DC) [38]. The prime objective is to improve the QoS of a cloud in association with two basic parameters i.e., completion time and makespan.

4.1 Cloud Allocation Policy

In the proposed cloudlet allocation policy, all the VMs are sorted in descending order according to their MIPS (Million Instruction per Second) and stored in *vm_list* [11] array which is shown in Table 4. Then, calculate the maximum load capacity (MLC) of each VM by using the Eq. (3).

$$\text{MLC}_j\ (\%) = \left(\text{MIPS}_j/\text{Total MIPS of all VMs}\right) \times 100 \qquad (3)$$

The batches of cloudlets are stored in *cloudlet_list* [11] array. Every cloudlet has its own cloudlet ID. The size of each cloudlet (cl_size) is calculated in term of percentage using Eq. (4).

$$\text{cl\_size}_i\ (\%) = (\text{MI}_i / \text{Total MI of all cloudlets}) \times 100 \qquad (4)$$

The cloudlets are sorted into descending order according to their Million Instruction (MI) value and placed them in *cloudlet_list*. This is shown in Table 7. Then, the cloudlets are assigned to the VMs guided by the First Come First Serve (FCFS) policy following the process in which the highest capacity VM will be allocated first, and after allocation, the remaining load capacity (RLC) of that VM will be compared with the MLC of the other VMs in the *vm_list*. If the comparison returns greater RLC value than the MLC value of other VMs, then that VM may continue with further cloudlet allocation until RLC value becomes lesser than the MLC value of the other VMs. This process will continue until all the cloudlets in *cloudlet_list* will get assigned into the VMs.

$$\text{RLC}_j(\%) = \text{MLC}_j - \text{cl\_size}_i \qquad (5)$$

The proposed work is explained with a suitable example in Sect. 5.

This work is completely simulated in Cloudsim 3.0.3 extending and modifying few required base classes.

4.2 Parameters of Proposed Algorithm

(*1.*) *Execution time (Exec)* Execution time is defined as the ratio of MI (Million Instruction) of the cloudlet to the MIPS (Million Instruction Per Second) of the allotted VM. The calculation of the execution time of a cloudlet is mentioned in the Eq. (6).

$$\text{Exec}_i = \text{MI of Cloudlet}_i / \text{MIPS of VM}_j \qquad (6)$$

(*2.*) *Starting time (ST)* Starting time is defined as the completion time of the previous cloudlets in the same VM. Whenever a cloudlet is assigned to a VM for the first time then its starting time is assumed to be zero. When the other cloudlets are assigned to that VM the completion time of the previous cloudlet on that VM will be treated as the starting time of next assigned cloudlet. This is known as the updated starting time for the assigned cloudlet. The calculation of the starting time of a cloudlet is mentioned in the Eq. (7).

$$\begin{aligned}&\text{Start}_i = 0;\ \text{if}\ (\text{VM}_i \text{is in no\_load condition})\\&\text{Start}_i > 0;\ \text{Completion time of the present}\\&\qquad\text{Cloudlet will be treated as start time of}\\&\qquad\text{next Cloudlet;}\ \text{if}\ (\text{VM}_i \text{is executing})\end{aligned}\tag{7}$$

3. *Completion time (CT)* It is defined as the sum of the execution time and the completion time of the previous cloudlet allocated in the same VM. The calculation of the completion time of a cloudlet is mentioned in the Eq. (8).

$$\begin{aligned}\text{Completion time}\ (\text{CT}_i)\ &= \text{Execution time}\ (\text{Exec}_i) +\\&\text{Completion time of the previous allotted cloudlet in}\\&\text{the same VM}\ (\text{CT}_{i-1})\end{aligned}\tag{8}$$

4. *Makespan (MS)* In this paper the makespan is used to measure the efficiency of both the VM (MSVM) as well as host (MSH) in a DC.

4.2.1 Makespan of VM (MSVM)

The makespan [32] is the time to complete a batch of cloudlets in a VM. For instance, consider 'm' virtual machines for scheduling, indexed by the set $M = \{1, \ldots, m\}$. There are furthermore given n cloudlets, indexed by the set $J = \{1, \ldots, n\}$, where cloudlet j takes $p_{i,j}$ units of time if scheduled on virtual machine i. Let J_i be the set of cloudlets scheduled on virtual machine i. Then, $l_i = \sum_{j \epsilon J} \text{P}_{i,j}$ is the load of virtual machine i. The maximum load $l_{\max} = \text{c}_{\max} = \max_{i \epsilon M} l_i$ is called the makespan of the schedule. Here, the makespan of a VM is calculated as the completion time (CT_j) of the last assigned cld_id$_i$ to the corresponding VM$_j$. The calculation of the makespan of a VM is mentioned in the Eq. (9).

$$\begin{aligned}\text{MSVM}_j\ &= \text{Completion time of last Allotted cloudlet into}\\&\quad\ \text{VM}_j\end{aligned}\tag{9}$$

4.2.2 Makespan of Host (MSH)

The makespan of a host is calculated as the maximum makespan of the VM from the allotted VMs in that host. The calculation of the makespan of a VM is mentioned in the Eq. (10).

$$\text{MSH}_j = \max\left(\text{MSVM}_j\right)\tag{10}$$

4.3 Algorithm of Proposed Work

Input: 'n' be the number of VMs present in the *vm_list.*

'm' be the number of cloudlets present in the *cloudlet_list.*

Output: Allocate the 'm' number of cloudlets to 'n' number of VMs following proper load balancing method.

```
1) Initially allocate the VMs into the Host in the DC and sort them into ascending order according to their MIPS value. Store the sorted VMs in vm_list.
2) Calculate Maximum Load Capacity (MLC) of each VM present in vm_list.
3) The batch of cloudlets is stored in cloudlet_list.
4) Calculate the size of each cloudlet (cl_size).
5) Sort the cloudlets in descending order according to the size of cloudlets.
6) Allocate the cloudlets to the VMs according to the following steps-
    For (i = 0; i < m; i++)
        For (j = 0; j < n; j++)
6.1) Find the maximum MLC valued VM from vm_list.
6.2) Allocate the cloudlet to the corresponding VM which has maximum MLC value.
6.3) Decreasing MLC value of the VM by the size of the assigned cloudlet (cl_size) into the particular VM and assigned the decreased MLC value to RLC.
6.4) Update the MLC value of allotted VM using the value of RLC of that VM.
6.5) Continue the process until all the cloudlets are assigned to the VMs.
        End for
    End for
```

```
7) Calculate the Starting Time, Execution Time and Completion Time for each of the cloudlet according to the following steps-
    For ($i = 0$; $i < m$; $i$++)
7.1)    Execution Time = MI of cloudlet / MIPS of the allotted VM;
        For ($j$ = o; $j < n$; $j$++)
7.2)        If VM is in no load condition, then-
                Starting Time = 0;
                Completion Time = Execution time of allotted cloudlet;
            End if.
7.3)        else
                Starting Time = Completion Time of the previously allotted cloudlet to the VM;
                Completion Time = Execution time of allotted cloudlet + Completion Time of the previously allotted cloudlet to the VM;
            End else
        End for
    End for
8. Find the makespan of the VMs present in the vm_list.
9. Find the makespan of the Host present in the DC.
10. End
```

4.4 Pseudo Code of Proposed Algorithm

```
1) Initially allocate the VMs into the host in the Data Centre and sort them in ascending order according to their MIPS value and store them in vm_list.
2) Calculate the Maximum Load Capacity (MLC) of each VM.
3) The batch of requested cloudlets is stored in cloudlet_list.
4) Calculate the size of each cloudlet (cl_size).
5) Sort the cloudlets in descending order according to their size (cl_size).
6) Place the cloudlets to the stored VMs according to the following steps:-
6.1) For m number of cloudlet and n number of VMs-
6.2) for $i$:= 0 to $m$
     {
       for $j$ : = 0 to $n$
     {
6.3)   /* Find the Maximum Load Capacity (MLC) VM from the VMs present in the vm_list. */
       $MLC_j$ = ($MIPS_j$ / total MIPS of all VMs) * 100%;
         max ( $MLC_j$); /* Find the maximum MLC valued VM.*/
     }
6.4) /* Allocate the cloudlet to the corresponding $VM_j$ which has maximum MLC value. */
         $VM_j$<- $CU_i$;
6.5) /* Decreased the MLC value of the $VM_j$ by the size of the assigned cloudlet ($cl\_size_i$) into the particular $VM_j$ */
         $RLC_j$ = $MLC_j$ - $cl\_size_i$; // Update the RLC value
```

```
6.6) /* Update the MLC value of the VM according to the value of RLC */
            $MLC_j = RLC_j$; // Update the MLC Value
        }
6.7) Return the step 6.1 until all the cloudlets present in cloudlet_list are allocated to their corresponding VM.
    }
7) Calculate Starting Time, Execution Time and Completion Time for each cloudlet according to the following step-
7.1) for $i$:= 0 to $m$
    {
        for $j$ := 0 to $n$
        {
            if ($VM_i$ is in no_load condition)
            {
7.1)            $ST_{ij} = 0$; /* if the cloudlet is firstly allocated to the $VM_j$, then the Starting Time of the cloudlet is equal to zero */
7.2)            $Exec_{ij} = MI_i / MIPS_j$; /* Calculate the Execution time of each cloudlet */
7.3)            $CT_{ij} = Exec_{ij}$; /* if the cloudlet is firstly allocated to the $VM_j$, then the completion time of the cloudlet is same as the Execution Time of that cloudlet*/
            }
7.4)        else
            {
7.5)            $ST_{ij} = CT_{i-1j}$; /* Starting Time of a cloudlet is the same as the completion time of the previously allocated cloudlet in the same $VM_j$ */
7.6)            $Exec_{ij} = MI_i / MIPS_j$;
7.7)            $CT_{ij} = Exec_{ij} + CT_{i-1j}$ ; /* Completion Time of a cloudlet is the same as the summation of the Execution Time of the present allotted cloudlet to the VM and the Completion Time of the previous cloudlet assigned into the same VM. */
            }
        }
7.8) Return step 7.1 until completing the calculation of the Completion Time of the cloudlets to the allotted VM.
    }
8) Find the makespan of n number of VMs
    for $j$ := 0 to $n$
    {
        $MSVM_j$ = Completion time of last allocated cloudlet in the $j$th VM; // makespan of the VMs
    }
9) Find the makespan of the Host with n number of VMs
    for $j$ := 0 to $n$
    {
        $MSH_j = \max(MSVM_j)$; // makespan of the Host
    }
10) Exit.
```

4.5 Flow Chart of Proposed Algorithm

Figure 2 shows the flow chart of the proposed cloudlet allocation policy.

4.6 Advantages of Proposed Algorithm

(a) The proposed algorithm has improved the makespan of the VM(s) as well as the host(s) in DC.
(b) The algorithm distributes the loads to each VM almost equally according to their capacity and here, the makespan of each VM is almost about same.

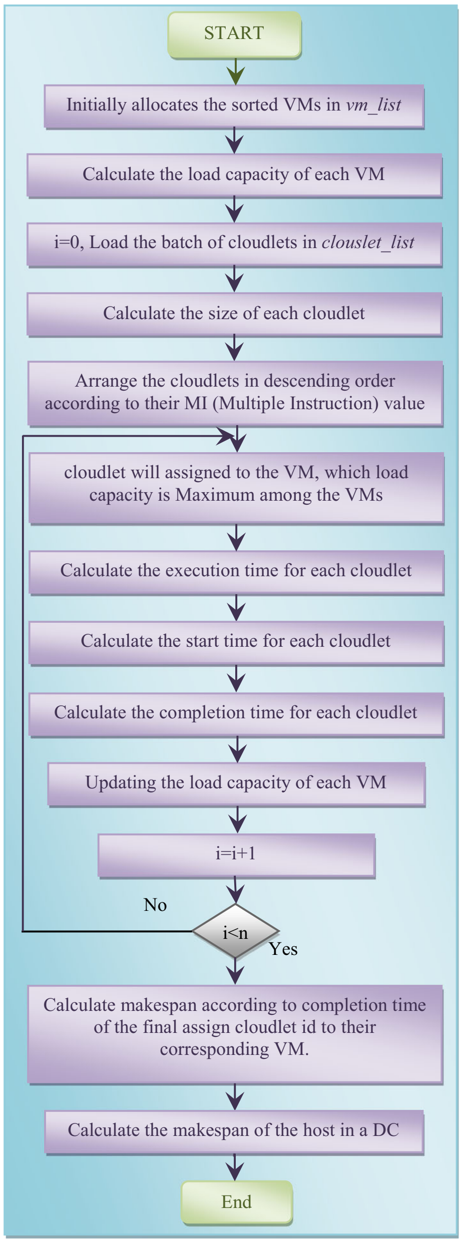
START
Initially allocates the sorted VMs in vm_list
Calculate the load capacity of each VM
i=0, Load the batch of cloudlets in closlet_list
Calculate the size of each cloudlet
Arrange the cloudlets in descending order according to their MI (Multiple Instruction) value
cloudlet will assigned to the VM, which load capacity is Maximum among the VMs
Calculate the execution time for each cloudlet
Calculate the start time for each cloudlet
Calculate the completion time for each cloudlet
Updating the load capacity of each VM
i=i+1
No
i<n
Yes
Calculate makespan according to completion time of the final assign cloudlet id to their corresponding VM.
Calculate the makespan of the host in a DC
End

Fig. 2 Flow chart of proposed algorithm

Table 2 Properties of cloudlets

cld_id	MI (million of instruction)
0	9000
1	16,000
2	11,000
3	6000
4	15,000
5	8000
6	12,000
7	17,000
8	10,000
9	7000

Table 3 Processing capability of VMs

vm_id	MIPS (million of instruction per second)
0	300
1	600
2	400

Table 4 Properties of arranged VMs

vm_id	MIPS (million of instruction per second)
1	600
2	400
0	300

Table 5 Maximum load capacity of each VM

vm_id	MIPS	Capacity of VM [(MIPS/total MIPS of all VMs) $\times$ 100] (%)
1	600	46
2	400	31
0	300	23

5 Experimental Result

The proposed algorithm has been described and analyzed with the help of a suitable example. Due to space constraint, ten cloudlets and three VMs in a host have been considered in these experiments.

Table 2 represents ten cloudlets with their sizes (MI) mentioned.

Table 3 shows three VMs with their MIPS that represents the processing capability of each VM.

Table 4 shows the VMs are sorted and stored in decreasing order according to their MIPS.

Table 5 represents the maximum load capacity of each VM, calculated in percentage.

Table 6 shows the size of each cloudlet, calculated in percentage.

Table 7 shows the size of each sorted cloudlet, calculated in percentage.

The cloudlets are allocated into the VMs in FCFS manner and follow this proposed policy. Each of the cloudlet will be assigned to the VM whose MLC is maximum. After allocation, the remaining load capacity (RLC) will be calculated using Eq. (3), and the RLC will be compared with MLC of the other VMs in the *vm_list* before assigning next cloudlet to the VM from *vm_list*. If the comparison returns greater RLC

Table 6 Size (%) of each cloudlet

cld_id	MI	Capacity of cloudlet [(MI/total MI of all VMs) × 100] (%)
0	9000	8
1	16,000	14
2	11,000	10
3	6000	5
4	15,000	14
5	8000	7
6	12,000	11
7	17,000	15
8	10,000	9
9	7000	6

Table 7 Size (%) of each sorted cloudlet

cld_id	MI	Capacity of cloudlet [(MI/total MI of all VMs) × 100] (%)
7	17,000	15
1	16,000	14
4	15,000	14
6	12,000	11
2	11,000	10
8	10,000	9
0	9000	8
5	8000	7
9	7000	6
3	6000	5

Table 8 Cloudlet allocated to VM

cld_ id	cl_size (%)	vm_id	MLC (%)	RLC (%)
7	15	0	23	8
7	15	**1**	46	31
7	15	2	31	16
1	14	0	23	9
1	14	**1**	31	17
1	14	2	31	17
4	14	0	23	9
4	14	1	17	3
4	14	**2**	31	17
6	11	**0**	23	12
6	11	1	17	6
6	11	2	17	6
2	10	0	12	2
2	10	**1**	17	7
2	10	2	17	7
8	9	0	12	3
8	9	1	7	−2
8	9	**2**	17	8
0	8	**0**	12	4
0	8	1	7	−1
0	8	2	8	0
5	7	0	4	−3
5	7	1	7	0
5	7	**2**	8	1
9	6	0	4	−2
9	6	**1**	7	1
9	6	2	1	−5
3	5	**0**	4	−1
3	5	1	1	−4
3	5	2	1	−4

Bold values represent the cloudlet allocated to that VM

Table 9 Final cloudlet allocation to VM

cld_id	cl_size (%)	vm_id	MLC (%)	RLC (%)
7	15	1	46	31
1	14	1	31	17
4	14	2	31	17
6	11	0	23	12
2	10	1	17	7
8	9	2	17	8
0	8	0	12	4
5	7	2	8	1
9	6	1	7	1
3	5	0	4	−1

value than the MLC value of other VMs that VM may be allowed to continue with further cloudlet allocation till RLC of that VM gets lesser value than the MLC of the other VMs. This process continues till all the cloudlets in *cloudlet_list* are assigned to the VMs which is illustrated in Table 8.

Table 9 represents the list of finally allotted cloudlets to their respective VMs.

Table 10 shows the allotted cloudlets to their respective VMs with their execution time, start time, and completion time.

Table 11 depicts the makespan of three VMs of a host.

Table 12 represents the makespan of the host present in the DC.

6 Comparison Result and Analysis

This section deals with analyzing the improvement of the QoS in association with two parameters, such as completion time and makespan. The performance of the proposed algorithm is compared and analyzed by two existing algorithms such as RRA [17] and CA [20]. The simulated results

Table 10 Execution time, completion time and start time of all cloudlet allocated into their corresponding VMs

cld_id	MI	vm_id	MIPS	Exec	ST	CT
0	9000	0	300	30	0	30
1	16,000	1	600	27	0	27
2	11,000	1	600	18	27	45
3	6000	0	300	20	30	50
4	15,000	2	400	38	0	38
5	8000	2	400	20	38	58
6	12,000	0	300	40	50	90
7	17,000	1	600	28	45	73
8	10,000	2	400	25	58	83
9	7000	1	600	12	73	85

Table 11 Makespan of VMs

vm_id	MSVM
0	90
1	85
2	83

Table 12 Makespan of host

host_id	MSH
0	90

are evaluated and analyzed in several aspects. The performance was measured by setting up different simulation environments with varying number of cloudlets and VMs. The improvement of completion time and makespan is explained with interpretation of the graphs in tabular form. Due to space constraint, only three sets of cloudlets and VMs are taken into account, and few cloudlets are mentioned in the interpretation table. The improvement in the result of the proposed work indicates the enhancement of the QoS of a cloud. Few abbreviations are mentioned in this section that will be used in rest of this paper. Those are *fig: Figure, algo: Algorithm, cld_id: cloudlet ID, vm_id: Virtual Machine ID, host_id:Host ID, Avg_CT: Average Completion Time, Roi_CT: Rate of Improvement in Completion Time*.

6.1 Completion Time

In Figs. 3 and 4, the improvement of completion time is presented when the system is loaded with 25 cloudlets and 2 VMs. The interpretations of these two figures are presented in Tables 13 and 14 with few cloudlets. The rate of improvement in completion time is 29.95 % using the proposed over RRA and 24.8 % using the proposed over CA.

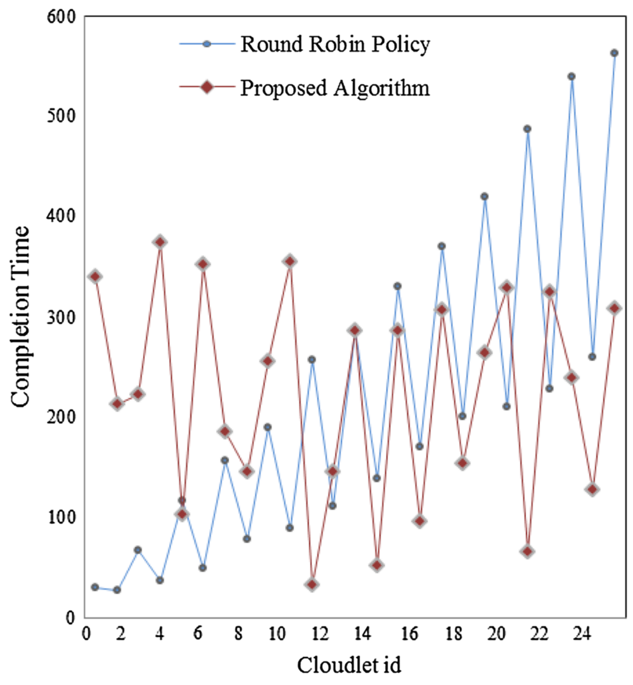

Fig. 3 Comparison graph of completion time for 25 cloudlets among Round Robin Policy versus proposed algorithm

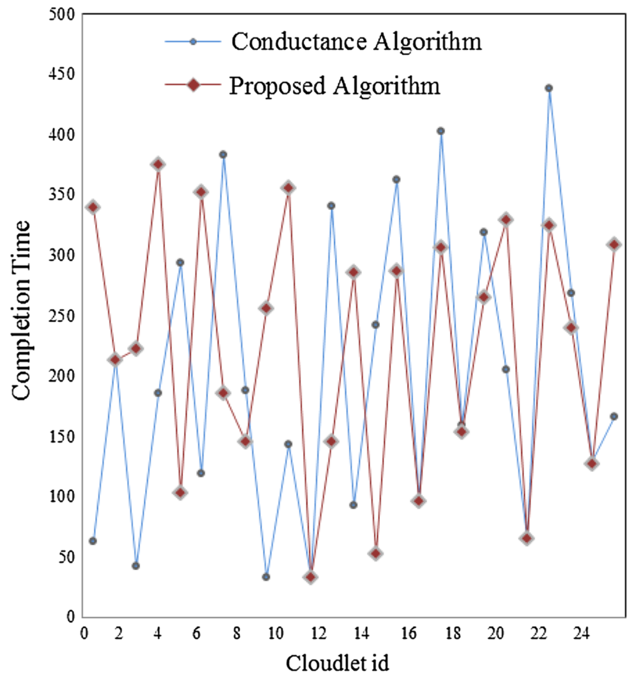

Fig. 4 Comparison graph of completion time for 25 cloudlets among Conductance algorithm versus proposed algorithm

The reflection of improvement of completion time with 50 cloudlets and 3 VMs is represented in Figs. 5 and 6 with RRA versus proposed and CA versus proposed, respectively. In Tables 15 and 16, the interpretation of the graphs are mentioned with few cloudlets. The rate of improvement of the

Table 13 Rate of improvement of proposed algorithm over RRA

Figure	algo	cld_id	CT	Avg_CT		Roi_ CT (%)
				RRA	Proposed	
3	RRA	4	117	214.75	165.25	29.95
	Proposed		103			
	RRA	15	171			
	Proposed		97			
	RRA	16	370			
	Proposed		307			
	RRA	17	201			
	Proposed		154			

Table 14 Rate of improvement of proposed algorithm over CA

Figure	algo	cld_id	CT	Avg_CT		Roi_CT (%)
				RRA	Proposed	
4	CA	7	188	305.75	245	24.8
	Proposed		270			
	CA	50	470			
	Proposed		267			
	CA	68	826			
	Proposed		632			
	CA	40	497			
	Proposed		364			

completion time is 14.85 % using the proposed over RRA and 29.25 % using the proposed over CA.

The improvement of completion time using 100 cloudlets and 4 VMs is also mentioned in the same way in Figs. 7 and 8. They are interpreted in Tables 17 and 18.

6.2 Makespan

In Figs. 9 and 10 the improvement of makespan for the VMs and the host is reflected and interpreted in Tables 19 and 20, respectively, when the system is loaded with 25 cloudlets and 2 VMs. The rate of improvement in makespan for VM_0 is 71 % and VM_1 is 17 %.

The rate of improvement of makespan for the host using the proposed algorithm over RRA Policy is 62 %. The rate of improvement in makespan for the host using the proposed algorithm over CA is 17 %.

The Figs. 11 and 12 represent the improvement in makespan for the VMs and the host when the system is loaded with 50 cloudlets and 3 VMs. The figures are interpreted

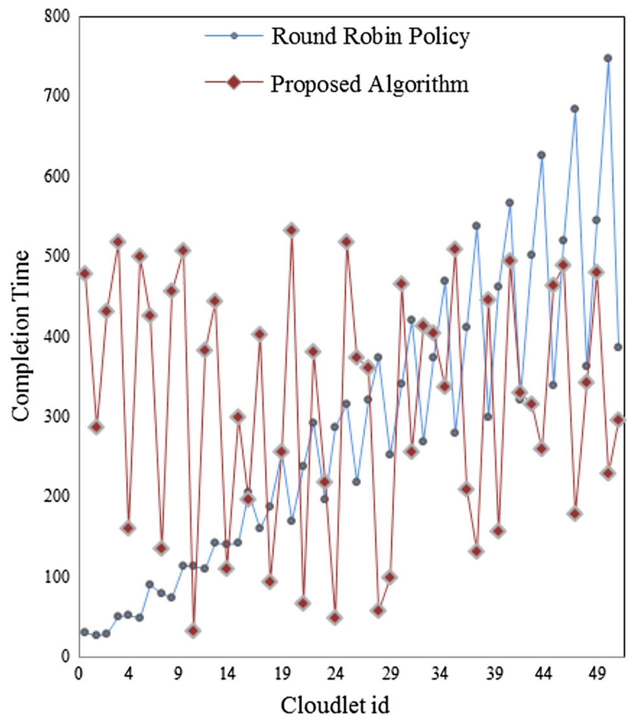

Fig. 5 Comparison graph of completion time for 50 cloudlets among Round Robin Policy versus proposed algorithm

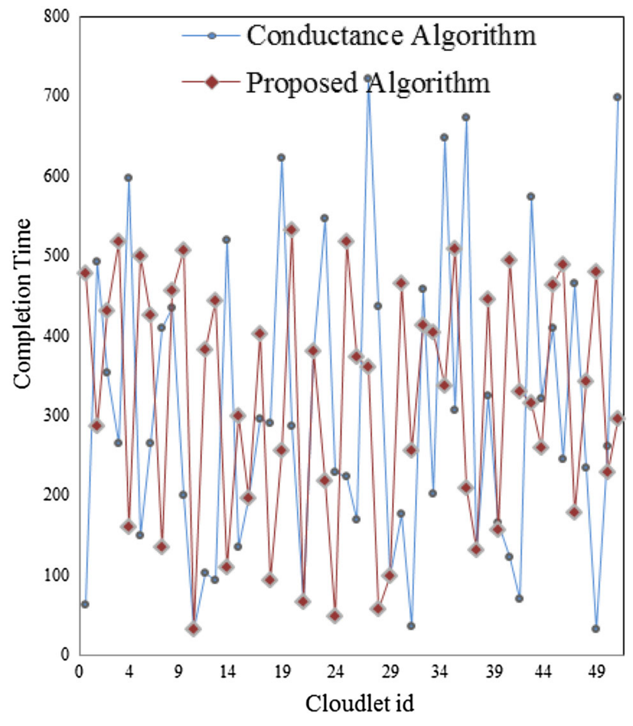

Fig. 6 Comparison graph of completion time for 50 cloudlets among Conductance algorithm versus proposed algorithm

Table 15 Rate of improvement of proposed algorithm over RRA

Figure	algo	cld_id	CT	Avg_CT		Roi_CT(%)
				RRA	Proposed	
7	RRA	13	140	313.25	272.75	14.85
	Proposed		110			
	RRA	15	206			
	Proposed		197			
	RRA	44	520			
	Proposed		489			
	RRA	49	387			
	Proposed		295			

Table 16 Rate of improvement of proposed algorithm over CA

Figure	algo	cld_id	CT	Avg_CT		Roi_ CT (%)
				RRA	Proposed	
8	CA	31	459	383.25	296.75	29.25
	Proposed		413			
	CA	42	321			
	Proposed		259			
	CA	48	261			
	Proposed		229			

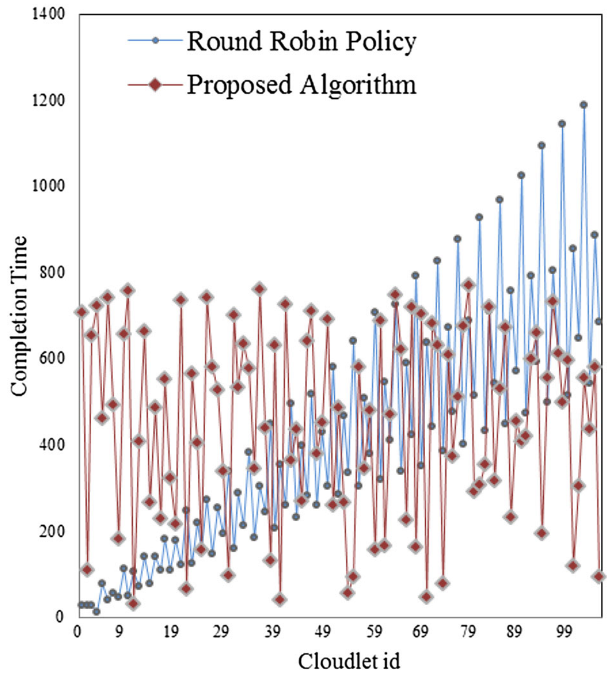

Fig. 7 Comparison graph of completion time for 100 cloudlets among Round Robin Policy versus proposed algorithm

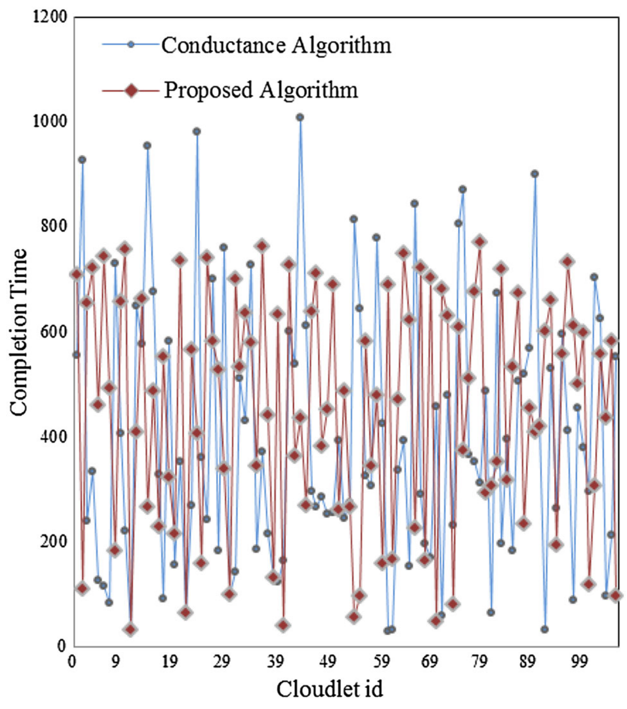

Fig. 8 Comparison graph of completion time for 100 cloudlets among Conductance algorithm versus proposed algorithm

Table 17 Rate of improvement of proposed algorithm over RRA

Figure	algo	cld_id	CT	Avg_CT		Roi_CT (%)
				RRA	Proposed	
11	RRA	42	399	548	383.25	43
	Proposed		270			
	RRA	50	470			
	Proposed		267			
	RRA	68	826			
	Proposed		632			
	RRA	40	497			
	Proposed		364			

Table 18 Rate of improvement of proposed algorithm over CA

Figure	algo	cld_id	CT	Avg_CT		Roi_CT (%)
				RRA	Proposed	
12	CA	11	650	559.5	362.75	54.24
	Proposed		410			
	CA	14	676			
	Proposed		487			
	CA	15	329			
	Proposed		230			
	CA	17	583			
	Proposed		324			

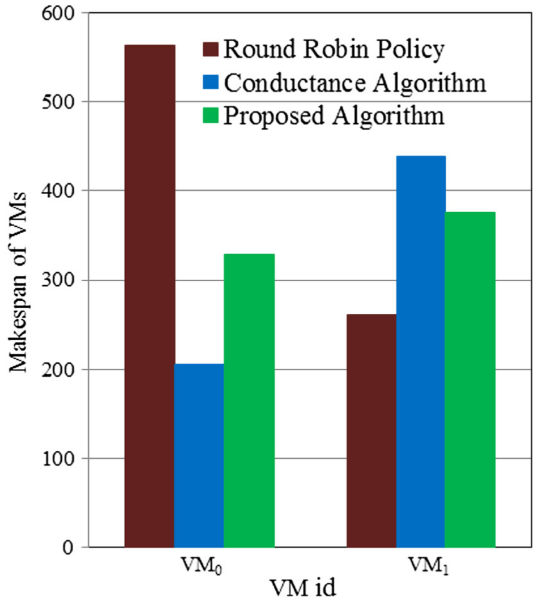

Fig. 9 Comparison graph of makespan of VMs among Round Robin Policy versus Conductance algorithm versus proposed algorithm

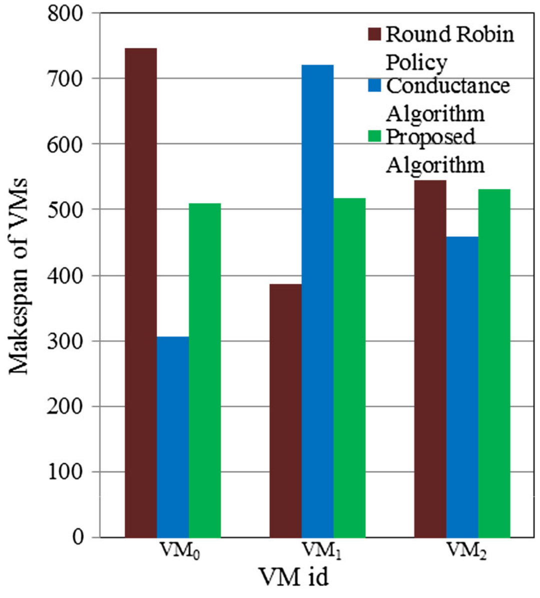

Fig. 11 Comparison graph of makespan of VMs among Round Robin Policy versus Conductance algorithm versus proposed algorithm

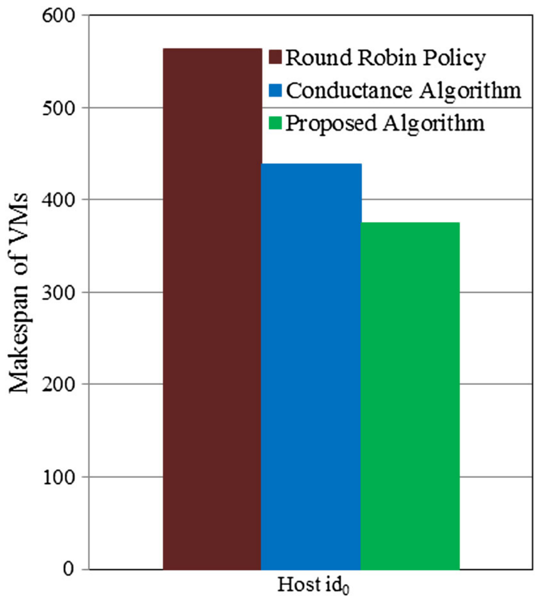

Fig. 10 Makespan of a host in a data center of 25 cloudlets

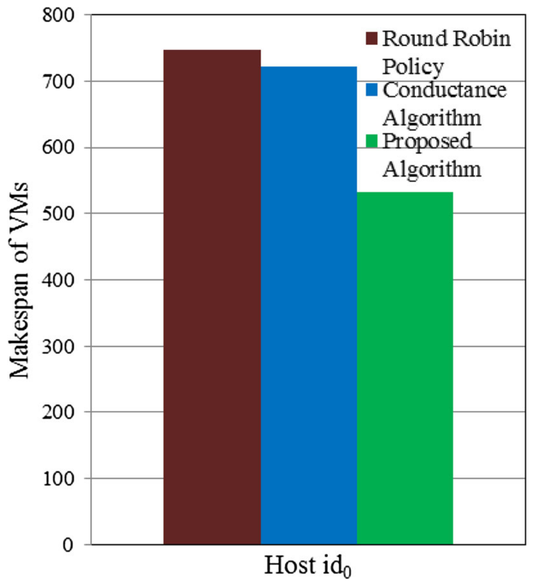

Fig. 12 Makespan of a host of a data center of 50 cloudlets

Table 19 Makespan of VMs

vm_id	Round Robin	Conductance algorithm	Proposed algorithm
0	563	206	329
1	261	439	375

Table 20 Makespan of the host of a data center

host_id	Round Robin	Conductance algorithm	Proposed algorithm
0	563	439	375

in Tables 21 and 22, respectively. The rate of improvement in makespan for VM$_0$, VM$_1$ and VM$_2$ are 47, 39 and 2 %, respectively.

The rate of improvement of makespan for the host using proposed algorithm over RRA policy is 40 %. The rate of improvement of makespan for the host using the proposed algorithm over CA is 36 %.

In the same way, the improvement of makespan of VMs and the host is represented in Figs. 13 and 14 when the system is loaded with 100 cloudlets and 4 VMs. The graphs are interpreted in Tables 23 and 24, respectively. The rates of

Table 21 Makespan of VMs

vm_id	Round Robin	Conductance	Proposed algorithm
0	747	306	509
1	387	721	517
2	545	459	532

Table 22 Makespan of a host

host_id	Round Robin	Conductance	Proposed algorithm
0	747	721	532

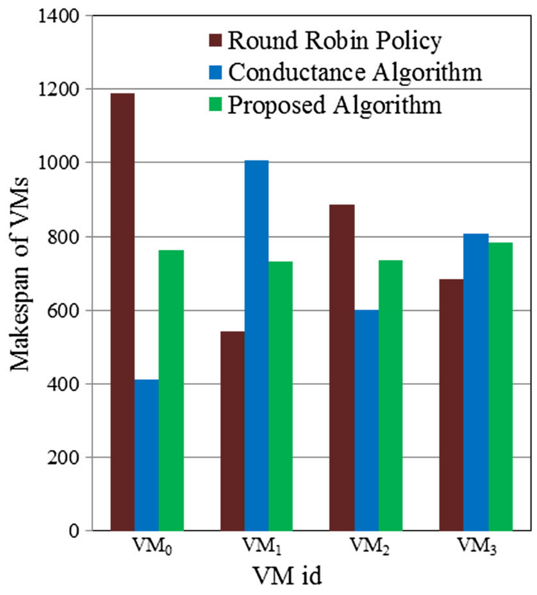

Fig. 13 Comparison graph of makespan of VMs present in the host among Round Robin Policy versus Conductance algorithm versus proposed algorithm

improvements for the VMs are 59 % for VM$_0$, 33 % for VM$_1$, 19 % for VM$_2$, 3 % for VM$_3$.

The rate of improvement of makespan for the host using the proposed algorithm in comparison with RRA is 52 and 28 % with CA.

From Fig. 15 it is clear that as the number of cloudlets increase in the host in a DC, the proposed algorithm gives better result. It is a nature of realistic approach.

7 Threats to Validity

This is a great challenge for the researchers to defend the threats to validity [33–35] in the area of cloud computing. The validity is determined in different ways; internal, external, structural, etc. We justify the validity of the results by discussing the possible threats to our result and the countermeasures that we have taken to minimize them.

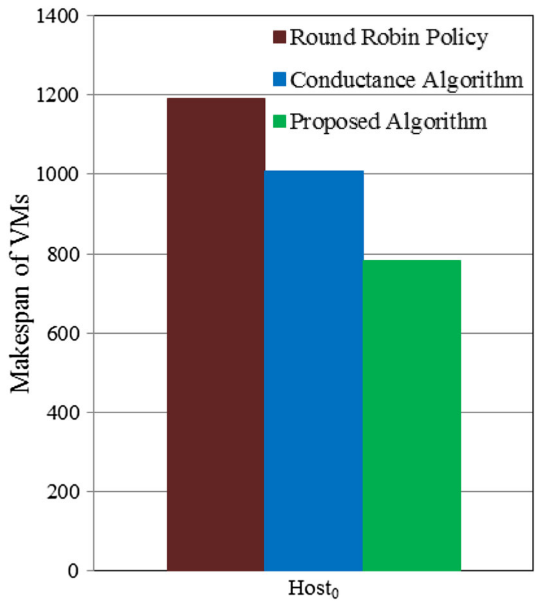

Fig. 14 Makespan of the host of a data center of 100 cloudlets

Table 23 Makespan of VMs

vm_id	Round Robin	Conductance	Proposed algorithm
0	1190	413	750
1	544	1007	760
2	886	600	747
3	685	806	784

Table 24 Makespan of host

host_id	Round Robin	Conductance	Proposed algorithm
0	1190	1007	784

7.1 Internal Validity

It is the extent to which an outcome of a study can be interpreted accurately. It refers to the extent to which the design and execution of the policy are likely to prevent methodical fallacy. The analysis of this work is a subjective measurement which may affect the result in various perspectives. This study involves of four researchers in the identification and evaluation process that minimizes the threats to internal validity. More than one batch of data sets are considered in this work to mitigate the internal validity. It provides a realistic impact of this study. This step also reduces the possibility of bias. During a script development phase, many errors may occur. So, each and every step is validated with the help of simulation environment and the output is verified. The proce-

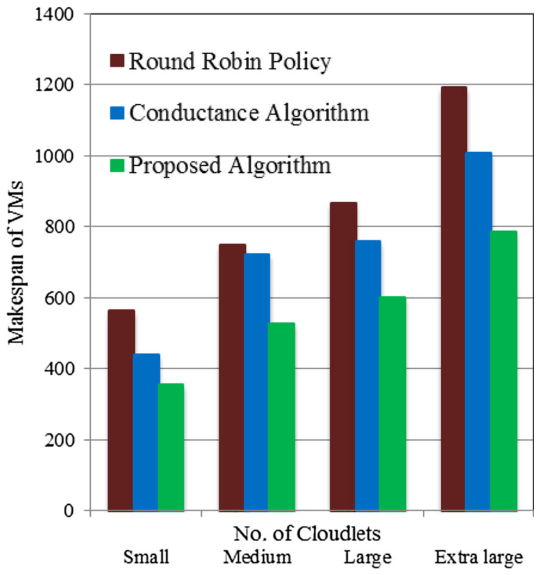

Fig. 15 Makespan of the host of a data center in different number of cloudlets

dure followed, is discussed with our guide. We have repeated the experiment more than one time, and the average has been calculated on each of the batches.

7.2 External Validity

It refers to the degree to which the results of a factual analysis can be generalized to and across several aspects. The scope of this work is purely based on the academic domain and peer-reviewed research papers. We are aware of the fact that QoS improvement approaches originate in industry and may not have been reported upon academically. Due to feasibility issues and to maintain the quality of the research, such industry-based QoS improvement approaches are not included in this paper.

7.3 Structural Validity

In this paper, related work section has been undertaken for cloudlet allocation to find the suitable method for developing a QoS improvement strategy. Those strategies continue allocation without considering the load capacity of VM(s). This has been considered as a threat. So, to minimize this threat, a new method has been proposed in this paper, and the results were analyzed meticulously before making a final conclusion.

8 Conclusion and Future Work

This work highlights a new cloudlet allocation policy with the help of a suitable load balancing technique which provides better completion time for the cloudlets and also improves the makespan of the VMs and the host in the DC. Hence, the QoS and the resource utilization of the overall system have been improved in comparison with the other existing cloudlet allocation policies.

In our future study, we shall focus on development of a new cloudlet allocation policy using soft computing tools to identify loads intelligently for the entire available VMs inside a system and keep all the VMs busy as much as possible so that makespan of the whole system will improve. The capacity of the VMs will be indexed in a hash table so that information regarding the execution load of all VMs will be updated dynamically. We shall also investigate live VM migration to the other host inside a data center with the help of the 'Vmotion' Distributed Service [9] in the cloud environment. Another future work is to defend various threats which is a challenging issue in the area of cloud security [20].

Acknowledgments This project is partially and financially supported by UGC DST-Purse Programme in University of Kalyani. The work is enriched by valuable recommendations from the reviewers and editors.

References

1. Xiong, K.; Perros, H.: Service performance and analysis in cloud computing. 978-0-7695- 3708-5/09 $25.00 © 2009 IEEE pp. 693–700
2. Sotomayor, B.; Montero, R.S.; Llorente, I.M.; Foster, I.: Virtual infrastructure management in private and hybrid clouds. 1089-7801/09/$26.00 © 2009 IEEE
3. Adhikari, M.; Banerjee, S.; Biswas, U.: "Smart task assignment model for cloud service provider" Special Issue of International Journal of Computer Applications (0975–8887) on Advanced Computing and Communication Technologies for HPC Applications - ACCTHPCA, (June 2012)
4. Lei, X.; Zhe, X.; Shaowu, M.; Xiongyan, T.: Cloud Computing and Services Platform Construction of Telecom Operator. In: Broadband Network & Multimedia Technology, 2009. IC-BNMT '09. 2nd IEEE International Conference on Digital Object Identifier, pp. 864 – 867
5. Calheiros, R.N.; Ranjan, R.; De Rose, C.A.F.; Buyya, R.: CloudSim: a novel framework for modelling and simulation of cloud computing infrastructures and services (2009)
6. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; Zaharia, M.: A Berkeley view of cloud computing. Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009
7. Aymerich, F.M.; Fenu1, G.; Surcis, S.: An approach to a cloud computing network. 978-1-4244-2624- 9/08/$25.00 ©2008 IEEE 113 pp. 113-118
8. Buyya, R.; Ranjan, R.; Calheiros, R.N.: Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: challenges and opportunities. In: Proceedings of the 7th High Performance Computing and Simulation Conference (HPCS 2009, ISBN: 978-1-4244-4907-1, IEEE Press, New York, USA), Leipzig, Germany, June 21–24, 2009
9. White Paper-VMware Infrastructure Architecture Overview, VMware

10. Ravimaran, S.; MalukMohamed, M.A.: Integrated Obj_FedRep: evaluation of surrogate object based mobile cloud system for Federation, Replica and Data Management. Arab. J. Sci. Eng. **39**, 4577–4592 (2014). doi:10.1007/s13369-014-1001-2
11. Bhatia, W.; Buyy, R.; Ranjan, R.: CloudAnalyst: a CloudSim based visual modeller for analysing cloud computing environments and applications. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446-452, (2010)
12. El-kenawy, E.S.T.; El-Desoky, A.I.; Al-rahamawy, M.F.: Extended max–min scheduling using petri net and load balancing. Int. J. Soft Comput. Eng. (IJSCE) **2**(4), 198–203 (2012)
13. Amalarethinam, D.I.G.; MalaiSelvi, F.K.: A minimum makespan grid workflow scheduling algorithm. 978-1-4577-1583-9/ 12/ $26.00 © 2012 IEEE
14. Syed Abudhagir, U.; Shanmugavel, S.: A novel dynamic reliability optimized resource scheduling algorithm for grid computing system. Arab. J. Sci. Eng. **39**, 7087–7096 (2014). doi:10.1007/s13369-014-1305-2
15. Wee, K.; Mardeni, R.; Tan, S.W.; Lee, S.W.: QoS prominent bandwidth control design for real-time traffic in IEEE 802.16e broadband wireless access. Arab. J. Sci. Eng. **39**, 2831–2842 (2014). doi:10.1007/s13369-013-0931-4
16. Brucker, P.: Scheduling algorithms, Fifth Edition. Springer Press, New York (2007)
17. Chatterjee, T.; Ojha, V.K.; Adhikari, M.; Banerjee, S.; Biswas, U.; Snasel, V.: Design and Implementation of a new Datacenter Broker policy to improve the QoS of a Cloud. In: © Springer International Publishing Switzerland 2014, Proceedings of ICBIA 2014, Advances in Intelligent Systems and Computing, vol. 303, pp 281-290 (2014). doi: 10.1007/978-3-319-08156-4_28
18. Ren, X.; Lin, R.; Zua, H.: A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast. In: Proceeding of IEEE CCIS2011, 978-1-61284-204-2/11/$26.00 ©2011 IEEE
19. A practice of dynamic network load balancingcluster [EB/OL]. http://www.linuxaid.com.cn/articles/1/4/14251644.shtml
20. Chou, T.-S.: Security threats on cloud computing vulnerabilities. Int. J. Comput. Sci. Inf. Technol. (IJCSIT) (2013). doi:10.5121/ijcsit.2013.5306
21. Ajith Singh, N.; Hemalatha, M.: An approach on semi distributed load balancing algorithm for cloud computing system. Int. J. Comput. Appl. **56**(12), 5–10 (2012)
22. Alakeel, A.M.: A guide to dynamic load balancing in distributed computer system. Int. J. Comput. Sci. Netw. Secur. **10**(6), 153–160 (2010)
23. Quansheng, G.; Jiwu, S.; Xiping, M.: Design and implementation of dynamic balance load based on LVS system. Comput. Res. Develop. **41**(16), 923–929 (2004)
24. Adbelzaher, T.F., Bhatti, N.: Web server QoS management by adaptive content delivery[C]. International Workshop on Quality of Service, London, UK (1999)
25. George Amalarethinam, D.I.; Muthulakshmi, P.: An overview of the scheduling policies and algorithms in Grid Computing. Int. J. Res. Rev. Comput. Sci. **2**(2), 280–294 (2011)
26. Mohammad Khanli, L.; Analoui, M.: Resource scheduling in desktop grid by grid-JQA. In: The 3rd International Conference on Grid and Pervasive Computing, IEEE, 2008
27. Ghalem, B.; Fatima Zohra, T.; Wieme, Z.: Approaches to improve the resources management in the simulator CloudSim. In: ICICA 2010, LNCS 6377, pp. 189–196, (2010). doi:10.1007/978-3-642-16167-4_25
28. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; Buyya, R.: CloudSim: a toolkit for modelling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Published online 24 August 2010 in Wiley Online Library (wileyonlinelibrary.com). doi:10.1002/spe.995
29. Rawat, P.S.; Saroha, G.P.; Barthwal, V.: Quality of service evaluation of SaaS modeler (Cloudlet) running on virtual cloud computing environment using CloudSim. Int. J. Comput. Appl. **53**(13), 35–38 (2012)
30. Gulati, A.; Chopra, R.K.: Dynamic round robin for load balancing in a cloud computing. IJCSMC, **2**(6), 274–278 (2013). ISSN 2320–088X
31. Parsa, S.; Entezari-Maleki, R.: RASA: a new grid task scheduling algorithm. Int. J. Digit. Content Technol. Appl. **3**, 91–99 (2009)
32. Makespan: http://www2.informatik.huberlin.de/alcox/lehre/lvws1011/coalg/makespan_scheduling.pdf
33. Campbell, D.T.; Stanley, J.C.: Experimental and quasi-experimental designs for research, Handbook of Research on Teaching, Copyright © 1963 by Houghton Mifflin Company, ISBN: 0-395-30787-2 Y-BBS-IO 09 08
34. Khadka, Ravi.; Saeidi, Amir.; Idu, Andrei.; Hage, Jurrian.; Jansen, Slinger.: Legacy to SOA evolution: a systematic literature review. Technical Report UU-CS-2012-006, March 2012, Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, ISSN: 0924-3275
35. Mattamadugu, L.N.S.; Pathan, A.A.K.: Supercomputing over Cloud using Quicksort algorithm. Master's Thesis, Electrical Engineering,June 2012, School of Computing Blekinge Institute of Technology, SE—371 79. Karlskrona,Sweden
36. Calheiros, R.N.; Ranjan, R.; De Rose, C.A.F.; Buyya, R.: CloudSim: a novel framework for modeling and simulation of cloud computing infrastructures and services. Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 2009
37. Mahajan, K.; Makroo, A.; Dahiya, D.: Round Robin with server affinity: a VM load balancing algorithm for cloud based infrastructure. J. Inf. Process. Syst. **9**(3) (2013). doi:10.3745/JIPS.2013.9.3.379. pISSN 1976-913X
38. Elgedawy, I.: NASEEB: an Escrow-based approach for ensuring data correctness over global clouds. Arab. J. Sci. Eng. **39**(12), 8743–8764 (2014). doi:10.1007/s13369-014-1427-6