Range wise busy checking 2-way imbalanced algorithm for cloudlet allocation in cloud environment

Mohammed Alanzy1, Rohaya Latip1,2, Abdullah Muhammed1

1Computer Sciences and Information Technology, Universiti Putra Malaysia, 2Institut Penyelidikan Matematik (INSPEM), Universiti Putra Malaysia, 43400 Serdang, Malaysia
m.alanzy.cs@gmail.com, rohayalt@upm.edu.my, abdullah@upm.edu.my.

Abstract. Cloud computing considers as a new business paradigm and a popular platform over the last few years. Many organizations, agencies, and departments consider responsible tasks time and tasks needed to be accomplished as soon as possible. These agencies counter IT issues due to the massive arise of data, applications, and solution scopes. Currently, the main issue related with the cloud is the way of making the environment of the cloud computing more qualified, and this way needs a competent allocation strategy of the cloudlet. Thus, there are huge number of studies conducted with regards to this matter that sought to assign the cloudlets to VMs or resources by variety of strategies. In this paper we have proposed range wise busy checking 2-way imbalanced Algorithm in cloud computing. Compare to other methods, it decreases the completion time to finish tasks' execution, it is considered the fundamental part to enhance the system performance such as the makespan. This algorithm was simulated using Cloudsim to give more opportunity to the higher VM speed to accommodate more Cloudlets in its local queue without considering the threshold balance condition. The simulation result shows that the average makespan time is lesser compare to the previous cloudlet allocation strategy.

1. Introduction
Cloud computing is the latest trend in computer science, and it is expected to be the future of the 21st century technology. Cloud computing is mostly common for its deployment of shared resources proficiently. The distribution of the cloudlets to the fittest resource, also known as virtual machines, is one of the main requirements in the environment of cloud computing [1]. A typical cloud environment holds a module called Data Centre Broker which carries out the controlling of the entire Data Centre, including the allocation of the cloudlets to the virtual machines. Similar to most computing performances, enhancement and optimization of the allocation algorithm will always be an open option. To engineer an effective cloudlet allocation algorithm is considered to be a challenging research area [2]. Numerous policies have been suggested, examined and compared on heterogeneous parallel computing environments. The introduction of a new mechanism was made; effective aggregated computing power (EACP) which improves and enhances the performance [3]. On the other hand, the adaptive weighted factoring (AWF) is responsible for scheduling parallel loops [4]. The active loop scheduling via strengthened learning is helpful and effective the most in time stepping scientific platforms with numerous steps [5]. The scheduling strategies for grid surroundings take advantage of a few methods that are alike, yet, some difference to the mechanism of cloudlet allocation strategies are witnessed [6].
Genetic algorithms are also used for scheduling [7], the opportunistic load balancing heuristic randomly picks one of the batch’s cloudlets and distributes it to the next virtual machine predicted to be available, not taking into consideration the cloudlet’s anticipated completing time on that virtual machine, causing a very deprived makespan [8][9]. The minimum execution time (MET) distributes every cloudlet picked randomly to the virtual machine that has the lowest execution time causing the inequality of load through the virtual machines [10]. Minimum completion time (MCT) heuristic, distributes every cloudlet to the virtual machine with the least execution time for the cloudlet [10]. It figuratively and practically merges the advantages of OLB and MET. The quality of service guided Min-Min allocates cloudlets that need more bandwidth. The QOS priority grouping scheduling presents the significance of deadlines [11]. The Grid-IQA scheduling solution takes advantage of a combination formula which chains the parameters together along with weighting factors, in order to evaluate QOS [12][13]. Alternative user oriented scheduling algorithm that takes advantage of using progressive reservation and resource assortment methods reduces the completion time of single cloudlets without bearing in mind the makespan [14]. The multiple resource scheduling (MRS) algorithm studies the abilities of the system and the resource necessities of cloudlets as common aspects [15]. Within cloud computing settings, the majority of the distribution regulations load some precise resources moderately more heavily, parting other resources either idle or with the minimum burden [16]. As a cause, load balancing is a critical concern in cloud computing which touches the performance of the cloud service supplier. The objective of this work is to develop and improve the current allocation strategies in this area by creating a new cloudlet allocation strategy based on (RB2B) algorithm [17] which mainly concentrates on decreasing the makespan, and simultaneously, it optimizes the virtual machine utilization to a notable amount by allocating the cloudlets to the virtual machines in the best way possible or most fairness way. The proposed algorithm is merged in the Data Centre Broker (DCB) module.

2. Related Works

Some resource allocation polices that consider the cornerstone for the proposed algorithm, described as follow:

2.1. Min-Min
Min-Min [18][19][20] at first, a matrix is occupied for all the free cloudlets. Two phases occur in Min-min, initially, the first phase, the set of minimum computing time for every single cloudlet within the matrix is calculated and tackled. As for the second phase, the cloudlet with the total least computation time is picked from the matrix and allocated to the analogous virtual machine. After that, the given cloudlet is taken out from the matrix and admissions of the matrix are improved consequently. The course of Min-min would be recurrent until the matrix runs out of cloudlets, thus, the cloudlets within the matrix are all mapped. It takes \(O(m^2n)\) time for this algorithm, where \(m\) is the number of virtual machines and \(n\) is the number of cloudlets. The Min-min algorithm has a lack in the utilization of the uniform resource due to select smaller Cloudlets at the beginning which makes use of the virtual machine with higher MIPS. Thus, the scheduling is not performing when the number with larger Cloudlets is less than the number of the small Cloudlets.

2.2. Max-Min
This algorithm [18] is very much like Min-min. However, there is a difference that stands out within the second phase. In Max-min, it first picks the cloudlet with the highest computation time from the matrix and allocates it to the virtual machine that has the selected cloudlet producing minimum time to complete. Like the previous algorithm Min-min, this algorithm also takes \(O(mn^2)\) time where \(m\) and \(n\) are numbers of virtual machines and number of cloudlets respectively. To solve the weakness of Min-min, Max-min algorithm is introduced that firstly executes the larger Cloudlet. However, the overall makespan may increase as well as the cloudlets with smaller length and its waiting time will increase.

2.3. Resource Aware Scheduling Algorithm (RASA)
The algorithm RASA [18]combinations of both Min-min and Max-min algorithms. If the virtual machines count is odd, the Min-min algorithm is applied, otherwise, Max-min is applied. The entire process may be distributed into a number of sequences, where two cloudlets are distributed to suitable virtual machines by any of the two policies, nonetheless. The rule dictates if the first cloudlet of current round is distributed to a virtual machine using the strategies of Min-min, the following cloudlet would be distributed by the max-min strategy. In the upcoming round, the allocation of the cloudlet initiates with an algorithm unlike the one from the previous round, to make it clearer, if the first round begins with max-min, the other initiates with Min-
min and vice versa. The experimental results indicate that if the amount of available resources is an odd number, then the application of the Min-min algorithm produces better results. Otherwise, it would be better to start with max-min strategy. Max-min and Min-min are replaced instead to result in successive implementation of both, small and large cloudlets on different virtual machines and hence, the times waited for small cloudlets in Max-min and the large cloudlets in Min-min are overlooked. RASA does not contain any time-consuming instructions, the complexity of time within RASA is 0(mn²) where m is the number of VMs and n is the number of cloudlets. RASA has weakness of both algorithms Max-min and Min-min, in spite of its manner to minimize them.

2.4. Round Robin Allocation (RRA)
In RRA [18][21][22], VMs (VM0, VM1, and VM2) was introduce in the system for the allocation strategy. According to this policy, cloudlet C0 allocated to VM0, C1 allocated to VM1, C2 allocated to VM2, C3 allocated to the VM0 and C4 allocated to VM1.

It distributes the cloudlets to the virtual machine that is available first, for instance, assume there are four cloudlets (C0, C1, C2, C3, C4) and three virtual machines {VM0, VM1, VM2} respectively available in the system. Based on this strategy, cloudlet C0 is allocated to VM0, C1 allocated to VM1, C2 to VM2, C3 is allocated to VM0 and C4 to VM1 respectively.

Round Robin Algorithm may assign large Cloudlets to the virtual machines with slowest processing speed thus, the processing time will get a long time to finish the execution that will Cause increasing the waiting time and the response time of the cloudlets as well. In addition, the highest virtual machine sometimes gets the smaller tasks that cause wasting in the resource utilization with reducing the whole performance.

2.5. Conductance algorithm (CA)
The CA algorithm [23] takes into consideration the size of every virtual machine as a pipe, the conductance is calculated using equation (1) as shown below, each virtual machine’s processing speed ratio to the total of processing speeds of the virtual machines within the system. Dispensation speed of VMj is measured in million instructions per second (MIPS), presentation, however, is using MIPSj. Conductance of VMj is represented by Conductance j.

\[
\text{Conductance } j = \frac{\text{MIPS}_j}{\sum_{i=1}^{n} \text{MIPS}_j} 
\]  

(1)

Subsequent to the calculation of conductance, the number of cloudlets distributed to VMj is calculated by the multiplication of the Conductance j of that specific VMj with the length of cloudlet list. In order to ensure the strip length of every VMj, where strip length shall be represented by strip length, the use of equation (2) as shown below is implemented, it ensures the number of cloudlets processed by the virtual machine.

\[
\text{striplength } = \text{Conductance } \times (\text{length of cloudlet list}) 
\]  

(2)

Conductance Algorithm here might get wasted in the resource utilization due to the low VMs processing speed, which sometimes release rapidly and vice versa in the high VMs processing speed especially that the high VM has a length of the Cloudlets that is so large.

2.6 Range wise Busy-checking 2-way Balanced (RB2B)
It is a three-phase allocation algorithm [17], it is as follows:

(a) VM categorization phase

(b) Two round busy checking phase

(c) Cloudlet still not allocated (CSNA) phase.

A VM is considered appropriate only when it is not busy and it satisfies two-way balancing conditions. Thus, the process of RB2B is designated as follows:

The VMs are created and allocated to the host(s) and are reorganized in increasing order of processing speed (MIPS). A cloudlet arrives from the global queue (GQ) to the ADCB where the cloudlet allocation algorithm is executed. In the first phase, ADCB measures the length of the cloudlet and correspondingly chooses a VM following a cloudlet size acceptability range. If the chosen VM is available then ADCB will check it for a condition defined as Balance threshold. 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 [24] 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. There are not many restrictions of available algorithms and suggestions of the new algorithm: Min-min algorithm does not have uniform resource use in a way that makes it initiates by picking smaller cloudlets which take advantage of both virtual machines with more processing speed. Thus, the scheduling does not reach an optimum state, when the number of smaller cloudlets surpasses the bigger ones, to solve this issue, Max-min algorithm starts by scheduling the bigger cloudlets initially, however, sometimes, because of the execution of bigger cloudlets first, there can be an increase in the makespan. In addition to that, Max-min also increases smaller cloudlets’ waiting time. RASA, on the other hand, has disadvantages from both sides, Min-min and Max-min, even though it rises to make them less. However, in RRA, big cloudlets are usually allocated to the virtual machines with low MIPS causing the delay in execution and the increase in the waiting time and response time within the cloudlets. Occasionally, it can occur that the most powerful virtual machines receive the smallest cloudlets making its resources use go waste, simultaneously depriving the overall performance, in the case of Conductance, low MIPS virtual machines occasionally achieve freedom quickly resorting to the waste of resources. The proposed algorithm RB2I performs in a way that the cloudlet will always be assigned to the most suitable virtual machine, based on the size of the cloudlet and the number of cloudlets distributed to the virtual machines is uniform to a precise extent. Making the percentage of resource allocation maximum and enhancing the completion time for every cloudlet so they become minimized when compared to other policies. This results in the minimization of the finish time, which brings us to the outcome of minimization of the makespan of the cloudlets.

3. Experiment Setup
The strategy proposed was simulated in Cloudsim, as illustrated in the figure 1. This model would interact with each other to act as a real system and each entity is described as below:
CIS (cloud information service) kind of registry that contains the resource available in the cloud
DC (a resource of the cloud) holds some hosts and it has some kind of hardware configurations, that can be a number of a processing element (PE it maybe (1,2,3)) and some RAM it might 512 MB, 1GB, 4GB in addition to the bandwidth.
The host might have the hardware configuration (Ram, BW and etc) and needs to register these characteristics to the CIS Broker. It needs to submit the task to DC and it has the characteristics of the DC from the CIS. Once the broker gets the characteristics of the DC and hosts, it interacts directly with the DC to assign the cloudlet to the VM.

![Figure 1. CloudSim model](image)
4. Range wise Busy-checking 2-way Imbalanced
The proposed work is done in an imbalanced way based on the previous algorithm i.e. RB2B but, without considering the threshold balance and due to that, the work speeds up the makespan and keeps the VMs active.

As shown in figure (2), the VMs are sorted in the ascending manner, i.e. the VMi+1 is faster than VMi, that will be occupied by the cloudlets depending on the algorithm phases after getting calculated in the data center broker (e.g. previously determined). The whole process details will be shown below.

A three-phase allocation algorithm consists of three phases as the following:
(a) Primarily selected of the VMs,
(b) Two full round busyness checking,
(c) Cloudlet still not assigned.

The algorithm will be distributing Cloudlets among the VMs in a justice way. The process of the algorithm in a nutshell. Creating the VMs and assigning it to the Host(s) with classifying in increasing arrangement based on MIPS (the speed of the processing). Generating the Cloudlets and sending it to the Datacenter Broker (DCB) whereas the algorithm will get implemented. After that, the phases will initiate: Firstly, some equations will calculate in the DCB and determine the upper and lower limit for each VM to assign for it primarily if that VM was not busy. If the planned VM was unavailable (busy) for that specific cloudlet, then the second part of the algorithm will be running start.

In the second part, the cloudlet checks the VMs for availability, the last phase will begin. Datacenter Broker will go through which VM has least finish time of its task in addition to the specific Cloudlet, the least one will get the cloudlet in its local queue. The phases are described below in more detail.

4.1 Primarily selected of the VMs
The first step of the algorithm is to select the suitable VM to the cloudlets based on its length. Thus at the beginning, the Data Center Broker (DCB) will be determined for each VM the acceptance range that illustrated below:

1. Determine the minimum and maximum cloudlet length (Cmax and Cmin):
2. Compute the total processing speed of the VMs that calls MIPS. As in equation (3) Total Processing Speed:

\[
MIPS_{total} = \sum_{i=0}^{n-1} MIPS_i
\]  

(3)

3. Compute the difference (X) between Cmax and Cmin over the total of MIPS as in equation (4), the different X,

\[
X = (C_{max} - C_{min}) / MIPS_{total}
\]  

(4)

By assuming Fi is a MIPS of VMi and VMi are sorted in increasing order of the MIPS speed therefore MIPSi+1>MIPSi. Thus, the distribution of the processing speed range will demonstrate in figure 3.
Figure 3. Range distributions among VMs

Compute the upper and lower limits of Cloudlets length that accepted by each VM by a linear sequential equation as per equation (5) and equation (6) respectively:

\[ C_{\text{min}} + F_0x + F_1x + \ldots + F_m^e \]  
\[ C_{\text{min}} + F_0x + F_1x + \ldots + F_m - 1X + 1 \]  

Table 1 below shows the acceptability range equations:

| #VM  | Parameters | Number of parameters |
|------|------------|----------------------|
| VM_0 | \( C_{\text{min}} \) | \( C_{\text{min}} + F_0x \) |
| VM_1 | \( C_{\text{min}} + F_0x + 1 \) | \( C_{\text{min}} + F_0x + F_1x \) |
| VM_m | \( C_{\text{min}} + F_0x + F_1x + \ldots + F_m - 1X + 1 \) | \( C_{\text{min}} + F_0x + F_1x + \ldots + F_m^e \) |
| VM_n-1 | \( C_{\text{min}} + F_0x + F_1x + \ldots + F_n - 2X + 1 \) | \( C_{\text{min}} + F_0x + F_1x + \ldots + F_n - 1 = C_{\text{max}} \) |

At the end of this phase, the DCB will target for each cloudlet a suitable VM based on the lower and upper limit and term it as targetedVM.

4.2 Two round busy checking phase
This phase will initiate once the Cloudlet finds the targetedVM is busy and not available at the time of the cloudlet arrival thus, two round of busyness checking will start:

The first round the DCB will check the next higher processing speed VM if it is busy or not. This process will continue until the cloudlet reaches to the highest processing speed VM, if all higher VMs are busy then the second round will begin. In second round, the DCB will check the next lower processing speed VMs, if it is busy or not. As illustrated in Figure 4.

Figure 4. Two round phase.

In this phase, the DCB will check the higher MIPS VMs one by one until finding a suitable VM to the cloudlet. If the DCB could not find the suitable VM, then the second round of this phase will initiate. The DCB will do same manner by trying to find any suitable VM starting from the higher to the lower MIPS VMs. If it is found, then the DCB will submit the cloudlet to the suitable and available VM. Otherwise, phase three will commence.

4.3 Cloudlet-still-not-allocated phase (CSNA)
In third phase of the algorithm, in case the cloudlet did not submit during the previous phases, then it will move to this phase to get submission by calculating the finish time of that cloudlet on all VMs, in addition to the waiting time for the VMs and make comparisons among the result of that to see which VMs will give least finish time to submit the cloudlet in its local queue.
RB2I Algorithm pseudo code

```java
class RB2I {
    set The MIPS speeds
    calculate the sum of MIPS
    set parameters for the VM
    for i=0 to no of VMs
        Create VM[i] //VMi<VMi+1
    }
    Set parameters for Cloudlets
    for i=0 to no of Cloudlets
        Create heterogenous Cloudlets
    }

    DataCenterBrocker Class {
        Determine the upper & lower limits for the Cloudlet
        PA=primary assign the VM to the Cloudlet
        Busy=vm.getTotalUtilizationOfCpu(CloudSim.clock())
        While (i<Cloudlet.size)
            if (PA!=Busy)
                submit the cloudlet
            if (PA==Busy)
                if (Cloudlet == upperlimit)
                    Check the VM next slower MIPS if not Busy then submit to it
                    if (VMs==checked)
                        {  
                            Go to Phase 3
                        }
                    else
                        CPA=current PA
                        for j=CPA+1 to no of VM
                            if (VM[j] != Busy)
                                {  
                                    submit to VM[j]
                                }
                        for j=CPA-1 to VM.0
                            if (VM[j] != Busy)
                                submit to VM[j]
                        if (VMs==checked)&&(PA==Busy)
                            {  
                                Go to Phase 3
                            }
        phase 3
        for (i=0 to no of VMs)
            {  
                FT[i]=finish time for the VM[i]
                FTS[i]=finish time for the specific task on that VM
                C[i]=FT[i]+FTS[i]
            }
        compute the min C time
        submit
    }
```
4. Result and discussion
The result shows the performance evaluation for the proposed work with the algorithm RB2B and the comparison between them. Due to the difficulties in the testing, handling and assessing on a real system, this research procedure has been done on the discrete event simulator called Cloudsim to emulate the actual environment. The main parameters applied to assess the proposed algorithm are shown in table (2).

| Parameters                | Number of parameters |
|---------------------------|----------------------|
| Number of Cloudlet        | 1000                 |
| Cloudlet length           | (1000-100,000) MI    |
| Number of VMs             | 10                   |
| MIPS for VM               | (1000-10,000) MIPS   |
| Number of DataCenters     | 1                    |
| Number of Hosts           | 2                    |
| pesNumber (Number of CPUs)| 1                    |
| num_user (Number of users)| 1                    |
| Host(s) Storage           | 1,000,000 MB         |
| Host(s) memory            | 4096 MB              |

The makespan is the maximum finish time (FTi) among all received tasks per time unit. Or in other words, it is the last finish time to the specific VM per time unit. It measures the proficiency and the intelligence of the way of the scheduling algorithm in mapping the tasks to the cloud suppliers. From mapping proficiency point of view, this parameter considers the reflector for the behavior of algorithms. Figure (5) depicts this parameter. It shows a set of 1000 cloudlets with length ranging from 1000 to 100000 MI (Million instruction) by using heterogeneous manner and set of 10 VMs with different processing speed ranging from (1000-10,000) MIPS.

The proposed work showed an enhancements of 34.53% in the average makespan time compared with the balanced algorithm (RB2B).

5. Conclusion
The cloud computing is a massive field of study hence the Cloudlet allocation takes a vital role in delivery the service in an appropriate way. Thus, there is an immense scope of improvement in this field, this work presents a three stages of cloudlet allocation algorithm that overcomes the existing allocation strategies weaknesses in a proficiently manner where in the three stages of RB2I tried to allocate the cloudlet in the suitable VM, hence in the first stage measures the length for each Cloudlet and accordingly elects VM pursuing a cloudlet size acceptability limit. If the condition is satisfied, the cloudlet will be allocated to the targeted VM. If such suitable
VM is available i.e. it will check the higher VMs by comparing the low VMs, once the cloudlet find the suitable and available VM, the cloudlet is allocated. However if it is still not suitable and available (all VMs are Busy), then the third stage of RB2I will initiate. Then, it examines for an appropriate VM by computing the consuming time to finish the cloudlet execution. Then it allocates the cloudlet to the local queue of chosen VM. Therefore, the objective of cloudlet allocation and well resources manipulation could be succeeded in the less average makespan time.

References
[1] Fu X and Zhou C 2015 Virtual machine selection and placement for dynamic consolidation in Cloud computing environment Front. Comput. Sci. 9 322
[2] Zhang Y, Koelbel C and 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. CCGRID. IEEE Computer Society, Washington, DC 521–8
[3] Radulescu A, Van Gemuend A 1999 On the complexity of list scheduling algorithms for distributed memory systems In: Proceedings of the 13th International Conference on Supercomputing ICS ’99. ACM, New York, NY 68–75
[4] Carino R and Banicescu I 2008 Dynamic load balancing with adaptive factoring methods in scientific applications. J. Supercomput. 44 41–63
[5] Rashid M, Banicescu I and 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 123–30
[6] Aziz A and El-Rehwini H 2008 On the use of meta-heuristics to increase the efficiency of online grid workflow scheduling algorithms Cluster Comput 11 373–90
[7] 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 Systems, Second International Workshop on P2P, Parallel, Grid and Internet computing – 3PGIC-08 (CISIS’08). IEEE Computer Society, Barcelona, Spain, ISBN 0-7695-3109-1 315–20
[8] Braun T D, Siegel H J, Maciejewski A A and Hong Y 2008 Static resource allocation for heterogeneous computing environments with cloudlets having dependencies, priorities, deadlines, and multiple versions J. Parallel Distrib. Comput 68 1504–16
[9] Makelainen M, Khan Z, Hauxinen T and Saarinsaari 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 125–30
[10] George Amalarethinam D I and Muthulakshmi P 2011 An Overview of the scheduling policies and algorithms in Grid Computing Int. J. Res. Rev. Comput. Sci 2 280–94
[11] Dong F, Luo J, Gao L and 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 58–61
[12] Khanli L M and Analoui M 2008 Resource scheduling in desktop grid by grid-JQA The 3rd International Conference on Grid and Pervasive Computing, IEEE 25–8
[13] Khanli L M and 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 34
[14] Elmroth E and Tordsson J 2008 Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions J. Future Gener. Comput. Syst. 24 585–93
[15] Benjamin Kho B T, Veeravalli B, Hung T and Simon See C W 2007 A multi-dimensional scheduling scheme a Grid computing environment J. Parallel Distrib. Comput 67 659–73
[16] Livny M and Melman M 2011 Load balancing in homogenous broadcast distributed systems Proceedings of the ACM Computer Network: Performance Symposium 47–55
[17] Roy S, Banerjee S, Chowdhury K R and Biswas U 2016 Development and analysis of a three phase cloudlet allocation algorithm J. King Saud Univ. Comput. Inf. Sci
[18] Parsa S and Entezari-Maleki R 2009 RASA: a new grid cloudlet scheduling algorithm World Appl. Sci. J. 7 152–60
[19] Kumar P and Dutta Pramanik P K 2012 Host selection methodology in cloud computing environment Int. J. Adv. Res. Comput. Eng. Technol. (IJARCT) 1 2278-1323
[20] El-kenawy E T, El-Desoky A I and Al-rahamawy M F 2012 Extended Max–min scheduling using petri net and load balancing Int. J. Soft Comput. Eng. (IJSCE) 2 2231-307
[21] Bhatia W, Buyya R and Ranjan R 2010 CloudAnalyst: a CloudSim based visual modeler for analysing cloud computing environments and applications, *24th IEEE International Conference on Advanced Information Networking and Applications* 446–52

[22] Banerjee S, Adhikari M, Kar S and Biswas U 2015 Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud *Arabian J. Sci. Eng* **40** 1409–25

[23] Chatterjee T, Ojha V K, Adhikari M, Banerjee S, Biswas U and 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 281–90

[24] Bittencourt L F, Sakellariou R and Madeira E R M 2010 DAG scheduling using a look ahead variant of the heterogeneous earliest finish time algorithm *In: 18th Euromicro International Conference on Parallel, and Network-Based Processing (PDP)* **34** 27