Journal of King Saud University – Computer and Information Sciences (2017) **29**, 473–483

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

Development and analysis of a three phase cloudlet allocation algorithm

Sudip Roy [a], Sourav Banerjee [a,*], K.R. Chowdhury [a], Utpal Biswas [b]

[a] *Kalyani Government Engineering College, Kalyani, Nadia, India*
[b] *University of Kalyani, Kalyani, Nadia, India*

Received 15 September 2015; revised 11 January 2016; accepted 11 January 2016
Available online 28 March 2016

KEYWORDS

Cloud computing;
Cloudlet allocation;
Cloud service provider;
Cloudsim

Abstract Cloud computing is one of the most popular and pragmatic topics of research nowadays. The allocation of cloudlet(s) to suitable VM(s) is one of the most challenging areas of research in the domain of cloud computing. This paper highlights a new cloudlet allocation algorithm which improves the performance of a cloud service provider (CSP) in comparison with the other existing cloudlet allocation algorithms. The proposed Range wise Busy-checking 2-way Balanced (RB2B) cloudlet allocation algorithm optimizes few basic parameters associated with the performance analysis. An extensive simulation is done to evaluate the proposed algorithm using Cloudsim to attest its efficacy in comparison to the other existing allocation policies.

1. Introduction

Cloud computing is the newest trend in the field of computer science and it is said to be the future of modern technology. Cloud computing is popular mostly for its special ability to utilize shared resources most efficiently. The allocation of the cloudlets to the suitable resources known as the virtual machines or VMs (Fu and Zhou, 2015) is an essential requirement in cloud computing environment. In a typical Cloud environment there is a module known as datacenter broker (DCB) which controls the entire datacenter including the cloudlet allocation to VMs. So, like any normal computing performance optimization and improvement of the allocation algorithm is always a possibility.

Engineering an efficient cloudlet allocation algorithm (Zhang et al., 2007) is a challenging research area and many such policies have been proposed, analyzed and compared on heterogeneous parallel computing environments. A new mechanism had been introduced, known as effective aggregated computing power (EACP) (Radulescu and Van Gemund, 1999) that improves the performance. The Adaptive weighted factoring (AWF) (Carino and Banicescu 2008) is used for scheduling parallel loops. The dynamic loop scheduling with reinforcement learning (Rashid et al., 2008) (DLS-with-RL) is very much effective for use in time stepping scientific applications with many steps. The scheduling (Aziz and El-Rewini, 2008) policies for Grid environment use several methods which are similar yet different to the mechanisms of cloudlet allocation policies. Genetic Algorithms (Pop, 2008) are also used for

* Corresponding author.
E-mail addresses: sanosuke009@gmail.com (S. Roy), mr.sourav.banerjee@ieee.org (S. Banerjee), papanthegenius@gmail.com (K.R. Chowdhury), utpal01in@yahoo.com (U. Biswas).
Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2016.01.003
1319-1578

scheduling. The Opportunistic Load Balancing (OLB) heuristic (Braun et al., 2008) (Makelainen et al., 2014) (Iordache et al., 2007) chooses a cloudlet from the batch of cloudlets arbitrarily and allocates it to the next VM which is estimated to be available, not considering the cloudlet's expected execution time on that VM, resulting in very poor makespan (Wang et al., 2006). The Minimum Execution Time (MET) (George Amalarethinam and Muthulakshmi, 2011) allocates each cloudlet chosen arbitrarily to the VM with the least possible execution time resulting in the severe imbalance of load across the VMs. The Minimum Completion Time (MCT) (George Amalarethinam and Muthulakshmi, 2011) heuristic allocates each cloudlet to the VM with the minimum completion time for that cloudlet. It literally combines the advantages of OLB and MET. The QoS (Quality of Service) guided Min–min (He et al., 2003) assigns cloudlets which require higher bandwidth. The QoS priority grouping scheduling (Dong et al., 2006) gives importance to deadlines. The QoS Sufferage (Ullah Munir et al., 2007) considers network bandwidth as a major factor and schedules tasks based on their bandwidth requirement. The Grid-JQA (Khanli and Analoui, 2007, 2008) scheduling solution uses an aggregation formula which combines the parameters together with weighting factors to calculate QoS. The proposal of a dissimilar and new scheduling algorithm (Afzal et al., 2008) that tries to minimize the cost of the execution as well as satisfying the QoS constraints, views the scheduling environment as a queuing system. Another user oriented scheduling algorithm which uses an advanced reservation and resource selection techniques (Elmroth and Tordsson, 2008) minimizes the execution time of individual cloudlets without considering the make span. The multiple resources scheduling (MRS) (Benjamin Khoo et al., 2007) algorithm considers both the system capabilities and the resource requirements of cloudlets as majority factors. In cloud computing environment, most of the allocation policies load some specific resources comparatively more heavily, leaving other resources either idle or least loaded (Livny and Melman, 2011). As a result, load balancing is an important issue in cloud computing which affects the performance of the cloud service provider.

The objective of this paper is to improve the existing allocation policies in this domain by devising a new cloudlet allocation algorithm RB2B that focuses mostly on reducing waiting time and make span, at the same time optimizing VM (Marisol García-Valls et al., 2014) utilization to a remarkable amount by distributing the number of cloudlets to the VMs in a most uniform way. The proposed algorithm is incorporated in the datacenter broker (DCB) module. The DCB policy is enhanced with this proposed work and termed as advanced datacenter broker (ADCB) module in this study.

2. Related works

In this paper, few existing allocation policies are taken into account to analyze and compare the advantages of the proposed RB2B. They are described as follows.

2.1. Min–min (Parsa and Entezari-Maleki, 2009; Kumar and Dutta Pramanik, 2012; El-kenawy et al., 2012)

Initially a matrix is taken for all unassigned cloudlets. There are two phases in Min–min. In the first phase the set of minimum computation time for each cloudlet in the matrix is calculated and found. In the second phase, the cloudlet with the overall minimum expected computation time is chosen from the matrix and assigned to the corresponding VM. Then the assigned cloudlet is removed from the matrix and the entries of the matrix are modified accordingly. This process of Min–min is repeated until there is no cloudlet left in the matrix, that is, all cloudlets in the matrix are mapped. This algorithm takes $O(mn^2)$ time where m is the number of VMs and n is the number of cloudlets.

2.2. Max–min (Parsa and Entezari-Maleki, 2009)

This algorithm is almost similar to Min–Min, but there is a distinct difference in the second phase. This Max–Min first chooses the cloudlet with maximum computation time from the matrix and assigns it to the VM on which the chosen cloudlet gives minimum time to compute. This algorithm also takes $O(mn^2)$ time where m is the number of VMs and n is the number of cloudlets.

2.3. RASA (Parsa and Entezari-Maleki, 2009)

This algorithm actually combines the advantages of both Min–min and Max–min. If the number of available VMs is odd, the Min–min algorithm is applied to allocate the first cloudlet, otherwise the Max–min algorithm is applied. The whole process can be divided into a number of rounds where in each round two cloudlets are allocated to appropriate VMs by one of the two strategies, alternatively. The rule is, if the first cloudlet of the current round is allocated to a VM by the Min–min strategy, the next cloudlet will be allocated by the Max–min strategy. In the next round, the cloudlet allocation begins with an algorithm different from the last round. For example if the first round begins with the Max–min algorithm, the second round will begin with the Min–min algorithm. Experimental results show that if the numbers of available resources are odd then starting with applying the Min–min algorithm in the first round gives the better result. Otherwise, it is better to apply the max–min strategy at first. Min–min and Max–min are exchanged alternatively to result in consecutive execution of small and large cloudlets on different VMs and therefore, the waiting time of the small cloudlets in Max–min algorithm and the waiting time of the large cloudlets in Min–min algorithm are ignored. As RASA doesn't consist of any time consuming instruction, the time complexity of RASA (Maheswaran et al., 1999) is $O(mn^2)$ where m is the number of VMs and n is the number of cloudlets.

2.4. Round Robin Allocation (RRA) (Parsa and Entezari-Maleki, 2009; Bhatia et al., 2010; Banerjee et al., 2015)

It allocates the cloudlet to first available VM. For example, consider there are four cloudlets (C_0, C_1, C_2, C_3, and C_4) and three

Table 1 RRA allocation style.

Cloudlet	VM
C_0	VM_0
C_1	VM_1
C_2	VM_2
C_3	VM_0
C_4	VM_1

VMs (VM_0, VM_1, and VM_2) present in the system. Table 1 illustrates the allocation fashion. According to this policy, cloudlet C_0 allocated to VM_0, C_1 allocated to VM_1, C_2 allocated to VM_2, C_3 allocated to the VM_0 and C_4 allocated to VM_1.

2.5. Conductance Algorithm (CA) (Chatterjee et al., 2014)

This algorithm considers the capacity of each VM as a pipe. It calculates the Conductance (processing capacity) as per Eq. (1) of each VM as the ratio of its processing speed to the sum of the processing speeds of all the VMs present in the system. The processing speed of VM_j is measured in million instructions per second (MIPS) and is presented as $MIPS_j$. The Conductance of VM_j is denoted by $Conductance_j$.

$$\text{Conductance}_j = \text{MIPS}_j \Big/ \sum_{j=1}^{n} \text{MIPS}_j \tag{1}$$

After the calculation of conductance, the number of cloudlets that should be allocated to VM_j is calculated by multiplying the $Conductance_j$ of that particular VM_j with the length of the cloudlet list. To determine the strip length of each VM where the strip length of VM_j is denoted by $Striplength_j$, (2) is used. It determines the number of cloudlets the VM can process.

$$\text{Striplength}_i = \text{Conductance}_i \times (\text{length of cloudlet list}) \tag{2}$$

Few limitations of existing algorithms & proposition of the new algorithm:

Min–min algorithm lacks uniform resource utilization in a way that it chooses smaller cloudlets first which makes use of VMs with higher processing speed. Therefore, the scheduling is not optimal when the number of smaller cloudlets is greater than the large ones. To overcome this drawback (Armbrust M et al., 2010), Max–min algorithm schedules larger cloudlets at first. But sometimes, due to the prior execution of larger cloudlets the makespan may increase. Max–min also increases the waiting time of smaller cloudlets. RASA has disadvantages of both Min–min and Max–min, although it approaches to reduce them. In RRA, large cloudlets are often assigned to the VMs with low MIPS and hence take a longer time to execute as well as increasing the waiting time and the response time of the cloudlets. Moreover, sometimes it may also happen that the most powerful VMs get the smaller cloudlets and hence its resource utilization gets wasted and at the same time decreasing the overall performance and in case of CA the low MIPS VMs sometimes get free too quickly thus wasting its resources and the high MIPS VMs sometimes get overloaded when the length of the longest cloudlets assigned to them are very large. Table 2 depicts a comparative briefing of the five existing algorithms described in this section.

To minimize these drawbacks and to improve the analytical parameters a new cloudlet allocation algorithm is proposed in this paper which is Range wise Busy-checking 2-way Balanced (RB2B) allocation algorithm. The proposed RB2B works in such a way that a cloudlet will always be allocated to a suitable VM according to the cloudlet's size and the number of cloudlets allocated to the VMs are almost uniformly distributed, maximizing the percentage of resource utilization and optimizing the finish time for each cloudlet so that they get minimized in comparison with the other policies. As a consequence of minimizing the finish time, the make span of the cloudlets is also minimized.

3. CloudSim

Several Grid simulators (George Amalarethinam and Muthulakshmi (2011)) such as GridSim, SimGrid, and GangSim are used to simulate the Grid application in a distributed environment, but don't support the infrastructure and application-level requirements to simulate Cloud computing paradigm (Khanli and Analoui, 2008). A Cloud computing environment simulator must support for real-time trading of services between customers and providers. The CloudSim framework (Belalem et al., 2010) which is an open source cloud computing environment simulator is developed on GridSim toolkit (Bhatia et al., 2010). The CloudSim supports resource management and application scheduling simulation and implementation of new policies. The CloudSim provides a series of extended classes and methods. It also helps to analyze new cloudlet allocation policies and scheduling criteria at different levels. The CloudSim supports modification of in built classes, module deployment techniques and performance analysis by

Table 2 A comparative study of existing policies.

	Min–min	Max–min	RASA	RRA	Conductance
Nature of allocation	Static	Static	Static	Dynamic	Static
Advantages	The idle time of the VMs is almost zero	Removes the disadvantages of Min–min	Advantages of both Min–min & Max–min	Much simpler to implement	Makespan is lesser than other four policies
Disadvantages	(i) Lacks uniform resource utilization (ii) Not optimal when the number of smaller cloudlets is greater	(i) Makespan is greater than the others (ii) Increases the waiting time of smaller cloudlets	Disadvantages of both Min–min & Max–min	Large cloudlets are often assigned to the VMs with low MIPS increasing waiting time and the response time	Resource utilization is highly non-uniform and wastage of resource
Time complexity	$O(mn^2)$	$O(mn^2)$	$O(mn^2)$	$O(mn)$	$O(mn^2)$

implementing few interfaces. The present study aims at utilizing CloudSim 3.0.3 by modifying the datacenter broker algorithm. The Datacenter Broker algorithm plays a key role in cloud service management. Few other important modules of CloudSim 3.0.3 toolkit are given below.

3.1. Cloud Information Service (CIS)

CIS is nothing but database level match-making service. User requests are mapped by CIS to suitable cloud providers. CIS and Datacenter Broker of CloudSim perform resource discovery and information interaction, it is the core of simulated scheduling (Belalem et al., 2010; Bhatia et al., 2010; Calheiros R. N et al., 2009).

3.2. Data center (DC)

Data center consists of hosts or physical (Buyya et al., 2009a,b) nodes.

3.3. Cloudlet

It is a package of processes or tasks. A cloudlet is sent from the user for processing (Calheiros et al., 2010) to the DC. It consists of fields such as cloudlet ID, cloudlet length, arrival time etc. The cloudlet length of cloudlets should be greater than or equal to one (Gulati and Chopra, 2013).

3.4. Virtual machine (VM)

A virtual machine is an image of shared resource that imitates the characteristics of an individual processing element.

3.5. Datacenter Broker (DCB)

This class encapsulates the properties of a broker, which is capable of mediating between service providers and users, depending on users' requirements (Buyya et al., 2009a,b). Service tasks are deployed across clouds by the brokers. New and developing scheduling algorithms and cloudlet allocation policies are implemented in Datacenter Broker method.

3.6. VM scheduler

VM scheduler is an abstract class. It is implemented by a Host component. It represents and specifies the policies whether it is space-shared or time-shared, according to the requirements of allocating cloudlets to VMs.

3.7. VM allocation

It is used as the default VM allocation to the host in CloudSim.

4. Range wise Busy-checking 2-way Balanced (RB2B)

The proposed RB2B is developed in such a way that it overcomes several drawbacks of the previous works to improve the performance.

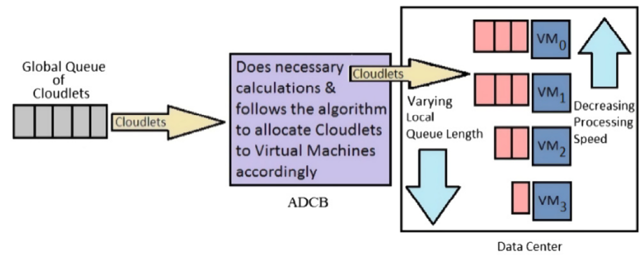

Figure 1 RB2B workflow model.

4.1. A brief description

It's a three phase allocation algorithm. The phases are as follows: (a) VM categorization phase, (b) Two round busy checking phase and (c) Cloudlet still not allocated (CSNA) phase. Also, there are two *balancing conditions* known as two way balancing conditions to *balance the cloudlet distribution among the VMs* as uniformly as possible. A VM is considered *suitable* only when it is *not busy* and *it satisfies two way balancing conditions*. The process of RB2B is described as follows:

In a nutshell, the VMs are created and allocated to the host(s) and are arranged in increasing order of processing speed. A cloudlet arrives from the global queue (GQ) to the ADCB where the proposed cloudlet allocation algorithm is implemented. The block diagram of the whole process how RB2B works is portrayed in Fig. 1.

In the first phase, ADCB measures the length (million instructions) of the cloudlet and accordingly chooses a VM (termed as *targeted* VM) following a cloudlet size acceptability range which is described in Section 4.3 in detail. If the chosen VM is available then ADCB will check it for a condition defined as Balance threshold which will be discussed in detail in 4.3. If the condition is satisfied the cloudlet will be allocated to the targeted VM. If the targeted VM is not available or the Balance threshold condition is not satisfied the ADCB will search for another VM which satisfies this condition. If such suitable VM is available, the cloudlet is allocated. But if still no suitable VM is found then the third phase of RB2B will commence. In this phase ADCB will search for a VM according to EFT (Bittencourt et al., 2010) which also satisfies the two-way balancing conditions i.e. Balance threshold and Local queue (LQ) length limitation, and the cloudlet will be queued to the local queue of this VM.

4.2. Phases of RB2B

There are three phases in RB2B. They are described in this section in detail.

4.2.1. VM categorization phase

In this phase, the VMs are categorized following the suitable acceptability length of cloudlet(s).

Assume, the total number of VMs created is '*m*'. Now the first priority of RB2B is to choose a suitable VM of certain MIPS for the arriving cloudlet according to the cloudlet length. So, ADCB will initially define a cloudlet length acceptance range for each VM. The distribution of different ranges is calculated in the manner described forthwith. Suppose C_{min}

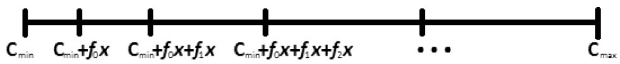

Figure 2 Range distributions among VMs.

Table 3 Cloudlet length acceptability ranges for each VM.

VM	Lower limit	Upper limit
VM_0	C_{min}	$C_{min} + f_0x$
VM_1	$C_{min} + f_0x + 1$	$C_{min} + f_0x + f_1x$
...	...	...
VM_m	$C_{min} + f_0x + f_1x + \cdots + f_{m-1}x + 1$	$C_{min} + f_0x + f_1x + \cdots + f_mx$
...	...	...
VM_{n-1}	$C_{min} + f_0x + f_1x + \cdots + f_{n-2}x + 1$	$C_{min} + f_0x + f_1x + \cdots + f_{n-1}x = C_{max}$

and C_{max} are the minimum and maximum cloudlet length (In MI). $MIPS_i$ is the processing speed of VM_i. So, the total of the processing speeds of the VMs is

$$MIPS_{Total} = \sum_{i=0}^{n-1} MIPS_i \tag{3}$$

Now, if the ratio of the difference of maximum and minimum cloudlet length and $MIPS_{Total}$ be x, then the calculation of x will be as per (4).

$$x = (C_{max} - C_{min})/MIPS_{Total} \tag{4}$$

Let f_i denote the MIPS of VM_i, since VMs are arranged in increasing order of processing speed so $MIPS_{i+1} > MIPS_i$. The distribution of the MIPS ranges is shown in Fig. 2.

The lower limit and upper limits of cloudlet length acceptability of VM_m are respectively:

$$C_{min} + f_0x + f_1x + \cdots + f_{m-1}x + 1 \text{ and}$$
$$C_{min} + f_0x + f_1x + \cdots + f_mx \tag{5}$$

The cloudlet length acceptability ranges for the VMs are depicted in Table 3.

After the arrival of a cloudlet, the ADCB finds a suitable VM considering the cloudlet's length and the VM chosen in this phase is termed as targeted VM.

4.2.2. Two round busy checking phase

This is the second phase of VM selection. After completion of the first phase the ADCB checks whether that targeted VM is available or not. If that VM is available the ADCB will check the Balance threshold condition. If the condition has been satisfied the cloudlet will be allocated to that VM.

Otherwise, ADCB searches for the other VMs following the two rounds:

(i) If the targeted VM is not the VM with the highest MIPS, then ADCB checks whether the next VM with higher MIPS is busy and whether it satisfies Balance threshold condition or not. If this VM is found to be suitable, the cloudlet is allocated to it. Otherwise, the next VM with higher MIPS will be checked in the same way. This round of busy checking will continue until either the cloudlet is allocated or the VM with the highest MIPS is checked. This round is illustrated in Fig. 3.

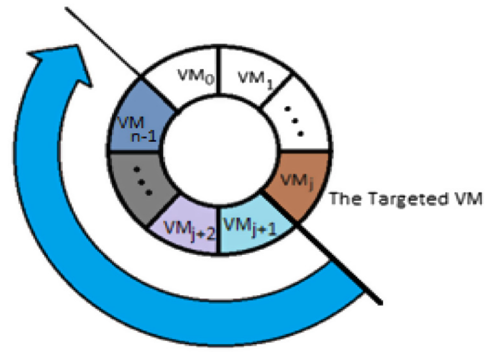

Figure 3 1st round of Busy Checking.

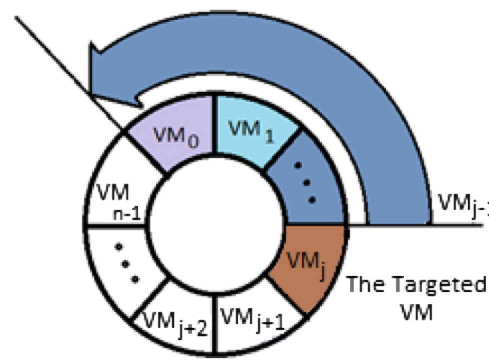

Figure 4 2nd round of busy checking.

(ii) If the cloudlet is still not allocated after the first round, then the second round of checking will commence. At first the VM with the lower MIPS which is next to the targeted VM is checked whether it is suitable or not. If it is suitable then the cloudlet is allocated to it. Otherwise, the next VMs with lower MIPS are checked in a similar manner until the cloudlet is allocated to a suitable VM or the VM with the lowest MIPS is checked. This round is illustrated in Fig. 4.

4.2.3. Cloudlet-still-not-allocated phase (CSNA)

It is the last phase of RB2B. After the first two phases if the balancing factors decide to allocate the arriving cloudlet to a VM, ADCB will move on to the next cloudlet. But if the arriving

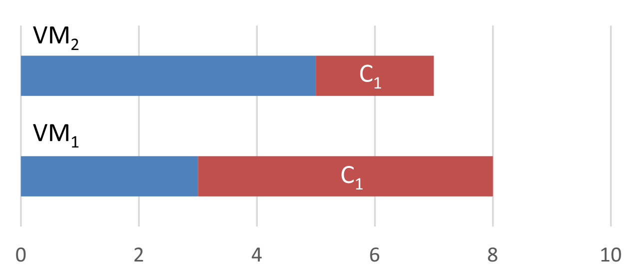

Figure 5 Cloudlets in CSNA phase.

cloudlet is still not allocated then the ADCB will search for the VM with earliest finish time for that cloudlet, provided that the two-way balance conditions described in 4.3 are satisfied as well.

In Fig. 5, arriving cloudlet C_1 arrives and ADCB finds both of the VMs busy. Now it is clear from the Fig. that VM_1 becomes free at 3 and VM_2 becomes free at 5. So the VM_1 becomes free earlier than VM_2 but the finish time of VM_2 for C_1 is lesser than that of the VM_1. Hence, C_1 is allocated to VM_2. If two or more VMs show the same amount of finish time for a cloudlet, then the VM which becomes free earlier is chosen for that cloudlet.

4.3. Two-way balancing condition for cloudlet distribution

There are two mechanisms introduced for uniformly balanced cloudlet distribution. They are:

(a) Balance threshold

A variable "*Balance*" is maintained for balancing or distribution of cloudlets among the VMs. The variable *Balance* is initialized with a certain value.

If the total number of cloudlets initially present in global queue is GQinit and total number of VMs initially deployed is n, then the initial value of Balance is set in such a manner –

$$\text{Balance}_{init} = (GQ_{init}/n) + 1 \quad (6)$$

When the number of cloudlets allocated to a VM reaches the Balance value, that VM stops receiving new cloudlets. When the total number of cloudlets allocated to all the VMs reaches the current Balance value then the ADCB increments the value of Balance by its initial value. This process goes on until the global queue becomes empty.

In Fig. 6, at first assume the initial value of Balance is set to 2. After allocating cloudlets C0 to C5 to the VMs the number of cloudlets allocated to each VM becomes equal to the current Balance value but still new cloudlet C6 arrives. Then ADCB

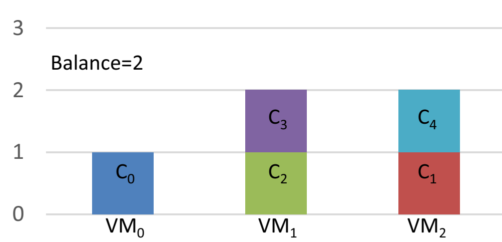

Figure 6 Before increment of "Balance" value.

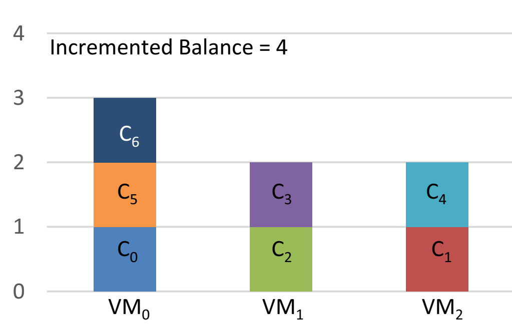

Figure 7 After increment of "Balance" value.

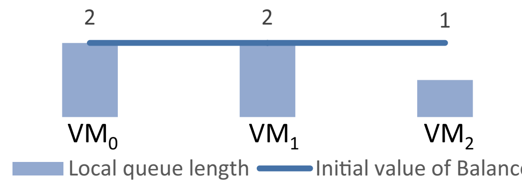

Figure 8 Variation of local queue length of VMs.

checks and increases the value of Balance to 2 + 2 = 4. This process is illustrated in Fig. 7.

(b) Local queue length limitation

The local queues of the VMs are not of equal lengths. The local queue length of a VM is set in following manner.

Let us consider, the total number of VMs be n. If n is even, set the ((n/2) + 1)th VM as median VM, and if n is odd, then set the ((n + 1)/2)th VM as the same. Let Mth VM be the median VM.

The local queue (LQ) length of all VMs from the 1st one to the Mth VM is set to the initial value of Balance.

The local queue lengths of the remaining VMs will be in decreasing order with a common difference of d, where

$$d = \text{Balance}/(n - M + 1) \quad (7)$$

In Fig. 8 the local queue lengths of three VMs are illustrated. Assume the initial value of Balance is 2. The number of VMs is 3. So the median VM is the ((3 + 1)/2)th VM or the second VM which is VM1. So the local queue lengths of VM0 and VM1 are set to the value of Balance which is 4, and the common difference d = (2/(3 – 2 + 1)) = 1. So the local queue length of the third VM is set to 2–1 = 1.

This helps to balance the cloudlet distribution because it is obvious that at most of the times the VMs with higher MIPS get exhausted sooner. This phenomenon could lead to an increase of the finish time. So to prevent the scenario the local queue length of the VM with higher MIPS will be shorter. This will distribute the cloudlets in a more balanced way at worst case scenario all of the local queues are exhausted then the arriving cloudlets will wait in the global queue.

5. Flow chart & time complexity

The entire procedure of the proposed RB2B is explained using a flowchart in Fig. 9.

5.1. Time complexity

The time complexity of RB2B is $O(mn)$ where m is the number of cloudlets and n is the number of VMs.

6. A small example of how RB2B works

To explain the working methodology of the RB2B a small example has been considered in this paper. Due to space constraint ten cloudlets and three VMs with limited length and processing speed respectively have been considered to demonstrate the working fashion of the proposed RB2B as shown in Table 4 and Table 5.

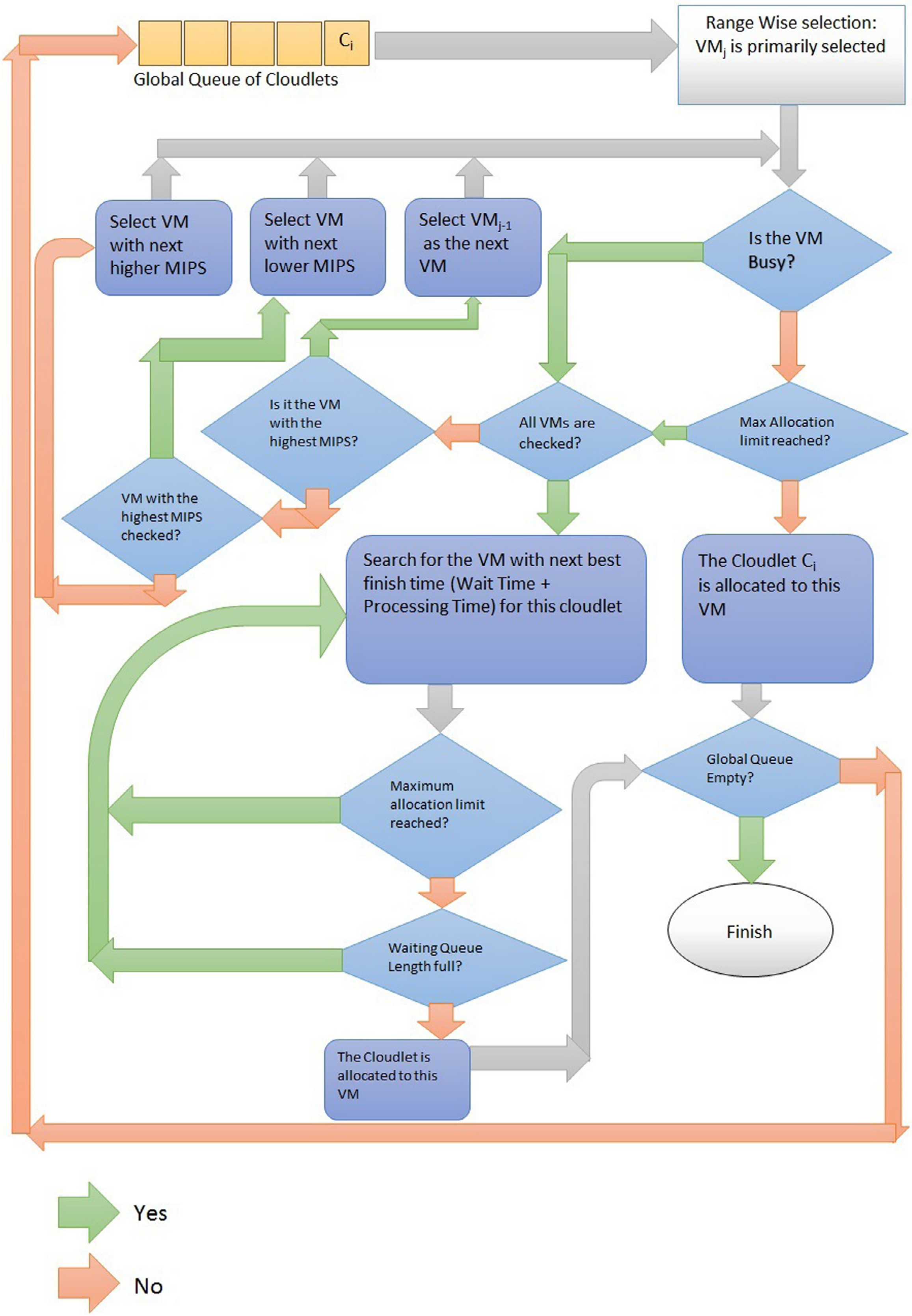

Figure 9 Flow Chart of RB2B.

Table 4 Reference cloudlet.

	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
Arrival time	0	1	1	2	2	3	5	6	8	9
Size (MI)	100	10	50	30	90	20	20	40	80	10

Table 5 Reference VM.

	VM_0	VM_1	VM_2
Processing speed (MIPS)	1	2	3

Table 6 Cloudlet length acceptability ranges of VMs.

VM	Lower limit	Upper limit
VM_0	10	10 + 15 = 25
VM_1	25 + 1 = 26	25 + 30 = 55
VM_2	55 + 1 = 56	55 + 45 = 100

Table 7 RB2B allocation procedure.

Cloudlets	Targeted VM	Targeted VM busy? (Y/N)	Allocated? (Y/N)	Queued	Finish time
C_0	VM_2	No	Yes	–	33.33
C_1	VM_0	No	Yes	–	10
C_2	VM_1	No	Yes	–	25
C_3	VM_1	Yes	No	VM_0	40
C_4	VM_2	Yes	No	VM_2	63.33
C_5	VM_0	Yes	No	VM_1	35
C_6	VM_0	Yes	No	VM_1	45
C_7	VM_1	Yes	No	VM_1	65
C_8	VM_2	Yes	No	VM_2	89.99
C_9	VM_0	Yes	No	VM_0	50

The initial value of balance = (10/3) + 1 = 4 following Eq. (6).

So, maximum four cloudlets can be allocated to each VM.

Here, C_{min} = 10 and C_{max} = 100 and $MIPS_{Total}$ = 1 + 2 + 3 = 6 according to Eq. (3).

So, the required ratio = (100 – 10)/6 = 90/6 = 15 as per Eq. (4).

Following the first phase of the RB2B Table 6 depicts the cloudlet acceptability length range for three VMs as per Eq. (5).

There are three VMs, so the median VM is (3 + 1)/2 = 2nd VM. So, the local queue length of first VM and the second VM are set to the initial value of *Balance* which is 4. The remaining VM i.e. the third VM will have a local queue length of (4 – (4/(1 + 1))) = 2 as discussed in 4.4.

Now, the length of the first cloudlet C_0 is 10 so VM_2 is primarily selected for it. After busy checking and checking of the balancing factor C_0 will be allocated to VM_2. Similarly C_1 and C_2 will be allocated to VM_0 and VM_1 respectively then C_3 will arrive and VM_1 is primarily selected for it. After all the checking ADCB will find that all the VMs are busy at that time so the VM with the earliest finish time is selected for C_3 to be queued in the local queue, provided that the Two-way balancing conditions are maintained, i.e. VM_1. This process will continue until the global queue becomes empty. The working procedure is shown in the following table (Table 7).

7. Performance evaluation

This section deals with analyzing the improvement of the results of the RB2B compared to RASA, Max–min, Min–min, RRA and CA. A reference stream of 1000 cloudlets with cloudlet length varying from 1000 MI to 100,000 MI and 10 VMs with processing speed varying from 1000 MIPS to 10,000 MIPS is used for the comparison. Due to space constraints, only this experimental scenario is being considered, but in experiments with larger number of cloudlets and VMs identical results have been found. The simulated results are evaluated and analyzed in several aspects. The performance was measured by setting up simulation environments (36) in

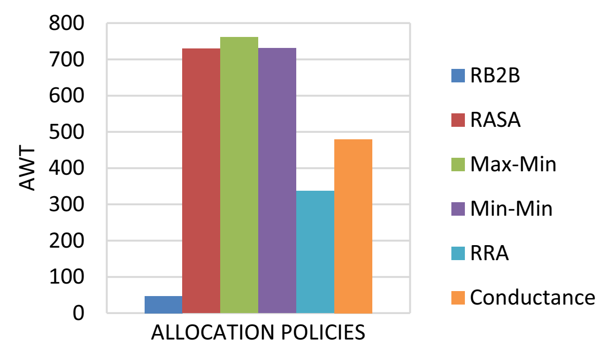

Figure 10 AWT comparison.

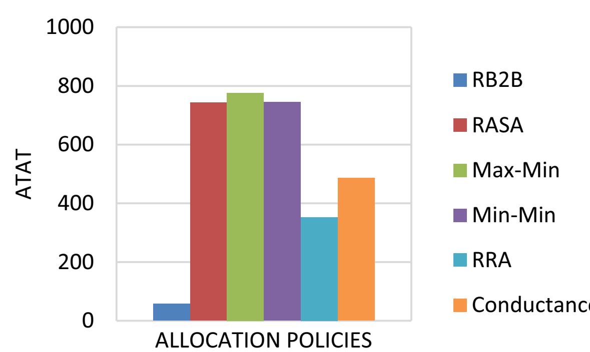

Figure 11 ATAT comparison.

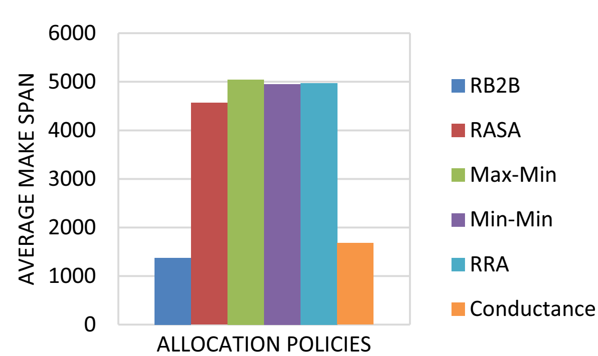

Figure 12 Average make span comparison.

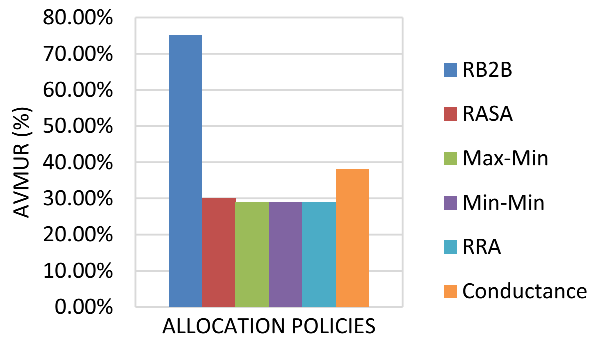

Figure 13 AVMUR comparison.

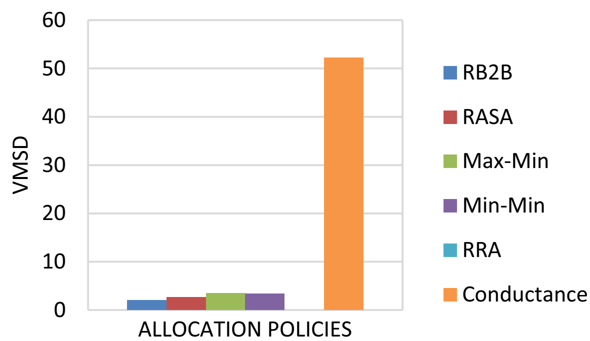

Figure 14 VMSD comparison.

CloudSim 3.0.3. The improvement of completion time and makespan is explained with an interpretation of the graphs in tabular form. Due to space constraint, the details of the reference cloudlet string and the VMs could not be given here. The improvement in the result of the proposed work indicates the efficiency of RB2B. Five parameters are taken into account to compare and analyze the performance of these algorithms. They are average waiting time (AWT), average turnaround time (ATAT), average make span, average VM utilization rate (AVMUR) and VM allocation standard deviation (VMSD).

Fig. 10 shows the comparison of average wait time. Average wait time is the average of each cloudlet's waiting time before getting allocated to a VM. An optimal algorithm will definitely try to minimize this parameter. The result clearly shows that AWT of RB2B is much less than five other algorithms. So RB2B gives better result for AWT.

Fig. 11 shows the average turnaround time comparison. This parameter is measured as the average of the turnaround time of all cloudlets, which is the time taken from a cloudlet's arrival to its completion. ATAT should also be minimized and the figure shows that RB2B again gives far better results than the other five algorithms.

The comparison result for average makespan has been mentioned in Fig. 12. The total time taken for a number of cloudlets to complete their execution is known as the makespan. This parameter should also be minimized and RB2B again proves its efficiency as per Fig.12.

Fig. 13 shows that the average VM utilization rate of RB2B is far better than that of the others. The AVMUR is a parameter which should always be maximized.RB2B gives more than 70% AVMUR where the 2nd best result given by CA doesn't even reach 40% of AVMUR.

The standard deviation of the number of Cloudlets allocated to the VMs is used to measure the deviations of the cloudlet distribution. Let z_i be the number of Cloudlets allocated to VM_i and μ be the mean of the number of allocated cloudlets to different VMs (z_i). So the standard deviation of cloudlet distribution for 'n' number of VMs is measured as:

$$\sigma = \sqrt{(1/n)\sum_{i=0}^{n-1}(z_i - \mu)^2}, \quad \mu = (1/n)\sum_{i=0}^{n-1}(z_i) \tag{8}$$

Fig. 14 shows the comparison of VM allocation standard deviation. If the total number of cloudlets arrived is $((g \times n) + h)$, where g and h are two arbitrary integer constant, then the VMSD of RRA would be

Table 8 Comparison result for performance evaluation.

Parameters	RB2B	RASA	Max–min	Min–min	RRA	Conductance
AWT	47.17	729.0	761.5	730.3	337.3	479.6
ATAT	57.68	743.4	776.5	745.1	352.1	486.
Average make span	1373.3	4567.2	5046.3	4954.2	4976.2	1678.8
AVMUR (%)	75	30	29	29	29	38
VMSD	2	2.63	3.5	3.5	0	52.3

Table 9 Percentage of Improvement.

Parameters	RASA	Max–min	Min–min	RRA	Conductance
AWT (%)	93.5	93.8	93.5	86.0	90.2
ATAT (%)	92.2	92.5	92.3	83.6	88.1
Average make span (%)	69.9	72.8	72.2	72.4	18.2
AVMUR (%)	45	46	46	46	37
VMSD (%)	23.9	42.9	42.9	–	96.2

Table 10 A comparative study of existing algorithms & proposed RB2B algorithm.

	Min–min	Max–min	RASA	RRA	Conductance	RB2B
Nature of Allocation	Static	Static	Static	Dynamic	Static	Dynamic
Advantages	The idle time of the VMs is almost zero	Removes the disadvantages of Min–min	Advantages of both Min–min & Max–min	Much simpler to implement	Makespan is lesser than other four policies	Better in every case compared to other five policies
Disadvantages	(i) Lacks uniform resource utilization (ii) Not optimal when the number of smaller cloudlets is greater	(i) Makespan is greater than the others. (ii) Increases the waiting time of smaller cloudlets	Disadvantages of both Min–min & Max–min	Large cloudlets are often assigned to the VMs with low MIPS increasing waiting time and the response time	Resource utilization is highly non-uniform and wastage of resource.	A little bit complicated and implementation might not be as simple as the other five policies
Time Complexity	$O(mn^2)$	$O(mn^2)$	$O(mn^2)$	$O(mn)$	$O(mn^2)$	$O(mn)$

$$\sigma_{\mathrm{RRA}} = (1/n)\sqrt{(h(n-h))} \tag{9}$$

It can be shown that the maximum value of VMSD for RRA will be 0.5. However, RB2B gives the second best result here.

Table 8 depicts the performance analysis based on the parameter values.

Table 9 shows the rate of improvement of RB2B over the other algorithms in terms of different performance metrics.

The Table 10 shows a comparative summary of the six algorithms.

8. Conclusions and future scope

The cloud computing is an immense area of research and cloudlet allocation plays a key role in good service delivery. There is a huge scope of development in this area. This paper presents a three phase cloudlet allocation algorithm which overcomes the major drawbacks of existing allocation policies very efficiently. The proposed RB2B works in a layered approach associated with two major balancing conditions where every layer tries to maximize the chances of better allocation. It measures the length of the cloudlet and accordingly chooses a VM following a cloudlet size acceptability range. If the chosen VM is available then the Balance threshold condition will be checked. If the condition is satisfied the cloudlet will be allocated to the targeted VM. If such suitable VM is available, the cloudlet is allocated. But if still no suitable VM is found then the third phase of RB2B will commence. Then it searches for a suitable VM and allocates the cloudlet to the local queue of this VM. Thus, the goal of cloudlet allocation and better resources manipulation could be achieved.

We are planning for developing this proposed RB2B policy with non-linear optimization technique as future work in association with soft computing. That will deal the challenges associated with cloudlet allocation in an intelligent way by adopting the performance learning mechanism. Some factors like the ratio of VMs and cloudlets, two balancing factors, the distribution pattern of size of the incoming cloudlets can be trained and analyzed so that the overall performance can be highly upgraded. Eventually this may also improve the cost and round trip time of the entire system.

Acknowledgments

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

References

Afzal, A., Mc Gough, A.S., Darlington, J., 2008. Capacity planning and scheduling in Grid computing Environment. J. Future Gener. Comput. Syst. 24 (5), 404–414. http://dx.doi.org/10.1016/j.future.2007.07.004.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.A., 2010. View of cloud computing. Commun. ACM 53 (4), 50–58. http://dx.doi.org/10.1145/1721654.1721672.

Aziz, A., El-Rewini, H., 2008. On the use of meta-heuristics to increase the efficiency of online grid workflow scheduling algorithms. Cluster Comput. 11 (4), 373–390. http://dx.doi.org/10.1007/s10586-008-0062-y.

Banerjee, S., Adhikari, M., Kar, S., Biswas, U., 2015. Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud. Arabian J. Sci. Eng., 1319-8025 40 (5), 1409–1425. http://dx.doi.org/10.1007/s13369-015-1626-9.

Belalem, G., Tayeb, F.Z., Zaoui, W., 2010. Approaches to improve the resources management in the simulator CloudSim. In: Zhu, R., Zhang, Y., Liu, B., Liu, C. (Eds.), . In: ICICA 2010. LNCS, 6377. Springer, Heidelberg, pp. 189–196.

Benjamin Khoo, B.T., Veeravalli, B., Hung, T., Simon See, C.W., 2007. A multi-dimensional scheduling scheme a Grid computing environment. J. Parallel Distrib. Comput. 67 (6), 659–673. http://dx.doi.org/10.1016/j.jpdc.2007.01.008.

Bhatia, W., Buyya, R., Ranjan, R., 2010. CloudAnalyst: a CloudSim based visualmodeller for analysing cloud computing environments and applications, 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446–452.

Bittencourt, L.F., Sakellariou, R., Madeira, E.R.M., 2010. DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm. In: 18th Euromicro International Conference on Parallel, and Network-Based Processing (PDP), 34, p. 27. http://dx.doi.org/10.1109/PDP.2010.56, 17-19 Feb. 2010.

Braun, T.D., Siegel, H.J., Maciejewski, A.A., Hong, Y., 2008. Static resource allocation for heterogeneous computing environments with cloudlets having dependencies, priorities, deadlines, and multiple versions. J. Parallel Distrib. Comput. 68 (11), 1504–1516. http://dx.doi.org/10.1016/j.jpdc.2008.06.006.

Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I., 2009a. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 2009 25 (6), 599–616.

Buyya, R., Ranjan, R., Calheiro, R.N., 2009b. Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: challenges and opportunities <http://arxiv.org/pdf/0907.4878>

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.

Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R., 2010. CloudSim: A Toolkit for Modelling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Published online 24 August 2010 in Wiley Online Library (wileyonlinelibrary.com). http://dx.doi.org/10.1002/spe.995.

Carino, R.L., Banicescu, I., 2008. Dynamic load balancing with adaptive factoring methods in scientific applications. J. Supercomput. 44 (1), 41–63. http://dx.doi.org/10.1007/s11227-007-0148-y.

Chatterjee, T., Ojha, V.K., Adhikari, M., Banerjee, S., Biswas, U., Snasel, V., 2014. Design and implementation of a new datacenter broker algorithm to improve the QoS of a cloud. In: Proceedings of ICBIA 2014, Advances in Intelligent Systems and Computing, 303. Springer International Publishing, Switzerland, pp. 281–290.

Dong, F., Luo, J., Gao, L., Ge, L., 2006. A grid task scheduling algorithm based on QoS priority grouping, The Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06). IEEE, pp. 58–61. http://dx.doi.org/10.1109/GCC.2006.7.

El-kenawy, E.T., El-Desoky, A.I., Al-rahamawy, M.F., 2012. Extended Max–min scheduling using petri net and load balancing. Int. J. Soft Comput. Eng. (IJSCE), 2231-2307 2 (4), 2231–2307.

Elmroth, E., Tordsson, J., 2008. Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions. J. Future Gener. Comput. Syst. 24 (5), 585–593. http://dx.doi.org/10.1016/j.future.2007.06.001.

Fu, Xiong, Zhou, Chen, 2015. Virtual machine selection and placement for dynamic consolidation in Cloud computing environment. Front. Comput. Sci. http://dx.doi.org/10.1007/s11704-015-4286-8.

García-Valls, Marisol, Cucinotta, Tommaso, Lu, Chenyang, 2014. Challenges in real-time virtualization and predictable cloud computing. J. Syst. Architect. 60, 726–740. http://dx.doi.org/10.1016/j.sysarc.2014.07.004.

George Amalarethinam, D.I., Muthulakshmi, P., 2011. An Overview of the scheduling policies and algorithms in Grid Computing. Int. J. Res. Rev. Comput. Sci. 2 (2), 280–294.

Gulati, A., Chopra, R.K., 2013. Dynamic round robin for loa balancing in a cloud computing. IJCSMC, 2320-088X 2 (6), 274–278.

He, X., Sun, X., Laszewski, G.V., 2003. QoS guided Min–min heuristic for grid cloudlet scheduling. J. Comput. Sci. Technol. 18, 442–451.

Iordache, G.V., Boboila, M.S., Pop, F., Stratan, C., Cristea, V., 2007. A decentralized strategy for genetic scheduling in heterogeneous environments. Multiagent Grid Syst. 3 (4), 355–367.

Khanli, L.M., Analoui, M., 2007. Grid_JQA: a QoS guided scheduling algorithm for grid computing, The Sixth International Symposium on Parallel and Distributed Computing (ISPDC'07). IEEE, p. 34, 10.1109/ISPDC.2007.25.

Khanli, L.M., Analoui, M., 2008. Resource scheduling in desktop grid by grid-JQA, The 3rd International Conference on Grid and Pervasive Computing. IEEE, pp. 25–28. http://dx.doi.org/10.1109/GPC.WORKSHOPS.2008.27, 25-28 May 2008.

Kumar, P., Dutta Pramanik, P.K., 2012. Host selection methodology in cloud computing environment. Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET), 2278-1323 1 (8).

Livny, M., Melman, M., 2011. Load balancing in homogenous broadcast distributed systems, Proceedings of the ACM Computer Network: Performance Symposium, pp. 47–55.

Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R., 1999. Dynamic mapping of a class of independent cloudlets onto heterogeneous computing systems, 8th IEEE Heterogeneous Computing Workshop (HCW '99), pp. 30–44, San Juan, Puerto Rico.

Makelainen, M., Khan, Z., Hanninen, T., Saarnisaari, H., 2014. Algorithms for opportunistic load balancing cognitive engine. In: Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM), 2014 9th International Conference, 2-4 June 2014, pp. 125–130.

Parsa, S., Entezari-Maleki, R., 009. RASA: a new grid cloudlet scheduling algorithm. World Appl. Sci. J. 7, 152–160 (Special Issue of Computer & IT).

Pop, F., 2008. Communication model for decentralized meta-scheduler in Grid Environments. In: Proceedings of The Second International Conference on Complex, Intelligent and Software Intensive System, Second International Workshop on P2P, Parallel, Grid and Internet computing – 3PGIC-2008 (CISIS'08). IEEE Computer Society, Barcelona, Spain, ISBN 0-7695-3109-1, pp. 315–320. http://dx.doi.org/10.1109/CISIS.2008.131, March 4–7, 2008.

Radulescu, A., Van Gemund, A.J., 1999. On the complexity of list scheduling algorithms for distributed memory systems. In: Proceedings of the 13th International Conference on Supercomputing (Rhodes, Greece, June 20 – 25, 1999). ICS '99. ACM, New York, NY, pp. 68–75.

Rashid, M., Banicescu, I., Carino, R.L., 2008. Investigating a dynamic loop scheduling with reinforcement learning approach to load balancing in scientific applications. In: Proceedings of the 2008 international Symposium on Parallel and Distributed Computing (July 01- 05, 2008). ISPDC. IEEE Computer Society, Washington, DC, pp. 123–130. http://dx.doi.org/10.1109/ISPDC.2008.25.

Ullah Munir, E., Li, J., Shi, S., 2007. QoS sufferage heuristic for independent task scheduling in grid. Inf. Technol. J. 6 (8), 1166–1170. http://dx.doi.org/10.3923/itj.2007.1166.1170.

Wang, T., Zhou, X., Liu, Q., Yang, Z., Wang, Y., 2006. An adaptive resource scheduling algorithm for computational grid. In: APSCC, 2006, Asia-Pacific Conference on Services Computing. 2006 IEEE, Asia-Pacific Conference on Services Computing. IEEE, pp. 447–450. http://dx.doi.org/10.1109/APSCC.2006.22.

Zhang, Y., Koelbel, C., Kennedy, K., 2007. Relative performance of scheduling algorithms in grid environments. In: Proceedings of the Seventh IEEE international Symposium on Cluster Computing and the Grid (May 14 - 17, 2007). CCGRID. IEEE Computer Society, Washington, DC, pp. 521–528. http://dx.doi.org/10.1109/CCGRID.2007.94.