Arab J Sci Eng
DOI 10.1007/s13369-017-2781-y

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

An Approach Toward Amelioration of a New Cloudlet Allocation Strategy Using Cloudsim

Sourav Banerjee[1] · **Aritra Roy**[1] · **Amritap Chowdhury**[1] · **Ranit Mutsuddy**[1] · **Riman Mandal**[2] · **Utpal Biswas**[2]

Received: 18 November 2016 / Accepted: 1 August 2017

Abstract Cloud computing is a varied computing archetype uniting the benefits of service-oriented architecture and utility computing. In cloud computing, resource allocation and its proper utilization, to achieve a higher throughput and quality of service (QoS), has become a great research issue. This paper highlights a new cloudlet allocation strategy that utilizes all available resources efficiently and enhances the QoS by applying deadline-based workload distribution. It is believed that this paper would benefit both cloud users and researchers in various aspects. The entire experiment is done in Cloudsim Toolkit-3.0.3, by modifying the required classes.

Keywords Cloud computing · Service-oriented architecture (SOA) · Quality of service (QoS) · Cloudsim Toolkit-3.0.3 · Cloud user (CU)

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

Aritra Roy
aritraroy.kolkata@gmail.com

Amritap Chowdhury
amribuchai@gmail.com

Ranit Mutsuddy
mutsuddyranit6@gmail.com

Riman Mandal
mandal.riman@gmail.com.com

Utpal Biswas
utpal01in@yahoo.com

[1] Kalyani Government Engineering College, Kalyani, Nadia, India

[2] Department of Computer Science and Engineering, University of Kalyani, Kalyani, India

1 Introduction

Cloud [1] is a group of computers or servers that are connected together to provide resources to the cloud users. It has emerged as a brand new computing architype that aims at supplying reliable quality of service (QoS) [2]. The main problems related to cloud computing are network bandwidth [3], response time [4] and minimum delay in data transfer. In cloud computing, resource allocation [5–10] plays an important role in reducing the computational cost and enhancing the performance, which in turn influences the level of satisfaction among cloud users, who uses the cloud resources and computing power as Internet services [11,12]. Cloud users can assess the data, resources and computing power without being concerned about the infrastructure, workload and processing underneath. This empowers the cloud users to access the services without any concern of infrastructure, workload or data processing. Cloud computing is a paradigm in which information is invariably stored in servers on the Internet and cached temporarily on cloud users PCs, entertainment centers, notebooks, sensors, and business organizations, etc.

Cloud computing is mainly based on distributed systems with parallel processing [13] capabilities that has high fault tolerance and data consistency throughout the environment (Handling the crash failures [14]). The service is provided to the cloud users as per demand over many geographical locations without any degradation of QoS by dynamically provisioning [15–17] the services through Internet. So utility based or Gossip based [18] resources allocation should be efficient to provide proper virtualization and physical independence; the basic characteristics of cloud.

Cloud computing paradigm makes the self-managed [19] resources as a single point of admittance to the number of cloud users and is implemented as pay per use basis. Though

Published online: 19 August 2017

there are a number of benefits of cloud computing such as virtual environment, dynamic infrastructure, pay per consume and installation-free infrastructure (both software and hardware), the major concern lies in the order in which the requests are satisfied which involves scheduling of the resources. Cloud computing is mainly rented or demanded by the cloud users on timely basis, usually in hours or minutes. So the scheduling algorithm has to be efficient enough to utilize the resources efficiently by a good resource management.

Cloud computing is a huge economical and commercial technology to deliver services like storage services, software services and infrastructure to the users with physical transparency. The processes involved in VMs (virtual machines) resources allocation [20,21], load balancing [2,15], load sharing, distributed shared memory [22] access, VM migration [23] and processor allocation [24] are completely abstracted from the user.

The end users or the cloud users can only access the cloud-based applications and infrastructure by logging into a cloud interface. The cloud service providers [25] strive to give the same or better service and performance than the software programs installed locally on end-user machines. Binding or allocating cloudlet to VM in a heterogeneous cloud environment is a challenging issue. The cloudlet [30,31] is considered as task in cloudsim. To make the environment more proficient, it requires an efficient allocation policy. There are many cloudlet allocation policies [26–28] available in cloud computing to allocate the cloudlets to the different resources or VMs in an optimal way. The cloudlet allocation policy plays a vital role in improving the overall system performance that is minimizing the completion time or makespan of VMs as well as the host. 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 the improvement in the quality of service [29] of the overall system.

1.1 Our Contribution

In this paper, a new cloudlet allocation policy has been introduced to allocate the cloudlets to two clusters namely high-end resource cluster (HERC) and low-end resource cluster (LERC). The main objective is to divide the cloudlets and reduce the completion time or the makespan as a whole. A deadline has been introduced (refer to Sect. 3) for the division of cloudlets in both of the clusters. The estimated finish time (T_{est}) is calculated for all the cloudlets and compared to the deadline if T_{est} is greater than the deadline, the cloudlet is sent to the HERC. The proposed policy improves the makespan (MSp) of the VMs, as well as MSp of the hosts, present in the Datacenter as a whole. Utilization of resources [32] is also taken care of by this algorithm, especially the utilization of HERC as the resources are significantly lower in number than the LERC. Various researches have been undertaken, based on scheduling techniques [33] with various network scenarios and combinations of service classes. The proposed policy has been simulated in this environment and compared with five other existing allocation policies, namely Round Robin Allocation (RRA) Policy [34] which has already been implemented primarily, greedy allocation policy [35,36], Improved Round robin Algorithm (IRRA) [37], Opportunistic Load Balancing (OLB) [38], and Minimum Completion Time(MCT) [39]. The major drawback of these algorithms is larger makespan of the VMs as well as of the hosts present in the Datacenter. The proposed cloudlet allocation policy provides better results than the aforementioned policies. A detailed discussion is done in Sect. 3.

1.2 Organization

The rest of the paper is organized as follows: Sect. 2 describes the different related work corresponding to cloudlet allocation policies. Section 3 describes the proposed work including the detailed algorithm and flowcharts. Section 4 describes the experimental results of proposed algorithm. Section 5 presents the comparison results and analysis to illustrate the prominence of this proposed policy over some existing algorithms. Finally, Sect. 7 has the conclusion and discussion of future scope of the proposed work.

2 Related Work

Cloud computing is growing rapidly throughout the world and has become an industrial beast economically. The cloud services are paid, fully real-time and non-intermittent. Therefore, so many parameters must be taken care of, like processing speed, resource utilization and above all a good cloudlet distribution. Many algorithms have been introduced for the distribution of cloudlets, and many simulators have been created that simulate the cloud-like scenarios with a proper algorithm.

Many simulating software [40] like Gridsim, Sim-Grid [41] and Cloudsim [42] are free softwares available in the Internet to implement an algorithm in a cloud-like environment. There is a potential difference between CloudSim and Grid simulator. Grid Simulators can simulate over a distributed environment with a great drawback of indistinguishable multilevel cloud architecture (IaaS, PaaS and SaaS). In contrast, Cloudsim provides the facility to use and customize all dimensions of cloud-like environment.

Cloudsim provides many scheduling [43–45], allocation and datacenter broker policies that make the simulation real-

istic and substantial w.r.t. actual cloud simulation [41,46]. There are various allocation policies to allocate the VMs [47] to a particular host as per capacity (discussed in Sect. 3), scheduling policies are ways to schedule VMs in each host and cloudlets in each VM, and these policies can be modified by the researchers as per their needs. Datacenter broker policies give a way to allocate cloudlets to a particular VM, and Cloudsim comes with default policies but also facilitates the power of developments by the scientists.

CloudSim is a java-based software which has various entities like Datacenters, Hosts, VMs (Virtual machines), Cloudlets, two types of processing queue named Deferred Queue and Future Queue and various services and functionalities like cloudlet allocation policies, cloudlet migration policies, various scheduling policies like Time shared or Space Shared Scheduling, VM allocation policies, VM migration policies [46]. Description of some of the main entities and functionalities are given below.

Datacenters Datacenters are the top most entities in the hierarchical order of host and VM. Datacenters are the pool of resources comprising various hardware resources and network infrastructures.

Hosts Hosts are the actual physical machines running in the Datacenter. Hosts may have one or more than one VMs. Host itself schedule VMs by VM scheduling policies.

Virtual Machines (VMs) Virtual machines run in the host by sharing the resources of host. VMs schedule cloudlets by various cloudlet allocation policies. Cloudlets run in the VMs according to scheduling policies implemented internally by CloudSim. The unit of the processing speed of VMs is MIPS (million instruction per second).

Cloudlets Cloudlets [46,48–53] are the units to be processed in CloudSim. Each cloudlet consists of many computational instructions which are measured in MI (Million Instruction Units). Cloudlets have input and output sizes, and output size may differ from the input size. Cloudlets are scheduled by VMs. Default scheduling policy is Round Robin scheduling.

Future Queue and Future Events CloudSim entities are stored in the queue as events at run time, so it is called future events. These events are sorted on their time parameter during insertion in the queue.

Deferred Queue The future events scheduled at each step of the simulation are removed from the future events queue and transferred to the deferred event queue.

VM Allocation Policy VM allocation policy is purely an intrinsic property of the host itself. Host imposes allocation policies of VMs with one of the two policies Time shared or Space shared policy.

Cloudlet Allocation Policy Cloudlet allocation policy is determined by VMs. VM not needed allocates the cloudlets by one of these two allocation policies, i.e., Time shared policy or Space shared policy.

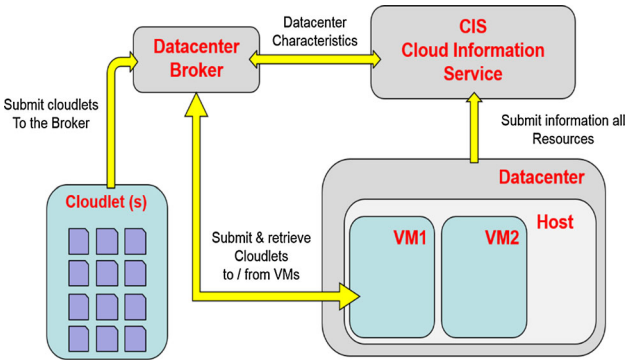

Fig. 1 Cloudsim architecture

Table 1 Round Robin VM allocation

Virtual machines	Cloudlets
VM_1	C_0
VM_0	C_1
VM_3	C_2
VM_2	C_3

In Fig. 1, a complete architecture of Cloudsim is shown. Cloudlets are submitted to the datacenter broker, and datacenter broker decides the allocation of cloudlets to VMs. Hosts schedule VMs, and VMs in turn schedules cloudlets within itself. CIS keeps all the information about entities and run time environment throughout the simulation.

2.1 Related Allocation Policies

In this paper, two existing algorithms have been compared with the proposed algorithm. Designing a good allocation policy is a distinguishable challenge in the cloud computing domain.

2.1.1 Round Robin Allocation (RRA) Policy [34]

In Cloudsim, cloudlets are executed in the VMs by RRA policy in a time shared manner. Set of cloudlets are submitted to the broker, and broker sends each cloudlets to the first available VM. If there are four VMs (VM_0, VM_1, VM_2, VM_3) and four cloudlets (C_0, C_1, C_2, C_3), then Table 1 shows the allocation of VMs for the cloudlets.

2.1.2 Greedy Allocation Policy [35]

For a set of cloudlets and the virtual machines, greedy-based algorithm depends on the local optimal method to allocate resources. That is the reason why we call it greedy algorithm. The scheduling is greedy upon the time preference cloudlet or bandwidth preference cloudlet. Greedy alloca-

tion scheduling algorithm mainly consists of three phrases, including classification based on QoS Cloudlet classifier.

- *Cloudlet Classification* Cloudlet classifier classifies the cloudlet based on various parameters, determined by preprocessing unit. For example, cloudlet can be classified into different types based on QoS (e.g., completion time, bandwidth). Then, it sends cloudlets to different VMs that are programmed with different scheduling algorithm.
- *Justice Evaluation Function* (*JEF*) This is defined by two parameters, actual reward (*AR*) [3] and expectation reward (*JF*), *JEF* is calculated using Eq. 1

$$JEF = theta\left(\frac{AR}{JF}\right) \tag{1}$$

- For time type preference, resource instances are chosen by calculating least execution time for a cloudlet (TP_{i}), according to Eq. 2

$$TP_i = \frac{TR_i}{TR_{max} - TR_{min} + 1} \tag{2}$$

TR_{i} is the finish time of ith cloudlet, TR_{max} is the max finish time, and TR_{min} is the minimum finish time.
- The bandwidth preference emphasizes the importance of choosing resource instances that provide the appropriate bandwidth for cloudlets, i.e., BP_{i}, BP_i is calculated in Eq. 3

$$BP_i = \frac{BR_i}{BR_{max} - BR_{min} + 1} \tag{3}$$

BR_{i} is the number of cloudlets preferring bandwidth i, BR_{max} bandwidth taken by maximum demand cloudlet, and BR_{min} bandwidth taken by minimum demand cloudlet.
- Makespan (MSp) of a cloudlet is calculated in Eq. 4

$$MSp = T_{iFin} - T_{iStart} \tag{4}$$

T_{iFin} is the finish time of the ith cloudlet, T_{iStart} is the starting time of the ith cloudlet

2.1.3 Improved Round Robin Algorithm (IRRA) [37]

IRRA for datacenter selection is an algorithm to select the datacenters in a way that minimizes cloud service response time as observed at the client site. In Cloudsim environment, the datacenter broker submits the cloudlets to the VM of a datacenter. The submission will be done by the datacenter broker according to faster service response time per IRRA. If there are N datacenters ($DC_1, DC_2, \cdots, DC_N$) each having $\mathrm{n_m}$ number of VMs, respectively, then a datacenter is will be chosen according to its distance from the user and the number of jobs that are currently executing in the datacenter.

As per IRRA, the algorithm works in the following way:

- The earliest reachable datacenter is being selected first to reduce the communication time.
- If there is a single datacenter to satisfy the upcoming requests, then the datacenter will be selected for execution of the jobs.
- If there are multiple datacenters at the nearest region, then the datacenters will be selected in a sequential order from the datacenters of the nearest region.

2.1.4 Opportunistic Load Balancing (OLB) [38]

OLB algorithm is used to keep each node busy without considering the present workload. Free order OLB assigns tasks to the nodes to make all nodes as useful as possible. If there are N numbers of VMs from various datacenters, OLB algorithm will assign tasks to all the VMs without calculating the capacity or present workloads of the VMs. Net completion time is very poor as OLB does not consider the execution time for the cloudlets.

2.1.5 Minimum Completion Time (MCT) [39]

MCT algorithm calculates the estimated completion time (ECT) of a cloudlet from a batch of cloudlets and assigns the cloudlet to the VMs that meet the deadline. The algorithm uses a subroutine called Estimate start that computes the latest completion time of a task from a set of tasks and decides when the VM will be ready to be assigned with a new task. The completion time of the task is calculated by adding up stage in and stage out time with the actual completion time of the task.

These existing procedures mentioned in this section are unable to find out the minimum makespan compared to the proposed algorithm, which finds out the smallest makespan among these algorithms. The experimental results obtained after applying the proposed allocation policy are given in Sect. 5.

3 Proposed Work

The proposed work defines a strategy to reduce the completion time of a batch of cloudlets by diverging them with respect to a deadline incorporated by cloud management itself. The main objective is to provide better QoS by increasing performance than other cloudlet allocation strategies like greedy allocation or RRA, by reducing the overall makespan of the batch of cloudlets.

3.1 Resource Allocation Policy in Cloud

In the proposed work, the cloudlets are to be allocated to two types of resource clusters [54]:

1. Low-end resource cluster (LERC)
2. High-end resource cluster (HERC)

In contrast, the high-end resources are comprised of larger main memory, higher processing speed in terms of MIPS with more number of powerful processing elements (PEs) or cores but smaller in scale than the LERC. The LERC can be compared with the average cloud resources that uses economically now-a-day resources, and the HERC is nothing but the advanced high-capacity cloud resources that are used for large, complex applications. Each datacenter consists of hosts. VMs are created, provisioned in hosts where the cloudlets execute. All cloudlet information is primarily stored in a list called $cloudlet\_list$, and all VM information is stored in a list named vm_list.

Initially all the cloudlets are allocated to the LERC, and based on different situation, the cloudlet(s) may be migrated to the HERC using various parameters (Sect. 3.2) to reduce the overall makespan in terms of providing better QoS.

In this case, we compute the total processing capacity of a Cloud host as shown in Eq. 5.

$$CapH = \frac{\sum_{i=1}^{i=totPEs} Capacity(i)}{MAX\left(\sum_{j=1}^{j=Ncld} Corereq(j), totPEs\right)} \tag{5}$$

- $CapH$ = capacity of a host.
- $Capacity\ (i)$ = processing strength of ith processing element, i.e., PE.
- $Corereq\ (j)$ = required number of cores for jth cloudlet
- $totPEs$ = total number of processing elements.
- $Ncld$ = total number of cloudlets.

3.2 Required Parameters

3.2.1 Execution Time for a Single Cloudlet $T_{exe}(i)$ From a Batch of Cloudlets

Execution time of each cloudlet will be determined by a sequential access of all the cloudlets that are submitted to be processed by the formula given in Eq. 6.

$$T_{exe}(i) = \frac{L_{cld}(i)}{PS(i,j) * Core_{cld}(i,j)} \tag{6}$$

- $T_{exe}(i)$ = execution time of ith cloudlet in single execution.
- $L_{cld}(i)$ = total length of ith cloudlet (MI), MI = Million Instructions.
- $PS(i, j)$ = processing speed of the processor of jth VM, in which ith cloudlet has been allocated.
- $Core_{cld}(i, j)$ = no. of cores in the processor of jth VM allocated for ith cloudlet.

3.2.2 Average Execution Time (T_{avg})

T_{avg} is the average execution time of all cloudlets. This parameter is directly related to the deadline calculation discussed in 3.2.4. This is a simple average calculated by the formula given in Eq. 7.

$$T_{avg} = \frac{\sum_{i=1}^{i=N_{cld}} T_{exe}(i)}{N_{cld}} \tag{7}$$

3.2.3 Threshold Percentage Th_p

Th_p is a very important parameter in the proposed algorithm. The idea is to split up and provision the cloudlets to HERC and LERC. But due to cost constraint, there must be some restriction on cloudlets to be migrated to the HERC so that HERC will never execute cloudlets with a higher makespan (discussed in Sect. 4.1) than the makespan of LERC. So the constraint Th_p is used here to set up the number of cloudlets that will be executed at the HERC.

This Th_p solely depends on the configuration of both clusters. It is the task of the cloudlet of administrator to set up the Th_p according to the configuration of the clusters, for full utilization model [55] one can use the method shown in equation 8 to set up the Th_p value Eq. 8.

$$Thp = \frac{H_{vmL} * Avg(M_L) * Avg(PE_L) * Avg(MIPS_L)}{H_{vmH} * Avg(M_H) * Avg(PE_H) * Avg(MIPS_H)} \tag{8}$$

- H_{vmH} = number of hosts at HERC
- H_{vmL} = number of hosts at LERC
- $Avg(M_L)$ = average main memory using by all VMs at LERC
- $Avg(M_H)$ = average main memory using by all VMs at HERC
- $Avg(PE_L)$ = average shared processing elements using by all VMs at LERC
- $Avg(PE_H)$ = average shared processing elements using by all VMs at HERC
- $Avg(MIPS_L)$ = average processing speed using by all VMs at LERC
- $Avg(MIPS_H)$ = average processing speed using by all VMs at HERC

The practical simulation and setup of the Thp is shown at the Sect. 5.

3.2.4 Deadline T_d) for a Batch of Cloudlets

The deadline is determined by using the T_{avg}. Though the HERC has more powerful resources, resources are lesser in quantity than the LERC; it is obvious that HERC will be degraded in terms of performance if the number of cloudlets crosses a threshold value. The deadline can be determined as shown in Eq. 9.

$$T_d = T_{avg} * N_{cld} * Th_p \tag{9}$$

- T_d = deadline for a batch of cloudlets.
- Th_p = threshold percent.
- N_{cld} = total Number of cloudlets.

Th_{p} solely depends upon a comparative calculation between the resources that are being used by HERC and LERC. Th_p is the factor that decides the proper workload on the HERC side to improve the overall performance (discussed in Sect. 5).

3.2.5 Estimated Finish Time (T_{est})

Estimated finish time of a cloudlet is an estimation of completion time of a cloudlet while executing. The estimated finish time will be calculated at each pass of cloudlet scheduling. This time is calculated as shown in Eqs. 10 and 11.

$$T_{est}(i) = T_{est}(i-1) + \frac{LR_{cld}(i,j)}{PS(i,j) * Core_{cld}(i,j)}, \quad When\ i > 1 \tag{10}$$

$$T_{est}(1) = \frac{LR_{cld}(1,j)}{PS(1,j) * Core_{cld}(1,j)}, When\ i = 1 \tag{11}$$

- $LR_{cld}(i,j)$ = length remaining of ith cloudlet after jth pass.

LR_{cld} is calculated as shown in Eq. 12

$$LR_{cld}(i,j) = L_{cldB}(i,j) - L_{cldP}(i,j) \tag{12}$$

- $L_{cldB}(i,j)$ = length of ith cloudlet before jth pass.
- $L_{cldP}(i,j)$ = portion of cloudlet that is executed after jth pass.

3.2.6 Makespan (MSp)

Makespan of a cloudlet is the completion time of a cloudlet. Makespan of a VM is the maximum makespan of a cloudlet among all cloudlets executed in that VM. Makespan of a host can be considered as the highest makespan of a VM among all VMs running in that host. The proposed algorithm is dealing with two MS_{p}, named MS_{pH} and MS_{pL}, makespan of HERC and makespan of LERC, respectively.

In the proposed algorithm, the Thp has been introduced in such a way that HERC will never be overloaded with cloudlets. In other terms, the Th_{p} has been set up so that makespan of HERC never exceeds the makespan of LERC (refer to Fig. 6). The effective makespan will be considered as the highest makespan of cloudlet at the LERC.

3.2.7 Round Trip Time (RTT)

Round trip time is the length of time it takes for a signal to be sent in addition to the length of time it takes for an acknowledgment of that signal to be received. This time delay therefore consists of the propagation times between the two points of a signal. In case of cloudlet migration from LERC to HERC, the round trip time required is shown in Eq. 13

$$RTT_i = T_x + 2 * T_p \tag{13}$$

- T_x = transmission delay can be calculated using Eq. 14.

$$T_x = \frac{L_{cld}}{BW} \tag{14}$$

- BW = bandwidth of transmitting channel
- T_p = propagation delay.

$$T_p = \frac{D_{hl}}{V_{hl}} \tag{15}$$

- D_{hl} = distance between high end and low end.
- Vhl = velocity of signal propagation between HERC and LERC through transmitting channel.

3.2.8 Completion Time (T_{com})

Completion time T_{com} of a cloudlet is the time difference between completion of execution of a cloudlet and submission of cloudlet to the processing queue in a VM. Completion time is the total time of execution of a cloudlet including waiting time of the cloudlet in the processing queue.

$$T_{\mathrm{com}} = (Execution\ time + Waiting\ time) of\ a\ cloudlet \tag{16}$$

$\sum T_{\mathrm{com}}$ is the summation of completion times of all cloudlets present in a batch of cloudlets. This parameter is used to measure the quality of performance for the proposed algorithm.

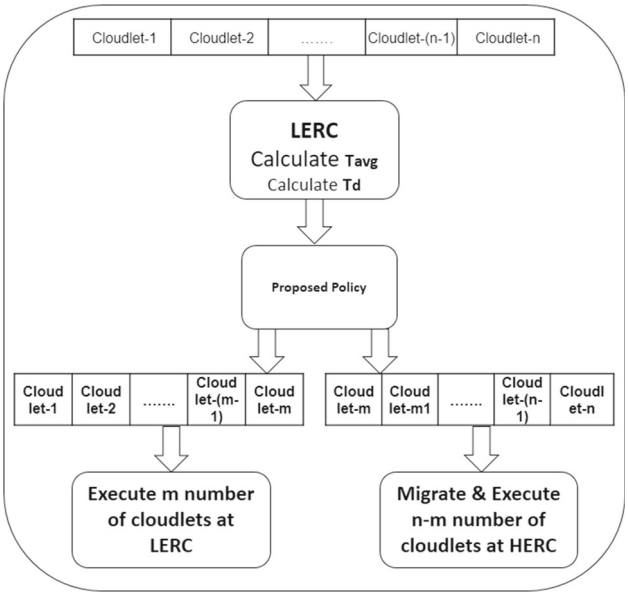

Fig. 2 Pictorial representation of proposed policy

3.3 Pictorial Representation of Proposed Policy

In the proposed model, two types of resource clusters are available in the cloud system, i.e., the low-end resource cluster and high-end resource cluster. In comparison with LERC, HERC consists of high-end computing resources. On the other hand, LERC is cost-effective. Combining LERC and HERC yields optimized outcome of the entire service provider. So, we need low-end resources for providing cloud users a trade-off between cost and performance. The proposed allocation policy is based on this simple theory. Suppose at any instance of time there is a batch of cloudlet having n number of cloudlets, waiting to be served by the cloud system. Initially the batch of cloudlet will be submitted to the LERC. Then, the deadline (T_d) of the batch of cloudlet is to be found out using the average execution time (T_{avg}) of each cloudlet i. At this stage, our proposed allocation policy is used to determine the cloudlets that are needed to be migrated to and executed at HERC so that the makespan (MSp) can be minimized. Suppose $(n-m)$ number of cloudlets are selected for migration. Then, m number of cloudlets will be executed at LERC and other $(m-n)$ will be executed at HERC. The entire procedure is depicted in Fig. 2. The policy starts submitting the set of cloudlets following the above HLC policy to different sets of resource clusters.

3.4 High-end–Low-end Cluster (HLC) Scheduling Algorithm

The High-end–Low-end Cluster (HLC) Scheduling Algorithm is stated as follows.

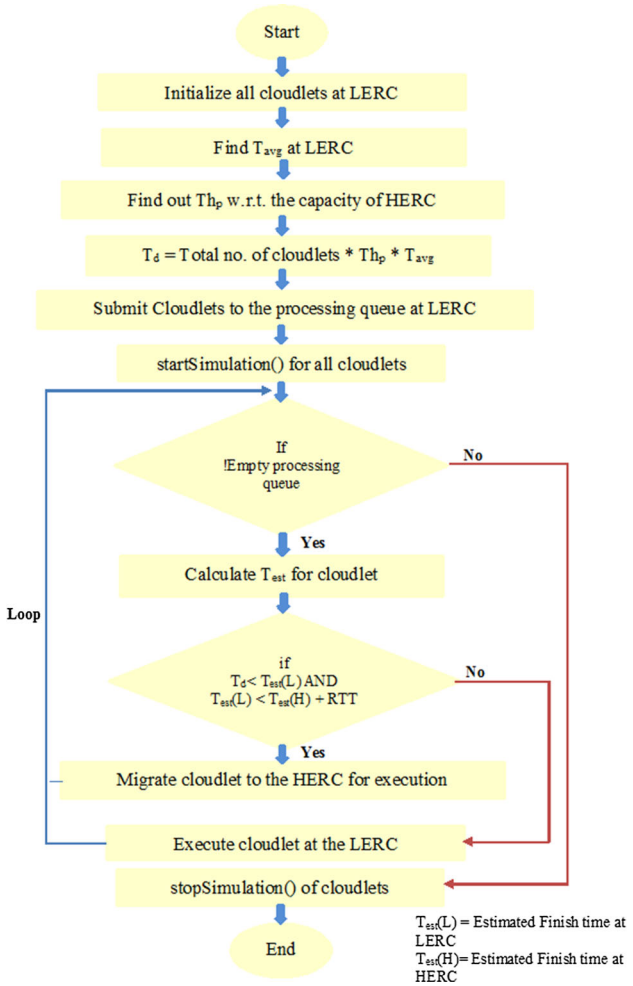

Fig. 3 Flowchart for HLC scheduling algorithm

Algorithm 1: High-end–Low-end Cluster (HLC) Scheduling Algorithm

Data: S = Set of all cloudlets at LERC
R = Set of all cloudlets at HERC, initially {∅}

```
initialize(S);
Calcualte T_avg of cloudlets present in S.;
T_d = T_avg * N * Th_p;
startSimulation();
for all cloutlet j, j ∈ S & size(s) > 0 do
    EFTL = Estimated Finish Time (j) at LERC;
    if EFTL > T_d & EFTL < Estimated Finish Time(j) at HERC + RTT_j then
        Migrate Cloudlet j to the HERC;
        S = S − {j};
        R = R ∪ {j};
    end
end
stopSimulation();
End;
```

Table 2 Host in LERC

Host id	RAM (MB)	Number of PEs
0	4096	4
1	4096	4
2	4096	4

Table 3 Host in HERC

Host id	RAM (MB)	Number of PEs
0	32768	16

Table 4 VM in LERC

VM id	Shared RAM (MB)	Shared	Number of PEs MIPS
0	2048	2	778
1	2048	2	610
2	2048	2	503
3	2048	2	750
4	2048	2	608
5	2048	2	690

Table 5 VM in HERC

VM id	Shared RAM (MB)	Shared	Number of PEs MIPS
0	8192	4	1869

3.5 Flowchart for HLC Scheduling Algorithm

Figure 3 shows a flowchart of HLC Scheduling Algorithm.

4 Experimental Result

The proposed work and algorithm have been described and analyzed with suitable example. This strategy has mainly two parts of execution; primary execution will be done in the LERC, and depending upon the deadline, the remaining cloudlets will be allocated to the HERC.

Tables 2 and 3 show the hosts along with corresponding processing speed in terms of MIPS (million instructions per second), and total RAM that will be shared by VMs that will be allocated to the host.

Tables 4 and 5 show the VMs and corresponding processing speed in terms of MIPS, and RAM shared by the VMs.

Table 6 shows the cloudlets and the size of the cloudlets.

Here, the LERC consists of six VMs; each VM has 2 GB of shared RAM and has a processing speed (MIPS) produced by a randomized function. In contrast, the HERC consists of one VM that has 8 GB of shared RAM, VM has a processing speed (MIPS), higher than the LERC produced by a randomized function. For the practical experiment, we have taken one lakh cloudlets, but due to space constraint we have given the experimental result with total twenty-four cloudlets.

Table 6 shows the log of cloudlets. This is a log prior to the submission of the cloudlets to the processing queue. In this phase, single execution time (T_{exc}) of a cloudlet is calculated and with the help of average execution time T_{avg} (Table 7), the deadline for the batch of the cloudlets is calculated using Eq. 7. Each cloudlet is allocated to a VM; selection of VMs is a completely dynamic process managed by Cloudsim itself.

$T_{avg} = 13.971$ s Now, the deadline T_d for the batch will be calculated, by using T_{avg} and threshold percentage Th_p. Th_p solely depends on the resources of the clusters. Th_p is calculated as 75%, shown in Sect. 4.1, so that any number of cloudlets is used makespan of HERC will not be overloaded.

$T_d = (13.971 * 24 * 0.75) = 251.478$ s (using Eq. 9)

After calculation of the deadline with the help of equation 9, the cloudlets are submitted for execution in the LERC; the estimated finish time T_{est} is calculated as Eq. 9 and based on the comparison between T_{est} and T_d, the cloudlets are executed at LERC and HERC.

As per the proposed algorithm, the cloudlets are submitted for execution serially as Table 6 and during the first pass some cloudlets are sent to the HERC as deadline has exceeded for those cloudlets. Tables 8 and 9 show the log of cloudlets that are executed in the HERC and the LERC.

During the first pass at the LERC, in the phase of estimated finish time calculation T_{est} of 17th cloudlet was 250.803 s it was executed at LERC as shown in Table 6, next to that cloudlet T_{est} of 18th cloudlet was

$$\begin{aligned} T_{est}L(18) &= T_{est}L(17) + \textit{Execution time of remaining length of } 18\text{th } \textit{cloudlet} \\ &= 250.803 + 11.043 = 261.846\,\text{s} \end{aligned}$$

From Eq. 13,

$$\begin{aligned} \textit{RTT of } 18\text{th } \textit{cloudlet} &= Tx + 2Tp \\ &= 0.00001 + 2 * 0.0003 \\ (Bw &= 10\ Gbps) \\ &= 0.00061\,\text{s} \end{aligned}$$

$T_{est}H(18) = 11.043$ (*As no cloudlet is initiated at HERC*)

As the proposed algorithm,

$T_{est}L(18) > Td\ And,$

$T_{est}L(18) < T_{est}H(18) + RTT\ of\ 18^{th}\ cloudlet$

Table 6 Cloudlets log before execution at LERC

Cloudlet id	Size of cloudlet (MI)	Number of Pes for execution	VM id	T_{exc}
0	17,356	2	3	11.57
1	18,432	2	0	11.845
2	17,354	2	0	11.152
3	19,008	2	3	12.672
4	17,264	2	3	11.509
5	18,448	2	0	11.586
6	19,988	2	5	14.484
7	16,640	2	5	12.057
8	18,066	2	5	13.091
9	16,432	2	1	13.458
10	19,744	2	4	16.23
11	19,972	2	4	16.424
12	17,264	2	1	14.15
13	19,840	2	4	16.315
14	16,746	2	1	13.726
15	17,024	2	2	16.992
16	16,752	2	2	16.652
17	16,992	2	2	16.89
18	17,184	2	0	11.043
19	19,600	2	1	16.065
20	16,940	2	2	16.838
21	18,722	2	3	12.481
22	18,472	2	4	15.19
23	17,780	2	5	12.884

Table 7 Average execution time for batch of cloudlets

Cloudlet id	Number of cloudlets	$\sum T_{com}$	T_{avg}
0–23	24	335.304	13.971

Table 8 Cloudlets log after execution at HERC

Cloudlet id	Status	VM id at HERC	T_{com}
20	SUCCESS	0	27.29
18	SUCCESS	0	27.62
23	SUCCESS	0	28.25
22	SUCCESS	0	28.81
21	SUCCESS	0	28.94
19	SUCCESS	0	29.18

Therefore, the cloudlet is sent to the HERC for execution, so as the next cloudlets also were sent to the HERC for execution as the log of execution is shown in Table 8.

Table 9 shows the completion log of the cloudlets at the LERC. All cloudlet logs in the both Tables 8 and 9 are arranged by the completion time or total execution time.

Table 9 Cloudlets log after execution at LERC

Cloudlet id	Status	VM id at LERC	T_{com}
9	SUCCESS	3	32.86
12	SUCCESS	0	33.28
0	SUCCESS	0	33.4
15	SUCCESS	3	33.65
3	SUCCESS	3	34.97
6	SUCCESS	0	35.09
17	SUCCESS	5	36.94
5	SUCCESS	5	39.05
11	SUCCESS	5	40.15
7	SUCCESS	1	40.92
16	SUCCESS	4	41.33
4	SUCCESS	4	42.17
1	SUCCESS	1	43.85
10	SUCCESS	4	44.21
13	SUCCESS	1	45.01
14	SUCCESS	2	49.94
2	SUCCESS	2	51.14
8	SUCCESS	2	51.85

Table 10 Makespan at LERC

Cloudlet id	VM id LERC	MSp
6	0	35.09
13	1	45.01
8	2	51.85 (MAX)
3	3	34.97
10	4	44.21
11	5	40.15

Table 11 Makespan at HERC

Cloudlet id	VM id LERC	MSp
19	0	29.18 (MAX)

For comparing the results easily, the makespan of cloudlets for the individual VMs is shown in Tables 10 and 11; from those the maximum makespan will be found out.

Both tables show the highest completion time of a cloudlet corresponding to the VM. In both of the makespan tables, the maximum makespan has been stroked out. The maximum makespan shows the time of completion of a batch of cloudlet. Allocation of all cloudlets in VMs and VMs in hosts has been taken care of by Cloudsim itself. Therefore, the experimental result of the execution of all 24 cloudlets at the LERC shows a fair contrast between the normal allocation and allocation based on the proposed algorithm is shown.

All 24 cloudlets here are executed at the LERC, i.e., two hosts and five VMs will execute all the cloudlets. In the Tables 12 and 13, Cloudlet execution log and makespan are shown, respectively.

As the results show that among all twenty-four cloudlets running at LERC the maximum makespan is 68.79185 s. While in the proposed algorithm maximum makespan at HERC and LERC is 29.18 and 51.85 s, respectively.

Experimental result of the proposed algorithm on makespan of different VMs of HERC and LERC are shown with a help of a graph in Fig. 4.

The makespans of the proposed algorithm show the end time of processing of cloudlets. Though there are two clusters: highest makespan of LERC is 51.85 s and highest makespan of HERC is 29.18 s, the effective makespan of those two clusters, as discussed in Sect. 3.2.6, will be makespan of LERC, i.e., 51.85 s.

4.1 Analyzing the Th_p

As it is discussed earlier at Sect. 4 that Thp is an important parameter in the proposed algorithm, for a large-scale simulation scenario the data and experimental result are given, a full utilization model is created in Cloudsim environment,

Table 12 All cloudlets log of execution at LERC

Cloudlet id	Status	VM id at LERC	T_{com}
9	SUCCESS	3	43.82
18	SUCCESS	0	44.17
12	SUCCESS	0	44.33
0	SUCCESS	0	44.45
15	SUCCESS	3	45
6	SUCCESS	0	46.14
21	SUCCESS	3	47.27
3	SUCCESS	3	47.46
17	SUCCESS	5	49.25
23	SUCCESS	5	50.96
5	SUCCESS	5	51.93
11	SUCCESS	5	53.04
7	SUCCESS	1	54.56
16	SUCCESS	4	55.1
4	SUCCESS	4	56.37
22	SUCCESS	4	58.35
1	SUCCESS	1	58.96
10	SUCCESS	4	59.4
19	SUCCESS	1	60.88
13	SUCCESS	1	61.07
14	SUCCESS	2	66.58
20	SUCCESS	2	67.16
2	SUCCESS	2	67.98
8	SUCCESS	2	68.69

Table 13 Makespan at LERC (all cloudlets at LERC)

Cloudlet id	VM id at LERC	MSp
6	0	46.14
13	1	61.07
8	2	68.69 (MAX)
3	3	47.46
10	4	59.4
11	5	53.04

and two clusters are created, namely LERC and HERC with hosts and VMs given in Tables 14 and 15 as shown; the parameters are discussed in Sect. 4.

4.1.1 Calculating the TH_p

Form Eq. 8,

$$Th_p = \frac{H_{vmL} * Avg(M_L) * Avg(PE_L) * Avg(MIPS_L)}{H_{vmH} * Avg(M_H) * Avg(PE_H) * Avg(MIPS_H)}$$
$$= 74.40\%$$
$$\approx 75\%.$$

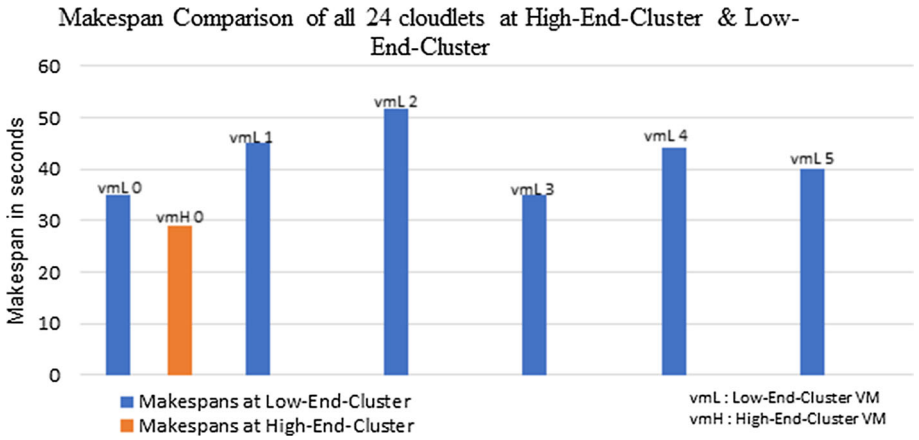

Fig. 4 MS_p of VMs applying proposed algorithm allocation

Table 14 Parameters for LERC to calculate Th_p

HvmL	Number of VMs	Avg (ML)	Avg (PEL)	Avg (MIPSL)
2500	5000	2	2	650

Table 15 Parameters for HERC to calculate Th_p

HvmL	Number of VMs	Avg (ML)	Avg (PEL)	Avg (MIPSL)
140	540	8	4	1950

For this full utilization model, the Th_p is taken 0.75, from the experimental results those are shown here do not properly fit full utilization model that is why everywhere 0.75 is taken as a standard Th_p. With this Thp, the simulation has been done with 10, 50, 100 and 200 k cloudlets and the results are shown in Fig. 5.

From Fig. 5, it is clear that selection of right Th_p distributes the cloudlets to the clusters in such a way that cloudlets at the HERC are executed by the resources in very precise way to make the makespan of the HERC a congruity to the makespan of LERC.

5 Comparison and Analysis

In Cloudsim, the evaluation of the proposed algorithm is done with different cloudlet size and different processing speeds of VMs. The evaluation of the proposed algorithm is done using the execution time (T_{exc}) and makespan (MS_p) discussed in Sect. 3. The proposed algorithm is compared with two pre-existing approaches: Round Robin approach and greedy approach.

In Round Robin approach, each cloudlet gets a fair chance to acquire the VMs, and datacenter broker allocates VMs in RRA policy.

The greedy approach considers the resources from the LERC solely on their performance. Greedy-based algorithm depends on the local optimal method to allocate resources. This allocation may base on two types of cloudlets, namely time type Cloudlet and BW type Cloudlet.

The proposed algorithm allocates VMs from HERC and LERC with help of various methods discussed earlier.

5.1 Simulation Configurations and Scenarios

The simulation environment was set up with Datacenters as HERC and LERC.

The LERC had two hosts and six VMs. VMs were allocated to hosts dynamically by Datacenter Broker. Each VM on an average shares two processors and on average has 2 GB shared RAM, and MIPS was allocated with a random function rand(500, 800), which generates MIPS between 500 and 800, randomly.

The HERC had one host and one VM. VM was allocated to host dynamically by Datacenter Broker. VM shares four processors and 8 GB RAM, and MIPS was allocated with a random function rand (1500, 2400), which generates MIPS between 1500 and 2400, randomly.

For the comparison and analysis, 10, 20, 30, 40 Cloudlets were generated with different MIs (Million Instructions), respectively, by the help of a randomized function rand (5000, 6000), which creates cloudlets of MIs between 5000 and 6000 randomly.

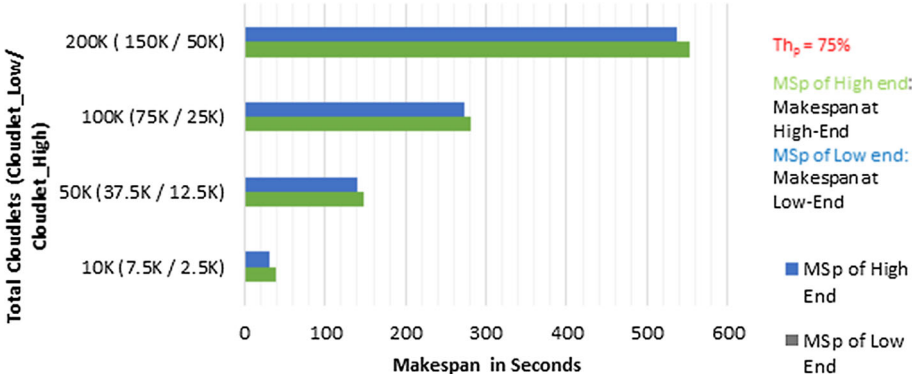

Fig. 5 Validation of threshold percent

Table 16 Comparison with 10 cloudlets between existing and proposed algorithms

Fig.	Algorithm	Cloudlet id	T_{com}	$\sum T_{com}$(10 cloudlets)			(%) of Improvement on T_{com} than	
				Proposed	RRA	Greedy	RRA	Greedy
6	Proposed	1	8.61	66.12	96.37	94.91	31.38	29.93
	RRA		8.66					
	Greedy		12.48					
	Proposed	4	8.72					
	RRA		8.66					
	Greedy		4.57					
	Proposed	6	8.26					
	RRA		10.93					
	Greedy		9.97					
	Proposed	9	1.78					
	RRA		10.07					
	Greedy		10.14					

In Round Robin approach, all cloudlets are allocated to the LERC, and allocation is done according to normal RRA policy.

The greedy approach requires the resources of LERC for executions of cloudlets according to performance basis, but as it does not concentrate on cost, it results in a huge drawback in greedy approach.

In a contrast, due to the threshold percent, described in Sect. 3.2.3, value the Cloudlets can be distributed into two clusters by the proposed algorithm the execution time can be reduced than the greedy algorithm.

5.2 Comparison on Completion Time

The tables (Tables 16, 17, 18, 19, 20, 21, 22, 23) compare the completion time between the proposed algorithm and existing algorithms. Due to space constraint, only four cloudlets among a batch of cloudlets are selected randomly and the completion time of the cloudlets is shown according to the applied algorithm in second third and fourth column. Fifth column shows the $\sum T_{com}$ for the batch of cloudlets, and the sixth column depicts the improvement of our algorithm than the two existing algorithms in '%'.

5.2.1 Comparison with 10 Cloudlets on Execution Time

Ten cloudlets were used to compare three algorithms. In Round Robin, all the 10 cloudlets were sent to the LERC and all cloudlets executed. In the greedy algorithm, the allocation was done dynamically as explained earlier and proposed algorithm splits the cloudlets. At the LERC, three VMs were used and at the HERC one VM was used. Improvement of execution time of our proposed to the Round Robin policy and greedy algorithm is 31.38 and 29.93%, respectively. The results are shown in Fig. 6 and Table 16.

Though all experimental values have been used to calculate the improvements, due to the space constraint few results are shown here for comparison.

As in Table 17 the execution time of 1, 4, 6, 9 is shown according to proposed algorithm, RRA (Round Robin Algorithm) and greedy algorithm, total execution time is 66.12, 96.37 and 94.91, respectively, as per algorithms, Fig. 4 shows

Table 17 Full comparison log with 10 cloudlets between existing and proposed algorithms

Cloudlet id	T_{com} Proposed Algo	T_{com} RRA	T_{com} Greedy Algo
0	8.39	11.07	10.59
1	8.61	8.66	12.48
2	5.99	9.13	9.56
3	8.26	10.93	11.03
4	8.72	8.66	4.57
5	5.7	8.55	8.43
6	8.26	10.93	9.97
7	8.5	8.42	9.07
8	1.89	9.32	9.07
9	1.78	10.7	10.14
$\sum T_{com}$	66.12	96.37	94.91

Table 18 Comparison with 20 cloudlets between existing and proposed algorithms

Fig.	Algorithm	Cloudlet id	T_{com}	$\sum T_{com}$ (10 cloudlets)			(%) of Improvement on T_{com} than	
				Proposed	RRA	Greedy	RRA	Greedy
7	Proposed	0	13.75	239.72	384.77	356.72	37.69	32.79
	RRA		19.51					
	Greedy		19.3					
	Proposed	7	14.38					
	RRA		20.17					
	Greedy		17.92					
	Proposed	14	16.13					
	RRA		19.29					
	Greedy		19.09					
	Proposed	9	4.67					
	RRA		18.96					
	Greedy		18.77					

execution time comparisons between three algorithms in a graphical form. As per proposed algorithm cloudlet id 0–7 total 8 cloudlets are executed at LERC and two, cloudlets id 8 and 9, at the HERC.

For such a low number of cloudlets, RRA shows a better result than any other algorithm as the experimental result and Fig. 6 show it.

5.2.2 Comparison with 20 Cloudlets on Execution Time

Twenty cloudlets were used to compare three algorithms. In Round Robin, all the 20 cloudlets were sent to the LERC and all cloudlets executed. In the greedy algorithm, the allocation was done dynamically as explained earlier and proposed algorithm splits the cloudlets. At the LERC, three VMs were used and at the HERC one VM was used. Improvement of execution time of our proposed to the Round Robin policy and greedy algorithm is 37.69 and 32.79%, respectively. The results are shown in Fig. 7 and Table 18.

Though all experimental values have been used to calculate the improvements, due to the space constraint few results are shown here for comparison in Table 19.

As in Table 19 the execution time of 0, 7, 14, 18 is shown according to proposed algorithm, RRA (Round Robin Algorithm) and greedy algorithm, total execution time is 239.72, 384.77 and 356.72, respectively, as per algorithms, Fig. 7 shows execution time comparisons between three algorithms in a graphical form. As per proposed algorithm cloudlet id 0–14 total 15 cloudlets are executed at LERC and 5 cloudlets id 15–19 at the HERC. As it is seen, Greedy algorithm improves with the increase in number of cloudlets. By comparing experimental results and Figs. 6 and 7.

5.2.3 Comparison with 30 Cloudlets on Execution Time

Thirty cloudlets were used to compare three algorithms. In Round Robin, all the 30 cloudlets were sent to the

Table 19 Full comparison log with 20 cloudlets between existing and proposed algorithms

Cloudlet id	T_{com} Proposed Algo	T_{com} RRA	T_{com} Greedy Algo
0	13.75	19.51	19.3
1	14.13	19.74	16.7
2	15.8	18.96	19
3	13.07	18.33	18.56
4	14.24	19.95	12.67
5	15.91	19.07	14.12
6	13.42	18.96	17.9
7	14.38	20.17	17.92
8	16.02	19.18	19.87
9	13.57	19.18	18.01
10	14.13	19.74	19.3
11	14.92	17.99	13.9
12	12.77	17.88	17.8
13	14.02	19.63	16.79
14	16.13	19.29	19.09
15	4.78	19.4	19.2
16	4.56	20.06	19.87
17	4.78	18.71	18.05
18	4.67	18.96	18.77
19	4.67	20.06	19.9
$\sum T_{com}$	239.72	384.77	356.72

Table 20 Comparison with 30 cloudlets between existing and proposed algorithms

Fig.	Algorithm	Cloudlet id	T_{com}	$\sum T_{com}$ (10 cloudlets)			(%) of Improvement on T_{com} than	
				Proposed	RRA	Greedy	RRA	Greedy
8	Proposed	1	22.6	541.3	846.06	731.13	36.02	25.96
	RRA		28.36					
	Greedy		27.99					
	Proposed	10	23.37					
	RRA		29.48					
	Greedy		27.97					
	Proposed	23	6.7					
	RRA		29.64					
	Greedy		22.61					
	Proposed	25	6.59					
	RRA		29.15					
	Greedy		27.01					

LERC and all cloudlets executed. In the greedy algorithm, the allocation was done dynamically as explained earlier and proposed algorithm splits the cloudlets. At the LERC, three VMs were used and at the HERC one VM was used. Improvement of execution time of our proposed to the Round Robin policy and greedy algorithm is 36.02 and 25.96%, respectively. The results are shown in Fig. 8 and Table 20.

Though all experimental values have been used to calculate the improvements, due to the space constraint few results are shown here for comparison in Table 21.

As in Table 21 the execution time of 1, 10, 23, and 25 is shown according to proposed algorithm, RRA (Round Robin Algorithm) and greedy algorithm, total execution time is 541.3, 846.06 and 731.13, respectively, as per algorithms; Fig. 6 shows execution time comparisons between three algorithms in a graphical form. As per proposed algorithm

Table 21 Full comparison log with 30 cloudlets between existing and proposed algorithms

Cloudlet id	T_{com} Proposed Algo	T_{com} RRA	T_{com} Greedy Algo
0	22.36	27.77	26.45
1	22.6	28.36	27.99
2	20.29	29.48	25.59
3	21.9	27.36	23.63
4	22.71	28.47	27.01
5	19.85	28.47	18.09
6	21.9	27.36	23.55
7	22.25	27.88	27.41
8	19.55	27.99	24.64
9	22.47	27.88	24.76
10	23.37	29.48	27.97
11	19.97	28.63	22.43
12	22.14	27.55	22.23
13	21.7	27.14	20.39
14	19.97	28.63	23.01
15	22.25	27.66	27.08
16	21.27	26.59	26.11
17	19.67	28.11	24.95
18	22.01	27.36	23.78
19	23.15	29.26	24.67
20	20.08	28.74	25.06
21	21.12	26.4	21.33
22	23.26	29.37	20.33
23	6.7	29.64	22.61
24	6.34	26.71	27.63
25	6.59	29.15	27.01
26	6.23	28.85	25.48
27	6.45	27.25	24.33
28	6.7	29.26	26.54
29	6.45	29.26	19.07
$\sum T_{com}$	541.3	846.06	731.13

cloudlet id 0–22 total 23 cloudlets are executed at LERC and 7, cloudlets id 23–29, at the HERC.

5.2.4 Comparison with 40 Cloudlets on Execution Time

Forty cloudlets were used to compare three algorithms. In Round Robin, all the 40 cloudlets were sent to the LERC and all cloudlets executed. In the greedy algorithm, the allocation was done dynamically as explained earlier and proposed algorithm splits the cloudlets. At the LERC, three VMs were used and at the HERC one VM was used. Improvement of execution time of our proposed to the Round Robin policy and greedy algorithm is 37.76 and 25.46%, respectively. The results are shown in Fig. 9 and Table 22.

Though all experimental values have been used to calculate the improvements, due to the space constraint few results are shown here for comparison in Table 23.

As in Table 23 the execution time of 0, 14, 29, 39 is shown according to proposed algorithm, RRA (Round Robin Algorithm) and greedy algorithm, total execution time is 931.31, 1496.46 and 1249.49, respectively, as per algorithms.

Figure 9 shows execution time comparisons between three algorithms in a graphical form. As per proposed algorithm cloudlet id 0–29 total 30 cloudlets are executed at LERC and 10, cloudlets id 30–39, at the HERC.

5.3 Comparison on Makespan

The cloudlet batches that are chosen for experimenting our proposed algorithm are executed in VMs in different clusters. The makespan of LERC and HERC is shown in tables differently. But the effective makespan will be the maximum makespan between the clusters as it is discussed in

Table 22 Comparison with 40 cloudlets between existing and proposed algorithms

Fig.	Algorithm	Cloudlet id	T_{com}	$\sum T_{com}$ (10 cloudlets)			(%) of Improvement on T_{com} than	
				Proposed	RRA	Greedy	RRA	Greedy
9	Proposed	0	27.49	931.31	1496.46	1249.49	37.76	25.46
	RRA		38.55					
	Greedy		34.77					
	Proposed	14	30.49					
	RRA		40.03					
	Greedy		36.56					
	Proposed	29	28.75					
	RRA		37.34					
	Greedy		29.28					
	Proposed	39	9.12					
	RRA		38.33					
	Greedy		26.24					

Table 23 Full comparison log with 40 cloudlets between existing and proposed algorithms

Cloudlet id	T_{com} Proposed Algo	T_{com} RRA	T_{com} Greedy Algo
0	27.49	38.55	34.77
1	28.1	36.31	31.59
2	30.16	39.59	34.06
3	26.14	36.64	30.63
4	28.42	36.64	36.01
5	30.27	39.7	35.83
6	26.84	37.63	29.11
7	28.64	36.86	29.03
8	30.38	39.92	35.41
9	27.14	38.11	29.1
10	28.1	36.31	24.17
11	29.27	38.22	34.93
12	25.61	35.82	22.01
13	27.99	36.2	31.5
14	30.49	40.03	36.56
15	26.59	37.23	30.25
16	28.32	36.53	31.91
17	28.97	37.74	28.9
18	27.38	38.44	32.23
19	27.49	35.71	32.59
20	28.53	37.12	33.97
21	25.5	35.6	24.12
22	28.42	36.75	32.78
23	28.42	36.86	30.17
24	26.44	37.01	35.59
25	28.21	36.42	29.11
26	28.53	37.12	33.68
27	27.25	38.33	34.6
28	28.32	36.53	31.94
29	28.75	37.34	29.28

Table 23 continued

Cloudlet id	T_{com} Proposed Algo	T_{com} RRA	T_{com} Greedy Algo
30	8.9	38.33	31.75
30	9.12	36.64	28.07
32	9.01	38.55	33.66
33	9.01	37.85	32.67
34	8.37	34.35	32.52
35	9.23	39.81	31.16
36	8.48	35.71	23.33
37	8.68	35.82	24.65
38	9.23	39.81	29.61
39	9.12	38.33	36.24
$\sum T_{com}$	931.31	1496.46	1249.49

Sect. 4. Based on the proposed algorithm, the makespan table is showing in the subsections.

5.3.1 Makespan of 10 Cloudlets

After execution of 10 cloudlets at LERC Cloudlet id 0, 4, 2 finished execution at last in the vm 0, 1, 2, respectively, and at the HERC cloudlet id 0 finished execution last. Maximum makespan at LERC is 8.72 and at HERC is 1.89. So the effective makespan is 8.72, as shown in Tables 24 and 25.

5.3.2 Makespan of 20 Cloudlets

After execution of 20 cloudlets at LERC cloudlet id 0, 7, 14 finished execution at last in the vm 0, 1, 2, respectively, and at the HERC cloudlet id 0 finished execution last. Maximum makespan at LERC is 16.13 and at HERC is 4.78. So the effective makespan is 16.13, as shown in Tables 26 and 27.

Table 24 Makespan at LERC (All cloudlets at LERC) 8 cloudlets

Cloudlet id	VM id LERC	MSp of VMs	MSp of host
0	0	8.39	8.27
4	1	8.72 (MAX)	
2	2	5.99	

Table 25 Makespan at HERC 2 cloudlets

Cloudlet id	VM id HERC	MSp of VMs	MSp of host
0	0	1.89 (MAX)	1.89

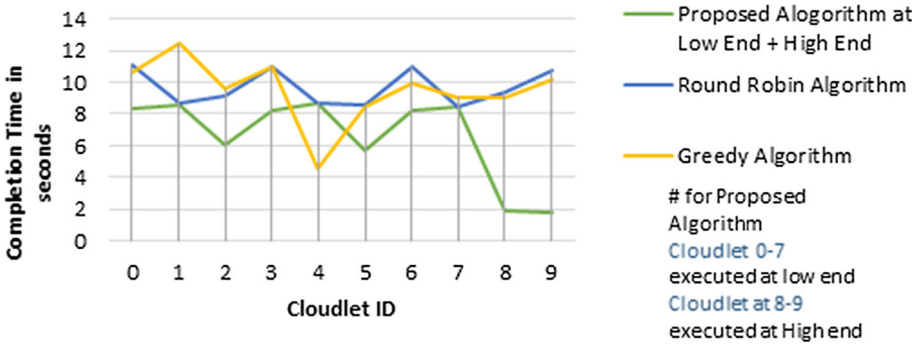

Fig. 6 Comparison with 10 cloudlet between existing and proposed algorithm

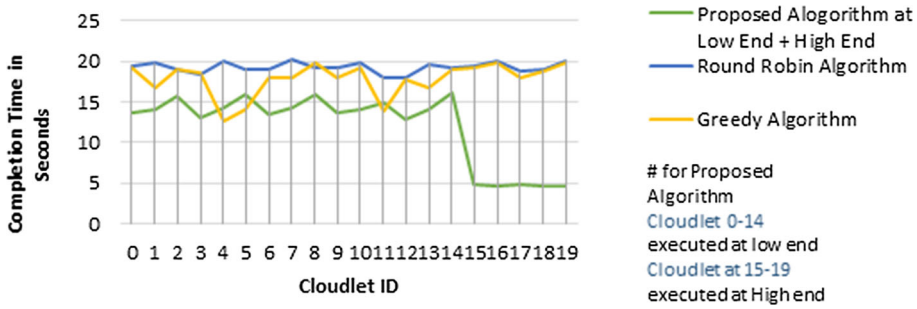

Fig. 7 Comparison with 20 cloudlets between existing and proposed algorithms

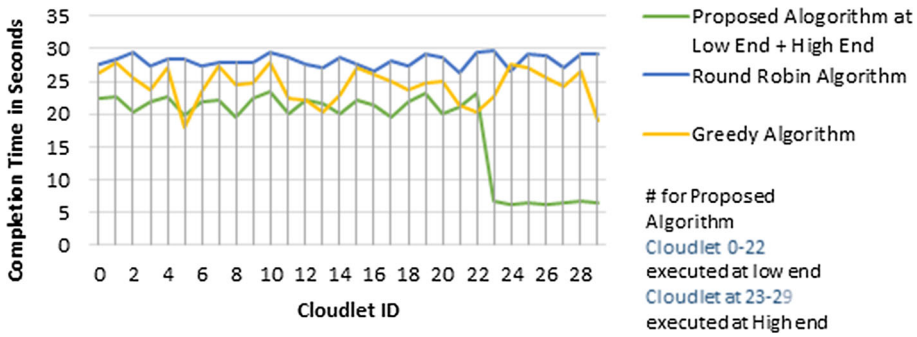

Fig. 8 Comparison with 30 cloudlets between existing and proposed algorithms

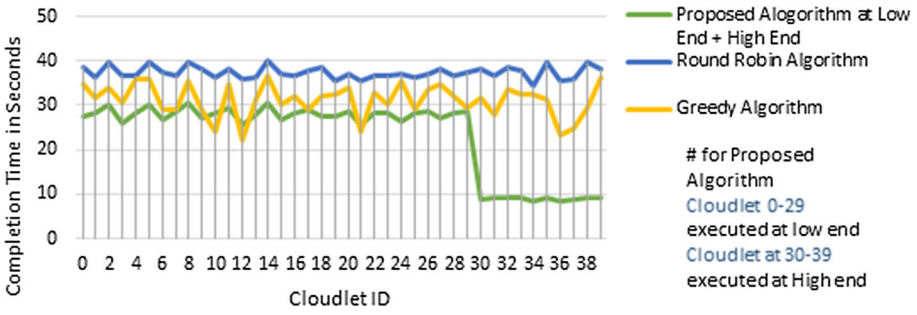

Fig. 9 Comparison with 40 cloudlets between existing and proposed algorithms

Table 26 Makespan at LERC (all cloudlets at LERC) 15 cloudlets

Cloudlet id	VM id LERC	MSp of VMs	MSp of host
0	0	13.75	16.13
7	1	14.38	
14	2	16.13 (MAX)	

Table 27 Makespan at HERC 5 cloudlets

Cloudlet id	VM id HERC	MSp of VMs	MSp of host
0	0	4.78 (MAX)	4.78

Table 28 Makespan at LERC (all cloudlets at LERC) 23 cloudlets

Cloudlet id	VM id LERC	MSp of VMs	MSp of host
9	0	22.47	23.37
10	1	23.37 (MAX)	
2	2	20.29	

5.3.3 Makespan of 30 Cloudlets

After execution of 30 cloudlets at LERC cloudlet id 9, 10, 2 finished execution at last in the vm 0, 1, 2, respectively, and at the HERC cloudlet id 5 finished execution last. Maximum makespan at LERC is 23.37 and at HERC is 6.6. So the effective makespan is 23.37, as shown in Tables 28 and 29.

Table 29 Makespan at HERC 7 cloudlets

Cloudlet id	VM id HERC	MSp of VMs	MSp of host
5	0	6.6 (MAX)	6.6

Table 30 Makespan at LERC (all cloudlets at LERC) 30 cloudlets

Cloudlet id	VM id LERC	MSp of VMs	MSp of host
0	0	27.49	30.49
7	1	28.64	
2	2	30.49 (Max)	

Table 31 Makespan at HERC 10 cloudlets

Cloudlet id	VM id HERC	MSp of VMs	MSp of host
8	0	9.23 (MAX)	9.23

5.3.4 Makespan of 40 Cloudlets

After execution of 40 cloudlets at LERC cloudlet id 0, 7, 14 finished execution at last in the vm 0, 1, 2, respectively, and at the HERC cloudlet id 8 finished execution last. Maximum makespan at LERC is 30.49 and at HERC is 9.23. So the effective makespan is 16.13, as shown in Tables 30 and 31.

A graphical representation of makespan of VMs for the proposed algorithm, Round Robin scheduling and greedy algorithm is described by Figs. 10, 11, 12 and 13.

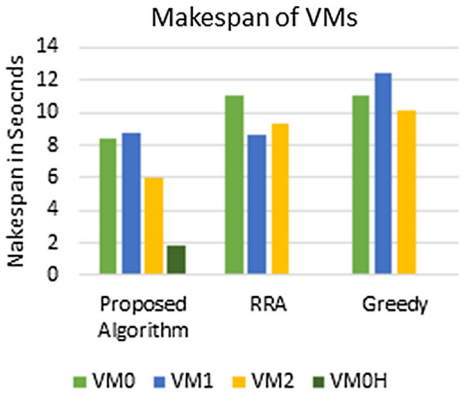

Fig. 10 Makespan of VMs for 10 cloudlets

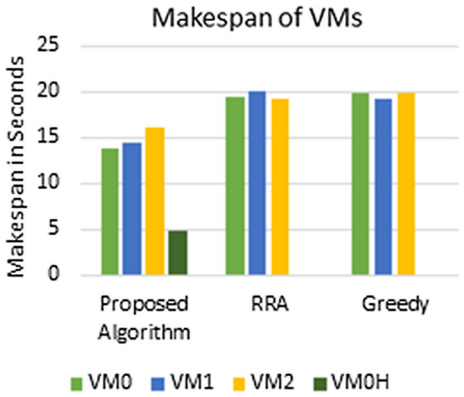

Fig. 11 Makespan of VMs for 20 cloudlets

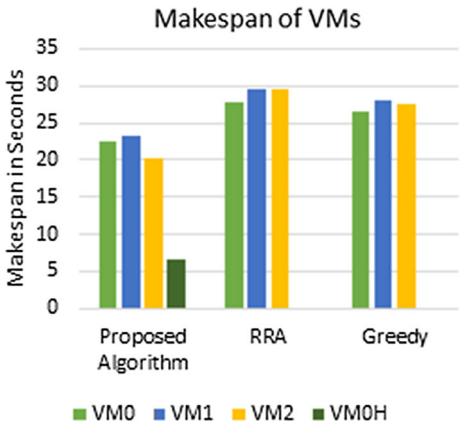

Fig. 12 Makespan of VMs for 30 cloudlets

The comparison of makespan of Round Robin Policy and greedy algorithm with proposed algorithm is graphically shown into Figs. 14, 15, 16 and 17.

For 10 cloudlets when the number of cloudlets is small, then it can be seen from Fig. 14 that the makespan of Round Robin policy is less than that of the greedy algorithm at host but as the number of cloudlets is increased, greedy algorithm improves than Round Robin Policy.

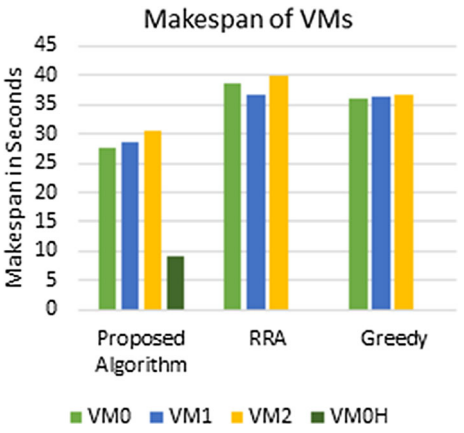

Fig. 13 Makespan of VMs for 40 cloudlets

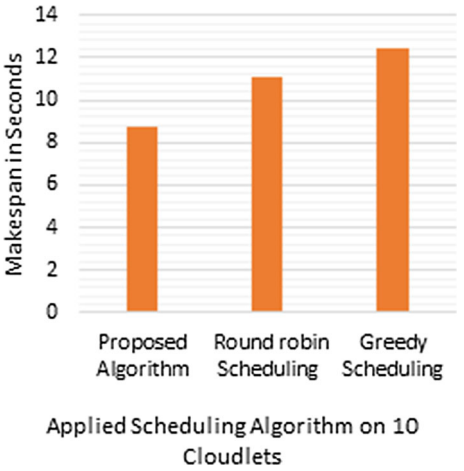

Fig. 14 Makespan for host 10 cloudlets

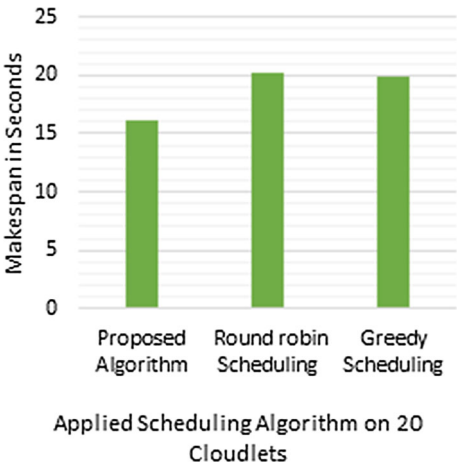

Fig. 15 Makespan for host 20 cloudlets

The makespan for proposed algorithm improves from the makespan of Round Robin policy and greedy algorithm by 21.23 and 30.01%, respectively, for 10 cloudlets.

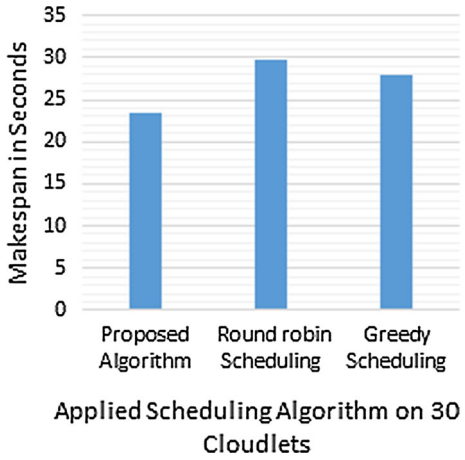

Fig. 16 Makespan for host 30 cloudlets

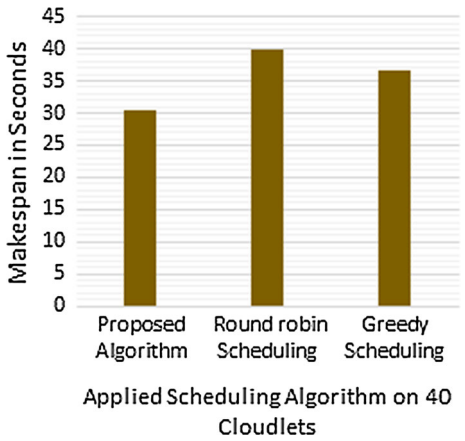

Fig. 17 Makespan for host 40 cloudlets

The makespan for proposed algorithm improves from the makespan of Round Robin policy and greedy algorithm by 20.29 and 19.02%, respectively, for 20 cloudlets.

The makespan for proposed algorithm improves from the makespan of Round Robin policy and greedy algorithm by 21.15 and 16.50%, respectively, for 30 cloudlets.

The makespan for proposed algorithm improves from the makespan of Round Robin policy and greedy algorithm by 23.83 and 16.60%, respectively, for 40 cloudlets.

5.4 Makespan Comparison with Some Prescribed Algorithms

In this section, the proposed work is compared with three advanced algorithms, Improved Round Robin algorithm (IRRA, Opportunistic Load Balancing (OLB), and Minimum Completion Time (MCT). The proposed work and other three advanced algorithms (IRRA, OLB, and MCT) are executed using high-end and low-end clusters both except the RRA and greedy algorithm, so the effective makespan will be considered as maximum makespan between HERC and LERC for the batches of cloudlets, as described in Sect. 3.2.6. RRA and greedy algorithm cannot be compared with the high-end cluster, as described in Sect. 6, so the makespan of high-end cluster set as nil for the RRA and greedy algorithm.

5.4.1 Makespan Comparison with 10 Cloudlets

Table 32 and Fig. 18 show the makespan comparison between proposed work and other advanced algorithms with RRA and algorithm. For relatively low number of cloudlets, MCT outperforms among with an effective makespan of 5.66 s (MAX (Msp at HERC, Msp LERC)), whereas the effective makespan of the proposed algorithm is 8.72 s. OLB has the highest makespan of 10.89 s among the advanced algorithms due to the poor utilization of resources.

5.4.2 Makespan Comparison with 20 Cloudlets

Table 33 and Fig. 19 show the makespan comparison between proposed work and other advanced algorithms with RRA and greedy algorithm with 20 cloudlets. Effective makespan of proposed algorithm improves than before as the number of cloudlet increases. Though MCT still performs well in terms of makespan. OLB always provides a good makespan at HERC, but due to the poor load balancing effective makespan increases at LERC. Effective makespan of the proposed algorithm is 16.13 s, the makespan of LERC.

5.4.3 Makespan Comparison with 30 Cloudlets

Table 34 and Fig. 20 show the makespan comparison between proposed work and other advanced algorithms with RRA and greedy algorithm with 30 cloudlets. Proposed algorithm outperforms this time among all other algorithms. For 30 cloudlets as the job distribution was done to the clusters according to the resource capacities, discussed in Sect. 4, the throughput and QoS improve thoroughly than the other advanced algorithms. MCT shows makespan of 23.48 s, whereas proposed algorithm shows an effective makespan of 23.37 s. IRRA and OLB have an effective makespan of 24.89 and 26.07 s, respectively. Makespan improves 0.11 s for the proposed algorithm than MCT.

5.4.4 Makespan Comparison with 40 Cloudlets

Table 35 and Fig. 21 show the makespan comparison between proposed work and other advanced algorithms with RRA and greedy algorithm with 40 cloudlets. Increasing the cloudlets further to 40 cloudlets the allocation improves dynamically. MCT shows an effective makespan of 31.35 minimum among all other advanced algorithms. But the proposed algorithm surpassed MCT in terms of makespan by 0.86 s. The improve-

Table 32 Makespan comparison with 10 cloudlets with advanced algorithm

Makespan at clusters	Proposed algorithm	IRRA	OLB	MCT	RRA	
Makespan at HERC	1.89	2.56	1.2	3.09		
Makespan at LERC	8.72	6.33	10.89	5.66	11.07	12.48

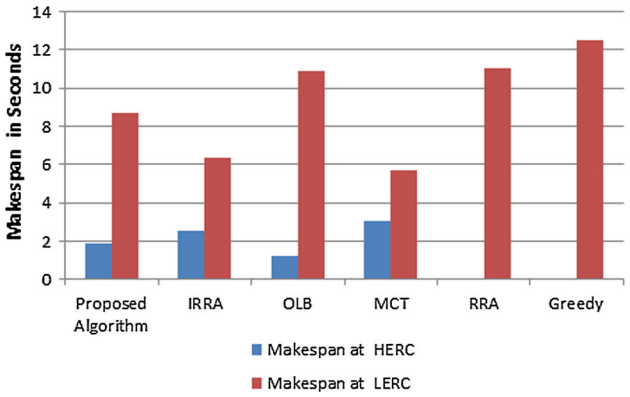

Fig. 18 Makespan comparison with 10 cloudlets

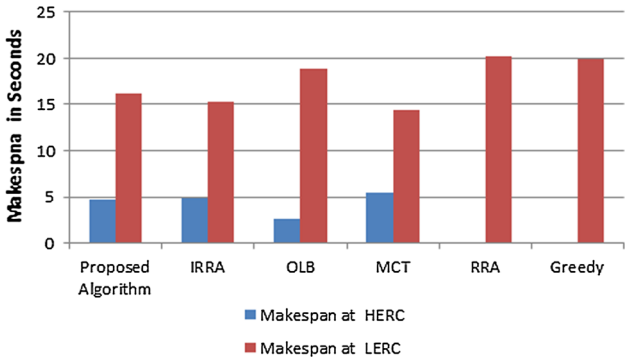

Fig. 19 Makespan comparison with 20 cloudlets

ment of makespan for 40 cloudlets of proposed algorithm w.r.t MCT is much higher than the batch of 30 cloudlets. Proposed algorithm shows a better QoS and throughput in terms of resource allocation according to the capacity with higher number of cloudlets.

6 Justification for Performance Improvement Over the Existing Algorithms

From the experimental results, it has been proved that the proposed algorithm outperforms the Round Robin and greedy algorithm if we use an extra cluster, i.e., HERC. The algorithm is developed from a very practical perspective, now-a-days cloud is being used as on-demand service provider and the cost is measured as on-demand cost per hour basis [56], more over the cloud service providers are using various servers, datacenters of various capacities to satisfy the on-demand service requests. Round Robin and greedy methods are focused only on the cloudlet length by considering that all resources are of same capacity in terms of storage and processing, but in practical scenarios to meet the on-demand service requests and increase the QoS proposed algorithm is capable of providing a better QOS by reducing the makespan of the batch of job hugely.

For further proof 200 cloudlets were taken and they are executed in both LERC and HERC in following manner, HERC and LERC are having configuration as discussed in Sect. 5.

But according to proposed algorithm, the cloudlets were distributed among these two clusters as, LERC executed 150 cloudlets and HERC executed 50 cloudlets with a makespan 145.22 and 127.90 s, respectively. Effective Msp is 145.22.

Table 36 (1st and 2nd row) show that if the cloudlets were executed in LERC or HERC only, then makespan would be much higher.

Table 36 (3rd row) also shows that if an abrupt distribution is made of cloudlets between HERC and LERC, then also makespan is much higher than that of the proposed algorithm. In real-time scenario, the datacenters will have a perfect mixture of clusters, i.e., high-end cluster and low-end cluster. So for distribution of cloudlets among those various capacity resources, cannot be done abruptly, that will cause a downgraded QoS and bad throughput with an overall lesser makespan (depicted in Table 36). But the proposed algorithm can be implemented for a well distribution of cloudlets among those various clusters with properly utilizing all resources with a greater QoS and a lesser makespan.

Table 33 Makespan Comparison with 20 cloudlets with advanced algorithm

Makespan at clusters	Proposed algorithm	IRRA	OLB	MCT	RRA	Greedy
Makespan at HERC	4.78	4.91	2.6	5.51		
Makespan at LERC	16.13	15.32	18.83	14.46	20.17	19.9

Table 34 Makespan comparison with 30 cloudlets with advanced algorithm

Makespan at clusters	Proposed algorithm	IRRA	OLB	MCT	RRA	
Makespan at HERC	6.49	7.22	3.64	7.78		
Makespan at LERC	23.37	24.89	26.07	23.46	29.64	27.99

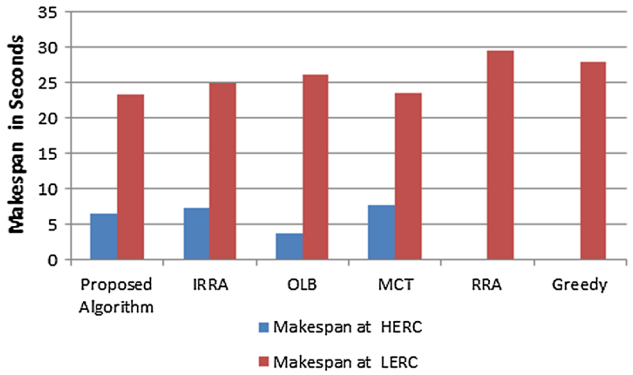

Fig. 20 Makespan comparison with 30 cloudlets

7 Conclusion and Future Work

From all the experimental studies and comparison analysis, we can conclude that the proposed algorithm reduces the makespan of hosts largely and so the algorithm can be implemented in real time to provide a better QoS.

In future study, we will focus on, the hybrid methods to increase the reliability of cloud systems [57] and invoke intelligence to our proposed algorithm. In Datacenter, there are several hosts and VMs; in future our proposed algorithm can further be extended for allocating the VMs with various intelligent information [58,59] and thus the makespan can be reduced further with the help of proper intelligent automatic virtual resource management. Another thing we should concentrate on is the migration of VMs. The nearest service-location to the user is very important to maintain QoS. At last the security concern about cloud information is also one of our primary objectives.

Table 35 Makespan comparison with 40 cloudlets with advanced algorithm

Makespan at clusters	Proposed algorithm	IRRA	OLB	MCT	RRA	Greedy
Makespan at HERC	9.23	9.45	4.89	9.98		
Makespan at LERC	30.49	33.05	35.16	31.35	40.03	36.56

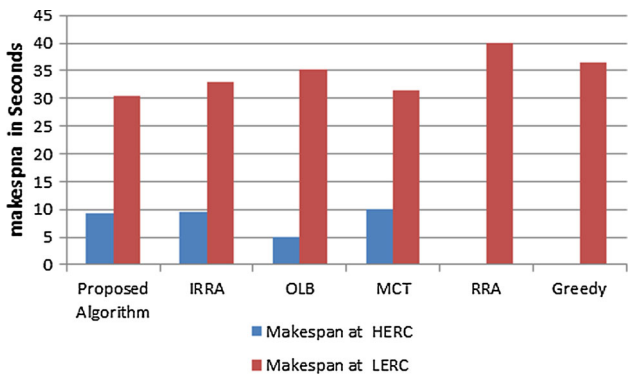

Fig. 21 Makespan comparison with 40 cloudlets

Table 36 Performance analysis based on Makespan

Total execution of 200 cloudlet	Makespan for RR method		Makespan for Greedy method	
200 cloudlet in LERC	361.53		189.67	
200 cloudlet in LERC	587.45		243.73	
100 Cloudlet in LERC and 100 cloudlet in HERC	LERC	HERC	LERC	HERC
	183.96	244.65 (MAX)	163.66 (MAX)	74.09

References

1. Tsai, W.T.; Shao, Q.; Sun, X.; Elston, J.: Real-time service-oriented cloud computing. In: 2010 6th World Congress on Services, Miami, FL, pp. 473–478 (2010). doi:10.1109/SERVICES.2010.127
2. Feng, Y.; Zhijian, W.; Feng, X.; Yuanchao, Z.; Fachao, Z.; Shaosong, Y.: A novel cloud load balancing mechanism in premise of ensuring QoS. Intell. Autom. Soft Comput. **19**(2), 151–163 (2013). doi:10.1080/10798587.2013.786968
3. Ibnouf, R.I.M.; Mustafa, A.B.A.N.: Bandwidth management on cloud computing network. IOSR J. Comput. Eng. (IOSR-JCE) **17**(2), 18–21 (2015). (ISSN: 2278–0661)
4. Singh, J.: Study of response time in cloud computing. Int. J. Inf. Eng. Electron. Bus. **5**, 36–43 (2014). doi:10.5815/ijieeb.2014.05.06
5. Vinothina, V.; Shridaran, R.; Ganpathi, P.: A survey on resource allocation strategies in cloud computing. Int. J. Adv. Comput. Sci. Appl. **3**(6), 97–104 (2012)
6. Lee, G.; Tolia, N.; Ranganathan, P.; Katz, R.H.: Topology aware resource allocation for data-intensive workloads. ACM SIGCOMM Comput. Commun. Rev. **41**(1), 120–124 (2011)
7. Pawar, C.S.; Wagh, R.B.: A review of resource allocation policies in cloud computing. World J. Sci. Technol. **2**(3), 165–167 (2012)
8. Goudaezi, H.; Pedram, M.: Multidimensional SLA-based resource allocation for multi-tier cloud computing systems. In: IEEE 4th International Conference on Cloud computing, pp. 324–331 (2011)
9. Kumar, K.; et al.: Resource allocation for real time cloudlets using cloud computing. In: Proceedings of 20th International Conference

on IEEE Computer Communications and Networks (ICCCN), pp. 1–7 (2011)
10. Endo, P.T.; et al.: Resource allocation for distributed cloud: concept and research challenges. IEEE Commun. Soc. **25**(4), 42–46 (2011)
11. Chen, Z.; Yoon, J.P.: Parallel, grid, cloud and internet computing. In: International Conference on P2P, pp. 250–257. IEEE (2010)
12. Keahey, K.; Tsugawa, M.; Matsunaga, A.; Fortes, J.A.B.: Sky computing. IEEE Internet Comput. **13**(5), 43–51 (2009)
13. Warneke, D.; Kao, O.: Exploiting resource allocation for efficient parallel data processing in the cloud. IEEE Trans. Parallel Distrib. Syst. **22**(6), 985–997 (2011)
14. Wuhib, F.; Stadler, R.: Distributed monitoring and resource management for large cloud environments. In: 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, pp. 970–975. IEEE (2011)
15. Inomata, A.; Morikawa T.; Ikebe M.; Rahman, Md.M.: Proposal and evaluation of dynamic resource allocation method based on the load of VMs on IaaS. In: 2011 4th IFIP International Conference New Technologies, Mobility and Security (NTMS), pp. 1–6. IEEE (2011)
16. An, B.; Lesser, V.; Irwin, D.; Zink, M.: Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In: Conference at University of Massachusetts, Amherst, USA, pp. 981–988 (2010)
17. Jung, G.; Sim, K.M.: Location-aware dynamic resource allocation model for cloud computing environment. In: International Conference on Information and Computer Applications (ICICA), pp. 37–41. IACSIT Press, Singapore (2012)
18. Yanggratoke, R.; Wuhib, F.; Stadler, R.: Gossip-based resource allocation for green computing in large clouds. In: 7th International Conference on Network and Service Management, Paris, France, pp. 24–28 (2011)
19. Gmach, D.; Rolia J.; cherkasova, L.: Satisfying service level objectives in a self-managing resource pool. In: Proceedings of Third IEEE International Conference on Self-Adaptive and Self-Organizing System, pp. 243–253 (2009)
20. Minarolli, D.; Freisleben, B.: Utility-based Resource allocations for virtual machines in cloud computing. In: Computers and Communications (ISCC), pp. 410–417. IEEE (2011)
21. Huu, T.T.; Montagnat, J.: Virtual resource allocations distribution on a cloud infrastructure. In: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), pp. 612–617. IEEE (2010)
22. Anbar, A.; Narayana, V.K.; El-Ghazawi, T.: Distributed shared memory programming in the cloud. In: 2012 12th IEEE/ACM International Symposium of Cluster, Cloud and Grid Computing (CCGrid), pp. 707–708. IEEE (2012)
23. Wood, T.; et al.: Black box and gray box strategies for virtual machine migration. In: Proceedings of 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI), pp. 229–242 (2007)
24. Huang, K.-C.; Lai, K.-P.: Processor allocation policies for reducing resource fragmentation in multi cluster grid and cloud environments. In: Computer Symposium (ICS), pp. 971–976. IEEE (2010)
25. Banerjee, S.; Adhikary, M.; Biswas, U.: Smart task assignment model for cloud service provider. In: Special Issue of International Journal of Computer Applications on Advanced Computing and Communication Technologies for HPC Applications—ACCTHPCA, June 2012, pp. 43–46 (2012). (ISSN: 0975 8887)
26. Banerjee, S.; Adhikary, M.; Biswas, U.: Advanced task scheduling for cloud service provider using genetic algorithm. IOSR J. Eng. **2**(7), 153–159 (2012). (ISSN: 2250-3021)
27. Banerjee, S.; Adhikari, M.; Kar, S.; Biswas, U.: Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud. Arab J. Sci. Eng. **40**(5), 14091425 (2015). doi:10.1007/s13369-015-1626-9. (ISSN: 1319-8025)
28. Banerjee, S.; Adhikari, M.; Biswas, U.: Design and analysis of an efficient QoS improvement policy in cloud computing, Service Oriented Computing and Applications, July, 2016, ISSN: 1863-2386 (Print) 1863-2394 (Online), Springer London. (Accepted, yet to be published) (2016)
29. Banerjee, S.; Adhikary, M.; Mondal, D.; Biswas, U.: Service delivery improvement for the cloud service providers and customers. Int. J. Comput. Appl. **51**(5), 20–23 (2012). (ISSN: 0975 8887)
30. Ali, S.K.F.; Hamad, M.B.: Implementation of an EDF algorithm in a cloud computing environment using the CloudSim Tool. In: International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), Khartoum, 2015, pp. 193–198. doi:10.1109/ICCNEEE.2015.7381360 (2015)
31. Shi, Y.; Lo, D.; Qian, K.: Teaching secure cloud computing concepts with open source CloudSim environment. In: IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, pp. 247–252 (2016). doi:10.1109/COMPSAC.2016.201
32. Khatua, S.; Ghosh, A.; Mukherjee, N.: Optimizing the utilization of virtual resources in Cloud environment. In: 2010 IEEE International Conference on Virtual Environments, Human–Computer Interfaces and Measurement Systems, pp. 82–87 (2010)
33. Nivodhini, M.K.; Kousalya, K.; Malliga, S.: Algorithms to improve scheduling techniques in IaaS cloud. In: 2013 International Conference on Information Communication and Embedded Systems (ICICES), pp. 246–250. IEEE (2013)
34. Pasha, N.; Agarwal, A.; Rastogi, R.: Round robin approach for VM load balancing algorithm in cloud computing environment. Int. J. Adv. Res. Comput. Sci. Softw. Eng. IJARCSSE **4**(5), 34–39 (2014)
35. Li, J.; Feng, L.; Fang, S.: A Greedy-Based Cloudlet Scheduling Algorithm in Cloud Computing. Academy Publisher, New York (2014)
36. Selvi, S.; Maheswari, R.; Kalaavathi, B.: Deadline cost based cloudlet scheduling using greedy approach in a multi-layer environment. Int. J. Comput. Trends Technol. (IJCTT) **7**(2), 74–79 (2014)
37. Kapgate, D.: Improved round robin algorithm for data center selection in cloud computing. Int. J. Eng. Sci. Res. Technol. (IJESRT) **3**(2), 686–691 (2014). (ISSN: 2277-9655)
38. Amandeep, V.; Mohammad, Y.F.: Different strategies for load balancing in cloud computing environment: a critical study. Int. J. Sci. Res. Eng. Technol. (IJSREC) **3**(2) (2014). (ISSN: 2278 0882)
39. Mehdi, N.A.; Mamat, A.; Amer, A.; Abdul-Mehdi, Z.T.: Minimum completion time for power-aware scheduling in cloud computing. Developments in E-systems Engineering (DeSE), pp. 484–489. IEEE (2011)
40. Malhotra, R.; Jain, P.: Study and comparison of CloudSim simulators in the cloud computing. Stand. Int. J. (The SIJ) **1**(4), 111–115 (2013)
41. Kaur, K.; Rai, A.K.: A comparative analysis: grid, cluster and cloud computing. Int. J. Adv. Res. Comput. Commun. Eng. **3**(3), 5730–5734 (2014)
42. Pagare, J.D.; Koli, N.A.: Design and simulate cloud computing environment using cloudsim. IJCTA **6**(1), 35–42 (2015)
43. Van den Bossche, R.; Vanmechelen, K.; Broeckhove, J.: Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In: 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD). IEEE (2010)
44. Sindhu, S.; Mukherjee, S.: Efficient Task Scheduling Algorithms for Cloud Computing Environment. High Performance Architecture and Grid Computing. Springer, Berlin (2011)
45. Li, J.; et al.: Feedback dynamic algorithms for preemptable job scheduling in cloud systems. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Vol. 1. IEEE (2010)

46. 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. Wiley, Hoboken (2011)
47. Uthaya Banu, M.; Saravanan, K.: Optimizing the cost for resource subscription policy in IaaS cloud. Int. J. Eng. Trends Technol. (IJETT) **V6**(6), 296–301 (2013). (ISSN: 2231–5381)
48. Cloudlet (cloudsim 3.0 API): The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, The University of Melbourne. Retrieved from http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/Cloudlet.html Retrived on Aug 14.
49. Rawat, P.S.: 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). doi:10.5120/8484-2424
50. Wickremasinghe, B.: CloudAnalyst: a CloudSim-based Tool for modelling and analysis of large scale cloud computing environments. MEDC Project Report, University of Melbourne, Melbourne, p. 44 (2009)
51. Mahmood, Z.: Cloud Computing Challenges Limitations and R and D Solutions. Springer (2014). ISBN: 978-3-319-10530-7, doi:10.1007/978-3-319-10530-7
52. Pop, F.; Potop-Butucaru, M.: Adaptive Resource Management and Scheduling for Cloud Computing. Springer (2015). ISBN: 978-3-319-13464-2, doi:10.1007/978-3-319-13464-2
53. Han Y.H.; Park D.S.; Jia W.; Yeo, S.S: Ubiquitous Information Technologies and Applications. Springer, Dordrecht, eBook ISBN: 978-94-007-5857-5, doi:10.1007/978-94-007-5857-5
54. Cluster-Defination of Cluster, Oxford Dictionary, Retrived on April 2017, From https://en.oxforddictionaries.com/definition/cluster
55. Magalhes, D.; Calheiros, R.N.; Buyya, R.; Gomes, D.G.: Workload modelling for resource usage analysis and simulation in cloud computing. Comput. Electr. Eng. **47**, 69–81 (2015). doi:10.1016/j.compeleceng.2015.08.016
56. Amazon Elastic Compute Cloud, Wikipedia, Retrived on March 2017, From https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud
57. Xuejie, Z.; Zhijian, W.; Feng, X.: Reliability evaluation of cloud computing systems using hybrid methods. Intell. Autom. Soft Comput. **19**(2), 165–174 (2013). doi:10.1080/10798587.2013.786969
58. Das, A.K.; Adhikary, T.; Razzaque, Md.A.; Hong, C.S.: An intelligent approach for virtual machine and QoS provisioning in cloud computing. In: The International Conference on Information Networking 2013 (ICOIN), pp. 462–467. IEEE (2013)
59. Roy, S.; Banerjee, S.; Chowdhury, K.R.; Biswas, U.: Development and analysis of a three phase cloudlet allocation algorithm. J. King Saud Univ. Comput. Inf. Sci. (2016). doi:10.1016/j.jksuci.2016.01.003