Wireless Pers Commun
DOI 10.1007/s11277-017-4946-0

An Approach Towards Amelioration of an Efficient VM Allocation Policy in Cloud Computing Domain

Sourav Banerjee[1] · **Riman Mandal**[2] · **Utpal Biswas**[2]

Abstract Cloud computing is on the horizon of the domain of information technology over the recent few years, giving different remotely accessible services to the cloud users. The quality-of-service (QoS) maintaining of a cloud service provider is the most dominating research issue today. The QoS embraces with different issues like virtual machine (VM) allocation, optimization of response time and throughput, utilizing processing capability, load balancing etc. VM allocation policy deals with the allocation of VMs to the hosts in different datacenters. This paper highlights a new VM allocation policy that distributes the load of VMs among hosts which improves the utilization of hosts' processing capability as well as makespan and throughput of cloud system. The experimental results are obtained by utilizing trace based simulation in CloudSim 3.0.3 and compared with existing VM allocation policies.

Keywords QoS · VM · Host · CloudSim · Throughput · Makespan

1 Introduction

Cloud computing [1–4] has been naturally evolved from the extensive demand of virtualization [5, 6], utility computing [7–9] and service-oriented architecture [10, 11]. The main idea of cloud computing is to serve cloud users [12] or customers on demand service.

✉ Sourav Banerjee
mr.sourav.banerjee@ieee.org

Riman Mandal
mandal.riman@gmail.com

Utpal Biswas
utpal01in@yahoo.com

1 Kalyani Government Engineering College, Kalyani, India 741235

2 University of Kalyani, Kalyani, India 741235

Published online: 08 September 2017

Cloud computing provides difference services [14] like software as a service (SaaS), infrastructure as a service (IaaS), platform as a service (PaaS), development as a service (Daas), and identity as a service (IdaaS) to the customers. Moreover, it gives the user flexibility, scalability and elasticity to acquire and release cloud resources as per the requirement of the application to avail the best performance. It also facilities the users with cloud storage that can be used to store a large amount of data on the cloud and user can access those that form anywhere in the world.

Cloud computing [3, 13] is an extension to the parallel processing [14] and distributed system [15]. It comprises of a collection of large-scale heterogeneous interconnected virtualized computers that are dynamically provisioned and presented as unified computing resource established through negotiation between cloud service providers and cloud users. This enables the cloud computing to use the concept of physical location independence [16] to its best. It provides users with high-end computational infrastructure even in a place where such infrastructure is impossible to establish physically.

Cloud users are served through cloud-based applications as 'pay-per-use' basis. The cloud users are not aware of any background activities like virtual machine (VM) [5, 6] allocation, cludlet distribution, VM migration, load sharing, load distribution, distributed shared memory. They can access cloud-based applications [10, 11] through an authentic login to the cloud system. This cloud applications are liable to perform same or better than the same applications installed on a local machine.

For IaaS one of the significant issues is scheduling of virtual resources and VMs. It has been extensively accepted that VMs can be employed as computing resources for high-performance computing. Thus, efficient VM allocation is imperative in cloud computing environment for optimizing resource utilization and efficient deployment of jobs. Binding or allocating VMs to hosts in a heterogeneous cloud environment is a challenging issue. There are many VM allocation policies [12, 17] in cloud computing which are available to allocate the VM to different hosts in an optimal way. The VM allocation policy plays a vital role in improving the overall system performance minimizing the completion time and makespan [18, 19] of cloudlets. Not only this, a proper allocation policy may lead to the improvement of quality of service (QoS) of the overall system.

1.1 Our Contribution

In this paper, a new VM allocation policy with a suitable load balancing [20] technique has been introduced that will allocate a VM from a sorted vm_list [21] to a host whose remaining load capacity is maximum among all the hosts present in the host_list [21]. The Basic idea behind the proposed allocation policy is to allocate the VMs which has maximum processing capability. Also, the allocation policy is distributing the load among all the hosts based on their remaining load capacity. This proposed policy improves the utilization of hosts processing capability as well as reduces the completion time and makespan of cloudlets. Hence, the performance of the cloud system is improved. Various researches have been conducted, based on scheduling algorithms with different network scenarios and combinations of service classes [22]. The simulation has been done using CloudSim 3.0.3 simulation toolkit [12, 21]. The result of the simulation has been compared with two existing VM allocation policies. The first one is VM allocation policy simple [12] which has been implemented primarily within CloudSim toolkit, and the second allocation policy is RoundRobinVmAllocation Policy [17, 23]. The major drawback of both the VM allocation policies is poor utilization of hosts' processing capability and a very low throughput from the cloud system. The proposed VM allocation policy provides better results than the aforementioned policies. An extensive description of the proposed VM allocation policy is presented in Sect. 3.

1.2 Organization

The rest of the paper is assembled as follows—Sect. 2 describes CloudSim toolkit and two existing VM allocation policies used in CloudSim toolkit. The proposed work has been highlighted with an algorithm and flowchart in Sect. 3. An illustration of proposed work has also been included in Sect. 3. The result of experiment has been described in Sect. 4.2. Section 5 emphasizes analysis of the result and comparative analysis of proposed VM allocation policy with existing policies. Finally, Sect. 6 concludes and discusses the future scope of the proposed work.

2 Related Work

2.1 CloudSim Toolkit

Recently, cloud computing leads the information technology for delivering secure, reliable, fault-tolerant, scalable and feasible computational services, presented as software (SaaS), infrastructure (IaaS), or platform (PaaS) as services. These services may be extended in private data centers (private clouds), may be commercially provided for clients (public clouds), or yet it might be possible that a hybrid cloud [24] can be formed combining both public and private clouds.

These widespread distributed cloud architectures demand a timely, repeatable, and controllable mechanism for evaluating algorithms, policies, and application before real world development of cloud products. A solution to the problem is to simulate the cloud system using a simulation tool, which gives the feasibility of evaluating the hypothesis before software development in an environment where one can recreate tests. In cloud computing where some real currency is used for the services, a Simulation-based approach offers huge benefits as it allows cloud users to test their services in a simulated environment free of cost and to enhance the performance before deploying to the real cloud. At the provider side, simulation environments allow testing different resource allocation strategies under different load and distribution that can maximize the utilization of resources and maximize the profit for the providers. In the absence of such simulation platforms, cloud users and cloud providers have to depend either on theoretical analysis and vague evaluations or on trial-and-error approaches that lead to inefficient service performance.

The general motivation behind CloudSim [12, 21, 25] was to provide a generalized, and extensible simulation framework which provides logical modeling, simulation, and testing of emerging cloud computing services. CloudSim is a customizable cloud infrastructure modeling and simulation toolkit with the support of real-time trading of services between providers and customers. It is possible for the researchers to implement their own work in the domain of cloud by extending different classes of CloudSim toolkit. Different component of CloudSim framework [12] along with the working of CloudSim is shown in Fig. 1.

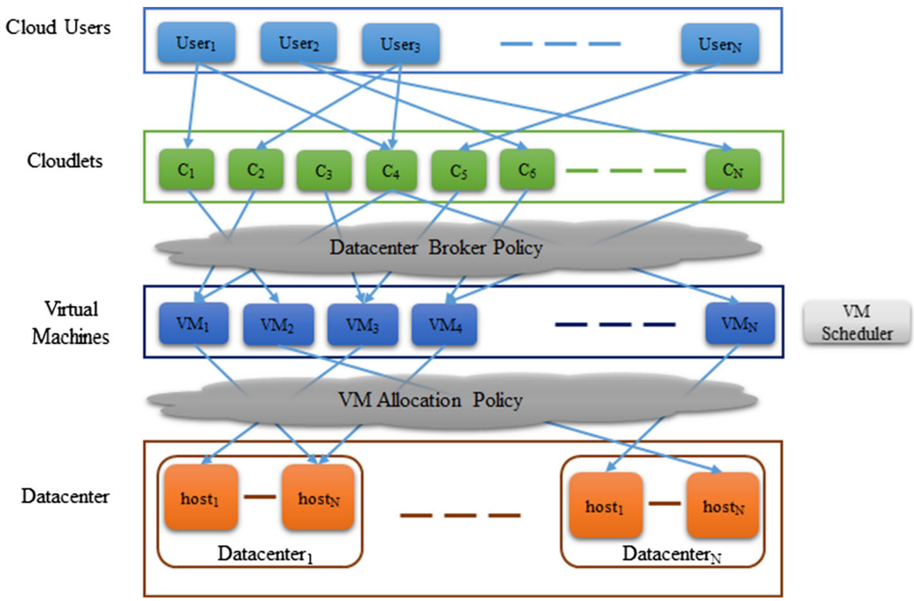

Fig. 1 CloudSim framework

Cloud users request for services to the datacenter through cloudlet. Cloudlet is a collection of requests from customer to availing for services. In CloudSim cloudlets are associated with some properties like cloudlet ID, length, output size, the number of processing element (PE) or cores required to execute the cloudlet etc. The service request of the cloudlets is served by the VMs. The correspondence between cloudlet and VMs is managed by a broker policy [12, 26] named as DatacenterBroker [27]. Each datacenter consists of physical nodes called hosts. VMs are considered as logical machines that can perform some specific tasks. The VMs are created or allocated to the hosts. The allocation of VMs [28] to the hosts are managed by VM allocation policy [12, 23]. VM allocation policy chooses hosts from a datacenter that fulfill the requirement for VM deployment. VM Scheduler [12] is responsible for assigning hosts processing cores to the VMs either in space-share [29] or in time-shared [29, 30] manner. Few allocation strategies (for VMs and cloudlets) are built in CloudSim 3.0.3. Researchers also may apply their allocation strategy by extending this basic classes. The Present study aims at expanding CloudSim by utilizing the VM allocation policy.

2.2 VM Allocation Policy

The VM allocation policy helps in creating a virtual machine (VM) instances to a host within a datacenter. Proper allocation of VMs can result in better utilization of hosts' processing capability. Also, it may help in reducing the completion time and makespan of cloudlets. This results in a better overall throughput of the cloud system. Designing a good VM allocation policy is a challenging work in the cloud computing domain. We have considered two basic VM allocation policy as the basis of our work. The VMs and hosts are represented using some of their properties. A host has the following properties like an identification i.e. host ID, the number of PEs or cores, processing capability of each PE, RAM, storage, bandwidth etc. The VMs also have same kinds of properties like host. A VM can be created to a host if only if can serve all the required physical aspects (number of PE, processing capability, RAM, storage etc) of the VM.

2.2.1 VM Allocation Policy Simple [12]

As the name specifies that this is a simple VM allocation policy. This policy comes inbuilt with CloudSim 3.0.3. This policy considers VM in first come first serve (FCFS) [28] manner and creates the VMs to appropriate host. The host having less number of PEs in use will be chosen for allocation of the VM if it satisfies all the hardware requirement of the VM.

Allocation of a VM will be failure if all the host have tried to allocate the VM but none of them were successful. Suppose there are five VMs say, vm_1, vm_2, vm_3, vm_4 and vm_5 required one PE each and two hosts $host_1$ and $host_2$. we have considered that initially $host_1$ has three PEs and $host_2$ has two PEs. The allocation of VMs to the hosts according to VM allocation policy simple are shown in the Table 1.

2.2.2 Round Robin VM Allocation Policy [23]

Round Robin VM allocation policy also created the VMs in FCFS manner. But the host selection process is slightly different from VM allocation policy simple. Here the host_list is consider as a circular_host_list [21]. The next host in the circular_host_list is always be chosen as the destination of current VM. Table 2 emphasizes the allocation fashion using Round Robin VM allocation policy considering the five VMs (vm_1, vm_2, vm_3, vm_4, and vm_5) and two hosts ($host_1$ and $host_2$).

Table 1 Allocation of VMs using VM allocation policy simple

VM	Host	Number of free PEs of $host_0$	Numnber of free PEs of $host_1$
vm_1	$host_1$	2	2
vm_2	$host_1$	1	2
vm_3	$host_2$	1	1
vm_4	$host_1$	0	1
vm_5	$host_2$	0	0

Table 2 Allocation of VMs using Round Robin VM allocation policy

VM	Host
vm_1	$host_1$
vm_2	$host_2$
vm_3	$host_1$
vm_4	$host_2$
vm_5	$host_1$

Both the existing policies have not considered the processing capability of VMs and hosts while trying to allocate the VMs to hosts. Sometimes it may happen that a VM having better processing capability does not allocated to any of the hosts. The VMs with higher processing capability can process the Cloudlets faster. Also the load on the hosts is not distributed properly in the existing policies. To overcome some of this problems, we have proposed a new VM allocation policy which will be discussed in Sect. 3, gives better utilization of hosts' processing capability and also reduces the makespan of cloudlets and gives better throughput from the cloud system.

3 Proposed Work

The proposed algorithm will allocate a VM with maximum MIPS to the host with maximum RLC (remaining load capacity) value. The RLC value of every host will be estimated dynamically and a new VM will be allocated to a host based on the present RLC value. Two major parameters used in the proposed work are

Load capacity This parameter defines the amount of allocable load to a host so that it performs in its optimal capacity.

$$LC = (PL_{Host}/MAL_{Host}) * 100\% \quad (1)$$

- LC = load capacity
- PL_{Host} = present load on host
- MAL_{Host} = maximum allowable load on host

Remaining load capacity This parameter reflects the remaining amount of load allocable to a host machine. The optimal capacity of a host can be deduced by evaluating this parameter.

$$RLC = 100 - (PL_{Host}/MAL_{Host}) * 100\% \quad (2)$$

- RLC = remaining load capacity

The entire methodology of the proposed work has been explained with a suitable example in Sect. 3.3.

3.1 Proposed Algorithm

The input to the algorithm will be the number of hosts i.e. host_list and number of VMs i.e. vm_list to be allocated to the hosts. This algorithm facilitates the allocation of VMs to the available hosts to enhance the performance service delivery. The entire host list will be traversed for finding out a host with maximum RLC among all. The VM cannot be allocated if there is no suitable host available.

Algorithm 1: Proposed Algorithm

```
   Data: host_list, vm_list
1  initialization;
2  for each VM_i from vm_list according to descending order of their processing
   capability do
3      for each Host_i in the host_list do
4          calculate RLC of host_i;
5      end
6      host_j = Host_RLC_MAX;
7      Try to allocate VM_i to host_j;
8      if the allocation failed then
9          host_list = host_list - host_j;
10         go to step 6;
11     end
12     if VM_i could not be allocated to any Host after trying all the Hosts in host_list
       then
13         The allocation of VM_i is failed;
14     end
15     else
16         VM_i is allocated to host_j;
17     end
18 end
```

3.2 Flowchart of Proposed Allocation Policy

The pictorial representation of proposed VM allocation policy has been shown in Fig. 2

3.3 Illustration of the Proposed Algorithm

Here, we have considered six VMs and two hosts. The entire work is evaluated using present load, present RLC, and load after allocation and RLC after allocation. Those are discussed below:

- *Present load (%)* shows the current load to the specific host before allocation of a VM.
- *Present RLC (%)* shows the current RLC in percent of the host before allocation of a VM.
- *Load after allocation (%)* shows the load after allocating a VM to the host.
- *ALC after allocation (%)* shows the RLC after allocating a VM to the host.

The list of VMs and hosts with their corresponding processing capabilities are given in Tables 3 and 4 respectively.

The entire allocation strategy of VM(s) to the host(s) has been explained in the following manner. The procedure is depicted stepwise that will facilitate the readers to understand the entire methodology very easily.

Step 1 At first the VMs will be sorted according to their processing capabilities which is shown in Table 5. In Fig. 3 the arrangement of VMs are shown. We assume that initially all the hosts are free. Hence Present load of each host is 0% and Present RLC value is 100% as shown in Table 6.

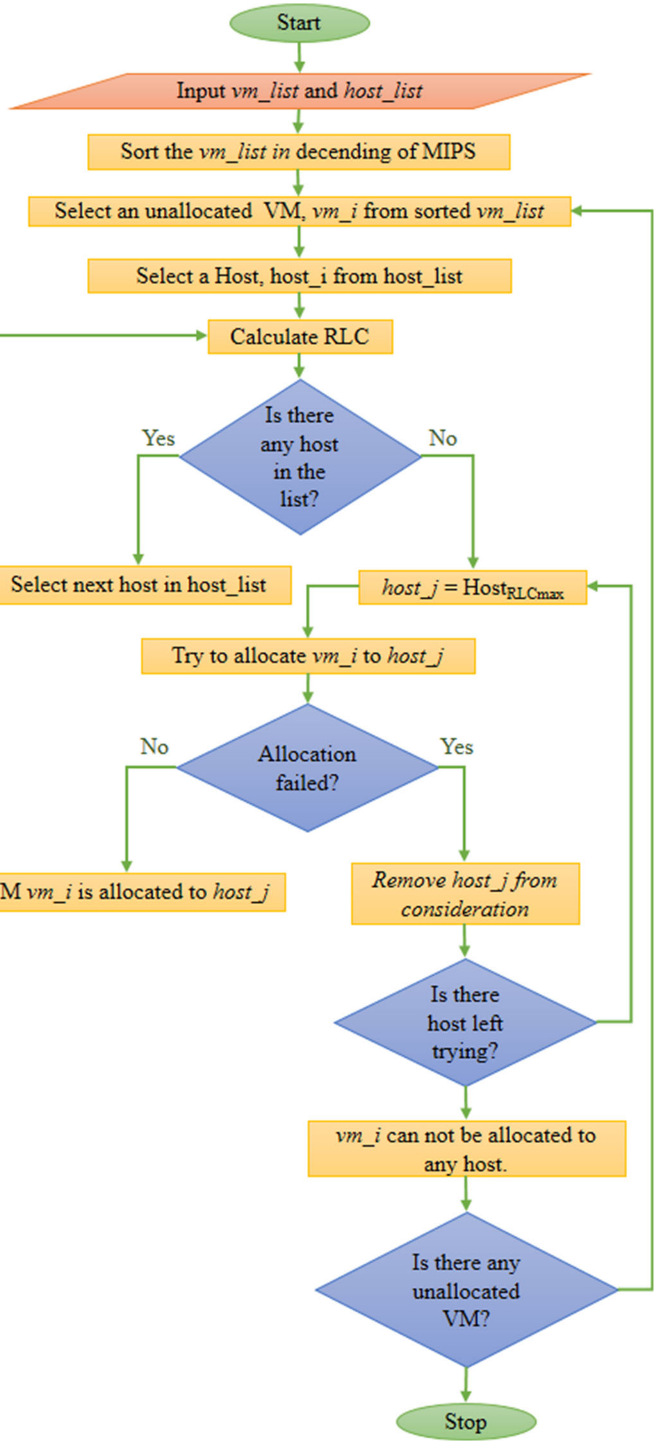

Fig. 2 Flowchart of proposed VM allocation policy

Table 3 List of VMs for illustration

VM	Processing capability
vm_1	200
vm_2	700
vm_3	400
vm_4	600
vm_5	500
vm_6	400

Table 4 Host list for illustration

Hosts	Processing capability (MIPS)
$host_1$	1500
$host_2$	1000

Table 5 Sorted VM list

VM	Processing capability
vm_2	700
vm_4	600
vm_5	500
vm_6	400
vm_3	400
vm_1	200

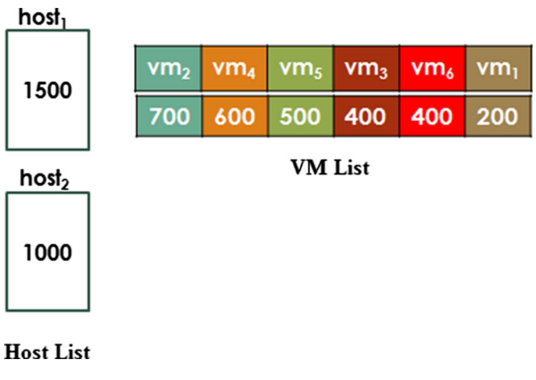

Fig. 3 VMs are shorted and ready to be allocated

Table 6 Load and RLC of hosts at initial stage

Host	Present load (%)	Present RLC (%)
$host_1$	0	100
$host_2$	0	100

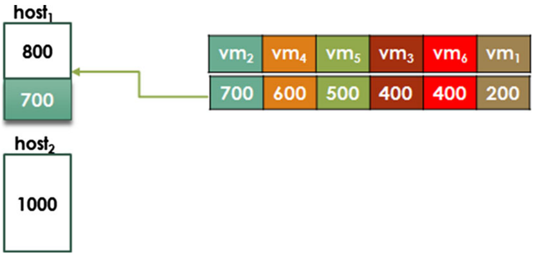

Fig. 4 Allocation of vm_2 to $host_1$

Table 7 Allocation of vm_2 to $host_1$

Host	Present load (%)	Present RLC (%)	Load after allocation (%)	RLC after allocation (%)
$host_1$	0	100	46.67	53.33
$host_2$	0	100	0	100

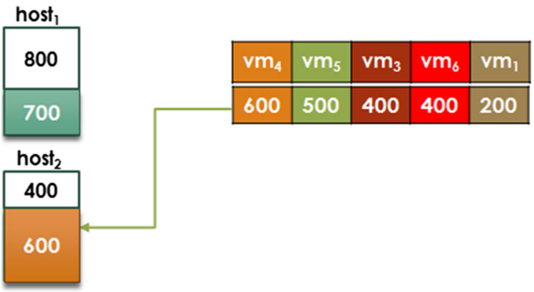

Fig. 5 Allocation of vm_4 to $host_2$

Step 2 According to the proposed algorithm vm_2 will be allocated to $host_1$ which is presented in Fig. 4. After this allocation, load of $host_1$ becomes 46.67% and RLC becomes 53.33% which is shown in Table 7.

Step 3 In this step vm_4 will be allocated. Based on the proposed algorithm vm_4 will be allocated to $host_2$. The allocation is shown in Fig. 5. After the allocation the RLC of $host_2$ becomes 40%. The allocation scenario is shown in Table 8.

Step 4 Now vm_5 will take part for this allocation process. Following the proposed algorithm vm_5 will be allocated to $host_1$ which is shown in Fig. 6. After successful allocation, the load of $host_1$ and $host_2$ becomes 66.67 and 60% respectively. The allocation scenario is depicted in Table 9.

Step 5 In this step, vm_3 will be allocated. Based on the proposed algorithm vm_3 will be allocated to $host_2$. The allocation is shown in Fig. 7. After the allocation, the load on $host_2$ becomes 100% and RLC becomes 0% which reflects no further allocation is possible in $host_2$ that is shown in Table 10.

Step 6 In this step the allocation of vm_6 will be attempted. It is shown in Fig. 8 though $host_1$ has available processing capability of 300MIPS its not possible to allocate vm_6. As it requires 400MIPS. So it is not possible to allocate vm_6 in $host_1$. As the $host_2$ has already been reached its maximum capacity so it cannot accommodate any VMs. As a result the allocation of vm_6 will be failed.

Table 8 Allocation of vm_4 to $host_2$

Host	Present load (%)	Present RLC (%)	Load after allocation (%)	RLC after allocation (%)
$host_1$	46.67	53.33	46.67	53.33
$host_2$	0	100	60	40

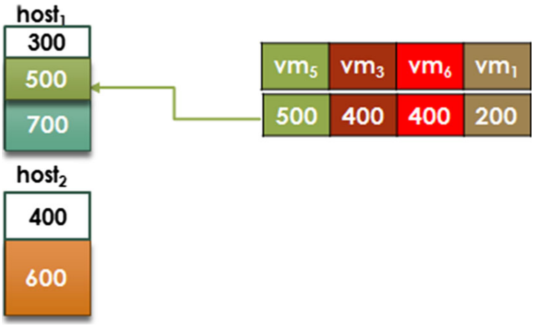

Fig. 6 Allocation of vm_5 to $host_1$

Table 9 Allocation of vm_5 to $host_1$

Host	Present load (%)	Present RLC (%)	Load after allocation (%)	RLC after allocation (%)
$host_1$	46.67	53.33	66.67	33.33
$host_2$	60	40	60	40

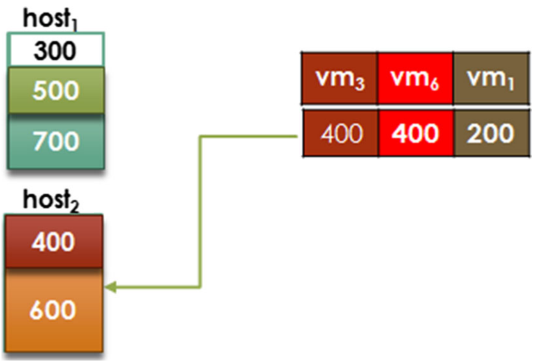

Fig. 7 Allocation of vm_3 to $host_2$

Step 7 In this step vm_1 will be attempted to allocate to a host. It is clear from Fig. 8 that $Host_1$ can allocate vm_1. Now after the allocation the load on $host_1$ becomes 93.33% and RLC becomes 6.67% which is shown in Table 11. The Fig. 9 depicts the present status after allocating vm_1 to $host_1$.

Table 10 Allocation of vm_3 to $host_2$

Host	Present load (%)	Present RLC (%)	Load after allocation (%)	RLC after allocation (%)
$host_1$	66.67	33.33	66.67	33.33
$host_2$	60	40	100	0

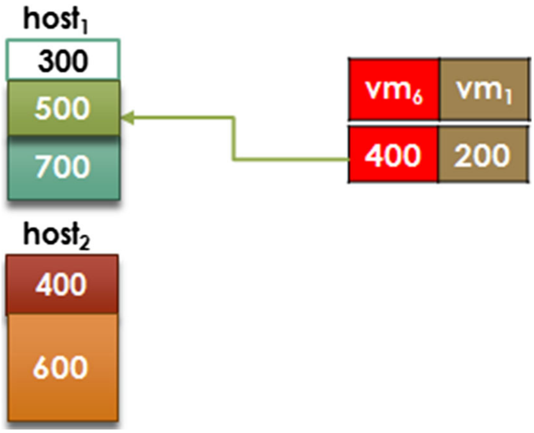

Fig. 8 Attempt to allocate vm_6

Table 11 Allocation of vm_1 to $host_1$

Host	Present load (%)	Present RLC (%)	Load after allocation (%)	RLC after allocation (%)
$host_1$	66.67	33.33	93.67	6.33
$host_2$	100	0	100	0

In this way the VM(s) will be allocated to the host(s). Table 12 shows the final VM to hosts mapping.

4 Experimental Result

4.1 Datasets Used in the Experiment

Due to space constraints we have considered 2 hosts, 35 cloudlets and 20 VMs in this paper. In Table 13 different characteristics of the hosts are mentioned. Each hosts has unique identification number (Host_ID), number of processing elements (PEs), processing capability of each PE and the total processing capability of the host. The total processing capability of a host can be calculated using the Eq. 3.

$$TPC_{Host} = PE_{Host} * PC_{PE} \tag{3}$$

Fig. 9 Allocation of vm_1 to $host_1$

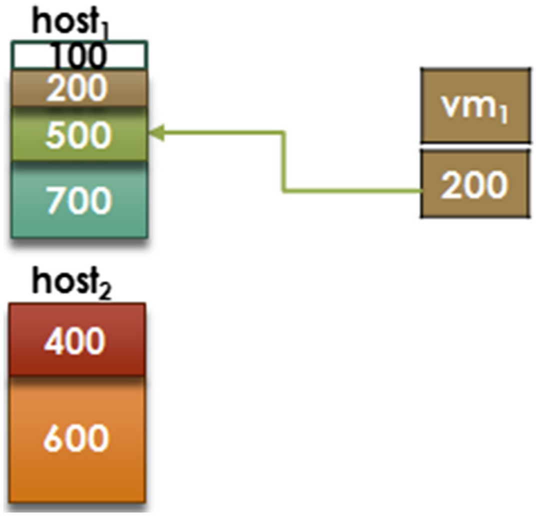

Table 12 Final allocation of VM(s) to the host(s)

VM	Host
vm_2	$host_1$
vm_4	$host_2$
vm_5	$host_1$
vm_3	$host_2$
vm_1	$host_1$
vm_6	Unallocated

Table 13 List of hosts in one datacenter

Host_ID	Number of PEs	Processing capability of PE	Processing capability of host
$host_1$	4	1000	4000
$host_2$	2	1000	2000

- TPC_{Host} = total processing capability of a host
- PE_{Host} = number of PEs in a host
- PC_{PE} = processing capability of a PE

Table 14 shows all the VMs that are to be allocated to the hosts with their different characteristic. Each VM has an identification number (VM_ID), processing capability.

In Table 15, all the cloudlets with their unique identification (Cld_ID) and their lengths are mentioned.

4.2 Result of the Experiment

The complete result set of the experiment are given in this section.

Table 14 List of VMs to be allocated

VM_ID	Processing capability
vm_1	800
vm_2	200
vm_3	600
vm_4	800
vm_5	200
vm_6	900
vm_7	200
vm_8	300
vm_9	500
vm_{10}	700
vm_{11}	500
vm_{12}	800
vm_{13}	500
vm_{14}	800
vm_{15}	500
vm_{16}	900
vm_{17}	400
vm_{18}	900
vm_{19}	1000
vm_{20}	1000

4.3 Allocation of the VMs to the Hosts

Table 16 shows the allocation of VMs using proposed VM allocation policy. vm_7, vm_6, vm_4 and vm_1 cannot be allocated using proposed VM allocation policy. For non-allocated VMs the datacenter is marked as not-applicable (NA) in the table. From the table it is cleared our proposed policy allocates the VMs having maximum processing capability. The non-allocated VMs are having least processing capability in the VM list.

4.4 Finish Time of Cloudlets in the Allocated VMs

In Table 17 the execution of cloudlets on the VMs allocated by the proposed VM allocation policy are shown.

5 Comparison and Analysis

We have compared proposed algorithm with two of the existing VM allocation policies i.e. VM allocation simple and Round Robin VM allocation. For the comparison purpose we have used same datasets shown in Sect. 4.1. We have considered different sets of cloudlets for better analyzing the performance of the allocation Policies. The allocations of VMs directly affects the execution of cloudlets and also it has a direct impact on throughput of the entire system. We have considered three metrics, i.e. utilization of hosts' processing capability (UHPC), makespan and throughput for evaluating the performance of the proposed algorithm.

Table 15 List of Cloudlets to be executed

Cld_ID	Length
cld_1	300
cld_2	1000
cld_3	900
cld_4	800
cld_5	1000
cld_6	600
cld_7	500
cld_8	1000
cld_9	800
cld_{10}	300
cld_{11}	400
cld_{12}	400
cld_{13}	400
cld_{14}	1000
cld_{15}	900
cld_{16}	200
cld_{17}	300
cld_{18}	700
cld_{19}	7000
cld_{20}	600
cld_{21}	1000
cld_{22}	400
cld_{23}	800
cld_{24}	200
cld_{25}	700
cld_{26}	900
cld_{27}	500
cld_{28}	400
cld_{29}	200
cld_{30}	400
cld_{31}	400
cld_{32}	700
cld_{33}	200
cld_{34}	800
cld_{35}	500

5.1 Utilization of Hosts' Processing Capability (UHPC)

The utilization of hosts' processing capability can be determined as a factor of total processing capability of VMs that were allocated and the total available processing capability that is shown is Eq. 4.

$$UHPC = (TPC_{AVM}/TPC_{Host}) * 100\% \tag{4}$$

Table 16 Allocated VMs with corresponding host and datacenter using proposed policy

VM_ID	Datacenter_ID	Host_ID	Processing capability
vm_{20}	1	$host_2$	1000
vm_{19}	1	$host_1$	1000
vm_{18}	1	$host_1$	900
vm_{16}	1	$host_1$	800
vm_6	1	$host_2$	900
vm_{14}	1	$host_1$	900
vm_{17}	1	$host_1$	400
vm_{12}	1	$host_2$	800
vm_4	1	$host_1$	800
vm_1	2	$host_1$	800
vm_{10}	2	$host_2$	700
vm_3	2	$host_1$	600
vm_{15}	2	$host_1$	500
vm_{13}	2	$host_1$	500
vm_{11}	2	$host_2$	500
vm_9	2	$host_1$	500
vm_8	NA	Failed	300
vm_7	NA	Failed	200
vm_5	NA	Failed	200
vm_2	NA	Failed	200

- TPC_{AVM} = total processing capability of allocated VMs
- TPC_{Host} = total processing capability of host

Following the Eq. 4, the utilization of hosts' processing capability has been calculated for all the algorithms and shown in Table 18. Figure 10 shows a comparative analysis of utilization of hosts' processing capability between the existing VM allocation policies and our proposed VM allocation policy. This figure shows the improvement in UHPC (%) performance using our proposed algorithm. The total processing capability of Allocated VMs is basically consumed processing capability of host.

5.2 Comparison Based on Makespan

For this comparison, We have considered five batches of cloudets. Each batch having 500, 1000, 5000, 10,000, 25,000 number of cloudlets respectively. Table 19 shows the comparison among the makespan of different algorithms running different number of cloudlets. In Fig. 11 gives a comparative analysis between three VM allocation policies based on makespan.

5.2.1 Rate of Improvement for Makespan

The rate of improvement (ROI) considering makespan of proposed allocation policy over the existing ones is given in Table 20.

Table 17 Finish time of Cloudlets

Cld_ID	Length	VM_ID	Finish time
cld_1	300	vm_{20}	8.2
cld_2	1000	vm_{19}	25.31
cld_3	900	vm_{18}	23.53
cld_4	800	vm_{16}	15.75
cld_5	1000	vm_6	22.42
cld_6	600	vm_{14}	12.7
cld_7	500	vm_{17}	25.2
cld_8	1000	vm_{12}	15.31
cld_9	800	vm_4	18.95
cld_{10}	300	vm_1	7.7
cld_{11}	400	vm_{10}	11.63
cld_{12}	400	vm_3	13.53
cld_{13}	400	vm_{15}	12.2
cld_{14}	1000	vm_{12}	28.2
cld_{15}	900	vm_{11}	26.2
cld_{16}	200	vm_9	8.2
cld_{17}	300	vm_{20}	8.2
cld_{18}	700	vm_{19}	21.2
cld_{19}	700	vm_{19}	21.31
cld_{20}	600	vm_{16}	13.53
cld_{21}	1000	vm_6	22.42
cld_{22}	400	vm_{14}	10.2
cld_{23}	800	vm_{17}	32.7
cld_{24}	200	vm_{12}	5.2
cld_{25}	700	vm_4	17.7
cld_{26}	900	vm_1	15.2
cld_{27}	500	vm_{10}	13.06
cld_{28}	400	vm_3	13.53
cld_{29}	200	vm_{15}	8.2
cld_{30}	400	vm_{13}	16.2
cld_{31}	400	vm_{11}	16.2
cld_{32}	700	vm_9	18.2
cld_{33}	200	vm_{20}	6.2
cld_{34}	800	vm_{19}	23.2
cld_{35}	500	vm_{18}	16.86

Table 18 UHPC

Allocation policy	Hosts' processing capability	Consumed processing capability	Utilization (%)
VM allocation policy simple	12,000	9600	80
Round robin VM allocation policy	12,000	10,100	84.17
Proposed VM allocation policy	12,000	11,600	96.6

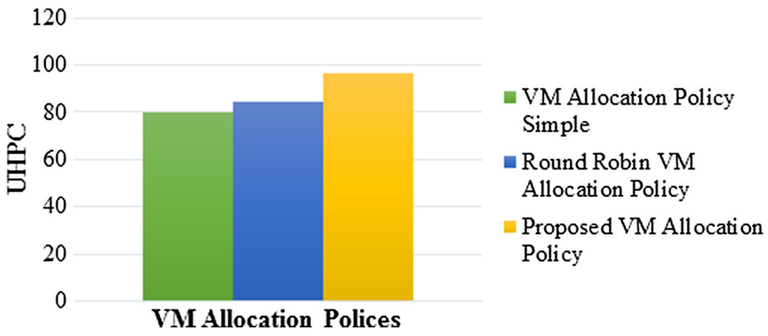

Fig. 10 Comparison based of UHPC

Table 19 Comparison based on makespan

Batch of Cloudlets	VM allocation policy simple	Round Robin VM allocation policy	Proposed VM allocation policy
500	840.14	995.10	485.14
1000	1610.02	1764.95	917.55
5000	8049.07	9083.82	4734.43
10,000	15,842.77	17,897.53	9283.67
25,000	39,898.31	44,952.97	23,486.35

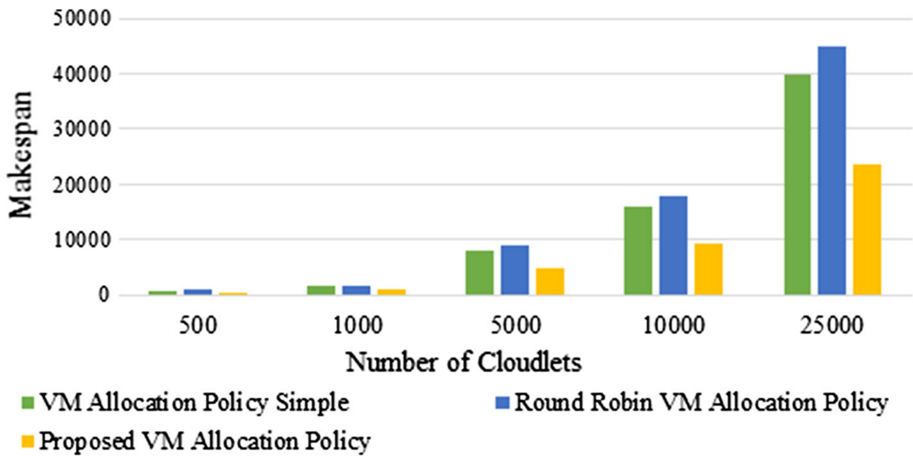

Fig. 11 Comparison based on makespan

5.3 Comparison Based on Throughput

The throughput of the cloud system can be calculated as a factor between number of cloudlets processed and the time taken to process all those cloudlets. The formula for throughput of cloud system is given in the Eq. 5.

Table 20 ROI for makespan

Batch of Cloudlets	VM allocation policy simple (%)	Round Robin VM allocation policy (%)
500	42.1	51.2
1000	43.4	48.0
5000	41.2	47.9
10,000	41.4	48.1
25,000	41.1	48.8

Table 21 Comparison based on throughput

Batch of Cloudlets	VM allocation policy simple	Round Robin VM allocation policy	Proposed VM allocation policy
500	0.59	0.50	1.03
1000	0.62	0.56	1.09
5000	0.62	0.55	1.07
10,000	0.63	0.56	1.08
25,000	0.63	0.57	1.06

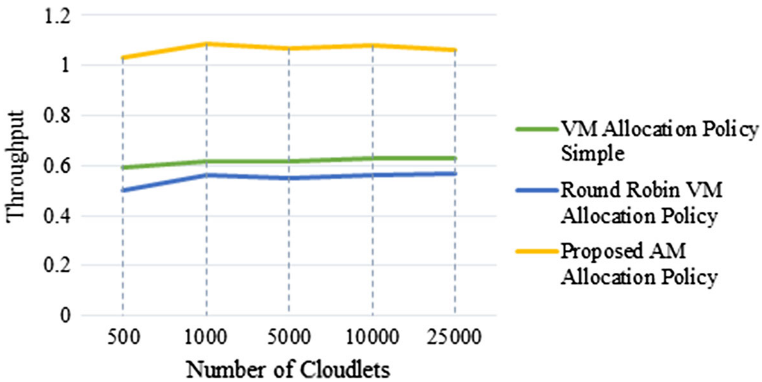

Fig. 12 Comparison based on throughput

$$Throughput = Number\ of\ Cloudlets\ Processed\ /\ makespan \tag{5}$$

Using Eq. 5, the throughput of the cloud system can be calculated for different number of cloudlets executed and shown in Table 21. Plotting these values in a graph we get Fig. 12 that shows the difference between the throughputs of the cloud system in case of using different VM allocation policies. The proposed policy provides better throughput in comparison with the other existing ones.

5.3.1 Rate of Improvement for Throughput

The rate of improvement (ROI) considering throughput of cloud system of proposed allocation policy over the existing ones is given in Table 22.

Table 22 ROI for throughput

Batch of Cloudlets	VM allocation policy simple (%)	Round Robin VM allocation policy (%)
500	74.6	106
1000	76.0	94.6
5000	72.6	94.5
10,000	71.4	92.9
25,000	68.2	86.0

6 Conclusion

This paper highlights a new VM allocation policy that gives better UHPC. We have also shown that allocating VM using our allocation policy decreases the makespan of cloudlet and increases the throughput of the cloud system. Hence the QoS of the cloud system has been improved over the existing policies.

References

1. Xiong, K., & Perros, H. (2009). Service performance and analysis in cloud computing. In *Conference congress on services—I* (p. 693–700), Los Angeles, CA, IEEE.
2. Sotomayor, B., Montero, R. S., Llorente, I. M., & Foster, I. (2009). Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, *13*(5), 14–22.
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, et al. (2009). *A Berkeley view of cloud computing, Technical Report No. UCB/EECS-2009-28*, University of California at Berkley, USA.
4. Yang, J., Khokhar, A., Sheikht, S., & Ghafoor, A. (1994). Estimating execution time for parallel tasks in heterogeneous processing (HP) environment. In *Heterogeneous Computing Workshop, 1994, Proceedings* (pp. 23–28).
5. Buyya, R., Ranjan, R., & Calheiro, R. N. (2009). Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *International conference on high performance computing & simulation*, 2009. HPCS 09 (pp. 1–11).
6. White Paper—VMware infrastructure architecture overview. *VMware* (2006).
7. Nickolov, P., Armijo, B., & Miloushev, V. (2013). *Globally distributed utility computing cloud*. U.S. Patent, US 8429630 B2, 429(8), 630.
8. Canali, C., Rabinovich, M., & Xiao, Z. (2005). Utility computing for Internet applications. In *Web Content Delivery*. Springer US (Vol. II, pp. 131–151).
9. Jiang, X., & Xu, D. (2003). Soda: A service-on-demand architecture for application service hosting utility platforms. In *Proceedings 12th IEEE international symposium on high performance distributed computing*, 2003, IEEE (pp. 174–183).
10. Lei, X., Zhe, X., Shaowu, M., & Xiongyan, T. (2009). Cloud computing and services platform construction of telecom operator. In *2nd IEEE international conference on digital object identifier*. Broadband Network & Multimedia Technology, IC-BNMT'09 (pp. 864–867)
11. Adhikari, M., Banerjee, S., & Biswas, U. (2012). 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*.
12. Calheiros, R. N., Ranjan, R., De Rose, C. A. F., & Buyya, R. (2009). *CloudSim: A novel framework for modelling and simulation of cloud computing infrastructures and services*, Echnical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia.
13. Xuejie, Zhang, Zhijian, Wang, & Feng, Xu. (2013). Reliability evaluation of cloud computing systems using hybrid methods. *Intelligent Automation & Soft Computing*, *19*(2), 165–174.
14. Almasi, G. S., & Gottlieb, A. (1988). *Highly parallel computing*. Menlo Park, CA: Benjamin-Cummings Pub. Co., United States.

15. Cole, R., & Vishkin, U. (1986). Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, *70*(1), 32–53.
16. Elgan, M. (2009). Is digital nomad living going mainstream? Computerworld, Retrived on 1 August 2009 From http://www.computerworld.com/article/2526618/mobile-wireless/is-digital-nomad-living-going-mainstream-.html.
17. Pasha, N., Agarwal, A., & Rastogi, R. (2014). Round robin approach for VM load balancing algorithm in cloud computing environment. *International Journal of Advanced Research in Computer Science and Software Engineering, 4*(5), 34–39.
18. Amalarethinam, D. I. G., & MalaiSelvi, F. K. (2012). A minimum makespan grid workflow scheduling algorithm. *International Conference on Computer Communication and Informatics (ICCCI)*, *2012*, 1–6.
19. Abudhagir, U. S., & Shanmugavel, S. (2014). A novel dynamic reliability optimized resource scheduling algorithm for grid computing system. *Arabian Journal for Science and Engineering*, *39*(10), 7087–7096.
20. Feng, Y., Zhijian, W., Feng, X., Yuanchao, Z., Fachao, Z., & Shaosong, Y. (2013). A novel cloud load balancing mechanism in premise of ensuring QoS. *Intelligent Automation & Soft Computing*, *19*(2), 151–163.
21. Bhatia, W., Buyya, R., & Ranjan, R. (2010). CloudAnalyst: a CloudSim based visual modeller for analysing cloud computing environments and applications. In *24th IEEE International conference on advanced information networking and applications*, 2010 (pp. 446–452).
22. Wee, K., Mardeni, R., Tan, S. W., & Lee, S. W. (2014). QoS prominent bandwidth control design for real-time traffic in IEEE 802.16e broadband wireless access. *Arabian Journal for Science and Engineering*, *39*(4), 2831–2842.
23. Leite, A. (2013). An implementation of Round-Robin VmAllocationPolicy of CloudSim framework, GitHub. Retrived from https://gist.github.com/alessandroleite/4072341.
24. Sajid, M., & Raza, Z. (2015). Turnaround time minimization-based static scheduling model using task duplication for fine-grained parallel applications onto hybrid cloud environment. *IETE Journal of Research*. doi:10.1080/03772063.2015.1075911.
25. Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, *41*(1), 23–50.
26. Chatterjee, T., Ojha, V. K., Adhikari, M., Banerjee, S., Biswas, U., & Snasel, V. (2014). Design and implementation of an improved datacenter broker policy to improve the QoS of a cloud. In: *Proceedings of ICBIA 2014, Advances in Intelligent Systems and Computing* (Vol. 303, pp. 281–290). Springer International Publishing Switzerland 2014.
27. Banerjee, S., Adhikari, M., Kar, S., & Biswas, U. (2015). Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud. *Arabian Journal for Science and Engineering*, *40*(5), 1409–1425.
28. Quiroz, A., Kim, H., Parashar, M., Gnanasambandam, N., & Sharma, N. (2009). Towards autonomic workload provisioning for enterprise grids and clouds. In *10th IEEE/ACM international conference on grid computing '09, IEEE* (pp. 5057).
29. Zhang, Y., Franke, H., Moreira, J. E., & Sivasubramaniam, A. (2002). A comparative analysis of space- and time-sharing techniques for parallel job scheduling in large scale parallel systems. In *IEEE transactions on parallel and distributed system.*
30. Shaw, J. C. (1964). JOSS: A designer's view of an experimental on-line computing system. In *Proceeding AFIPS '64 (Fall, part I) Proceedings of the October 27–29, fall joint computer conference, Part I* (pp. 455–464)
31. Sundarapandian, V. (2009). *Queueing theory, probability, statistics and queueing theory chapter 7* (pp. 686-749). New Delhi: PHI Learning.

Sourav Banerjee is working as an Assistant Professor in the Department of Computer Science and Engineering, Kalyani Government Engineering College since 2008. He has completed his Bachelor of Engineering degree from the University of Burdwan in the year 2004 and in 2006 he completed his M.Tech. in Computer Science and Engineering from the University of Kalyani. His working domain is Distributed System, Cloud Computing, Cyber Security, and Mobile Cloud. He is a member of IEEE, ACM, MIR Lab USA, IAENG. Contribution: Supported suggested, proposed and simulated, improvisation and edited most of the document.

Riman Mandal is perusing Master of Technology in Computer Science and Engineering. He has completed his Master of Computer Application (M.C.A.) from the Kalyani University in the year 2015 and in 2012 he completed his Bachelor of Computer Application (B.C.A.) from Malda College under University of Gour Banga. 2015. His research interests are in Cloud Computing, Distributed System, Mobile Cloud, Operating System and Algorithms. Contribution: Proposed the main theory and simulated the work. Also managed to write most of the document.

Dr. Utpal Biswas received his B.E., M.E. and Ph.D. degrees in Computer Science and Engineering from Jadavpur University, India in 1993, 2001 and 2008 respectively. He served as a faculty member in NIT, Durgapur, India in the department of Computer Science and Engineering from 1994 to 2001. Currently, he is working as an associate professor in the department of Computer Science and Engineering, University of Kalyani, West Bengal, India. He has over 100 research articles in different journals, book chapters and conferences. His research interests include optical communication, ad-hoc and mobile communication, semantic web services, E-governance, Cloud Computing etc. Contribution: The entire work is done under his keen supervision.