Indonesian Journal of Electrical Engineering and Computer Science
Vol. 22, No. 3, June 2021, pp. 1697~1707
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v22.i3.pp1697-1707

Enhancement of cloud performance metrics using dynamic degree memory balanced allocation algorithm

Aparna Joshi[1], Shayamala Devi Munisamy[2]
[1, 2]Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India
[1]Department of Information Technology, Army Institute of Technology, Pune, India

Article Info

Article history:

Received Feb 5, 2021
Revised May 17, 2021
Accepted May 19, 2021

Keywords:

Cloud computing
CloudSim
Degree of imbalance
Load balancing
Task scheduling

ABSTRACT

In cloud computing, load balancing among the resources is required to schedule a task, which is a key challenge. This paper proposes a dynamic degree memory balanced allocation (D2MBA) algorithm which allocate virtual machine (VM) to a best suitable host, based on availability of random-access memory (RAM) and microprocessor without interlocked pipelined stages (MIPS) of host and allocate task to a best suitable VM by considering balanced condition of VM. The proposed D2MBA algorithm has been simulated using a simulation tool CloudSim by varying number of tasks and keeping number of VMs constant and vice versa. The D2MBA algorithm is compared with the other load balancing algorithms viz. Round Robin (RR) and dynamic degree balance with central processing unit (CPU) based (D2B_CPU based) with respect to performance parameters such as execution cost, degree of imbalance and makespan time. It is found that the D2MBA algorithm has a large reduction in the performance parameters such as execution cost, degree of imbalance and makespan time as compared with RR and D2B CPU based algorithms

This is an open access article under the CC BY-SA license.

Corresponding Author:

Aparna Joshi
Department of Computer Science and Engineering
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology Chennai
Chennai, Tamil Nadu 600062, India
Email: aparna.joshi82@gmail.com

1. INTRODUCTION

Cloud computing allows user to store data remotely and access it from anywhere, using an internet connection [1]. In cloud computing, demand of resources is directly proportional to the number of users. Therefore, in cloud computing, load balancing among the resources is required to schedule a task, which is a key challenge [2], [3]. Load balancing improves system performance, provide backup plan in case of system failure and maintain its stability [4], [5]. Load balancing is carried out by two methods viz. Virtual machine (VM) scheduling and task scheduling. In VM scheduling method, VMs are created on a best suitable host within datacenter. In task scheduling method, tasks were allocated to a best suitable resource for execution. In load balancing, task scheduling is a non-polynomial (NP) hard problem because number of tasks and length of tasks vary rapidly, therefore it is difficult to calculate possible mapping of tasks to resources and evaluate an optimal mapping [6], [7].

To solve the NP hard problems in load balancing, researchers developed both static and dynamic category of algorithms [8]. Statics algorithms requires advanced information about tasks and resources. Also, static algorithm works better in an environment where there is a low variation of nodes in cloud. However, static algorithms are not suitable for cloud environments where load varies rapidly [9], [10]. In that case,

researchers used dynamic category of load balancing algorithms as they utilize information of tasks and resources during run time.

Researchers worked on methods to improve the performance of algorithms used for load balancing. The performance parameters considered to evaluate system performance in load balancing are execution time, makespan, cost, resource utilization, throughput, migration time, and degree of imbalance. Li *et al.* [11] introduced greedy based algorithm by classifying tasks based on quality of service (QoS) parameters. A reduction in the completion time of submitted task was observed by selecting an appropriate branch function [11]. Sahoo *et al*. [12] reduced makespan by using consistent expected time to compute (ETC) matrix on a heterogeneous distributed computing system (HDCS) [12]. Lakra and Yadav [13] applied multi-objective task scheduling algorithm to map task to VM and observed reduction in throughput time and execution cost [13]. Ren *et al*. [14] quantified load and processing power of VMs in a dynamic load balancing algorithm. In this algorithm, by using single exponential mechanism, a reduction in the server load and an improvement in the quality of client service is observed [14]. Tawfeek *et al.* used ant colony optimization algorithm to allocate the incoming jobs to virtual machine and observed a reduction in makespan of given tasks [15]. Babu *et al*. [16] used behaviour of honey bee foraging strategy to balance underloaded and overloaded virtual machines, in cloud computing environments [16]. Sheeja and Jayalekshmi [17] used cost as a parameter to select optimal virtual machine based on honey bee behaviour and obtained a cost-effective method of load balancing. However, in this technique, quality and overall performance of system decreased due to a greater number of VM migrations [17]. Babu and Samuel [18] applied an enhanced bee colony algorithm in which a job priority was considered to migrate tasks from an overloaded VM to an underloaded VM in order to reduce system imbalance. However, in this algorithm, a high rate of migration of task adversely affected the performance of the system [18]. A Joshi *et al*. assigned VMs to host based on number of processors in use and assigned tasks to the resources based on balance condition of VMs. This reduced degree of imbalance of system and also waiting time of tasks [19]. Joshi and Munisamy [20] assigned VMs to host based on membership value of host. This algorithm improves degree of imbalance, execution cost, throughput time, execution time, makespan and central processing unit (CPU) time. In this algorithm, VM allocation and task allocation policy are modified in order to find optimal mapping of resources. To modify VM allocation policy, membership value of host is calculated. Also, to modify task allocation policy, underutilization and overutilization of VMs were calculated [20]. Krishnadoss and Jacob [21] develop oppositional cuckoo search algorithm (OCSA) to improves execution cost and makespan parameter. This algorithm is combination of cuckoo search algorithm (CSA) and oppositional based learning (OBL). This hybrid version provide solution to task scheduling for the dynamic allocation of resources [21]. Krishnadoss and Jacob [22] develop oppositional lion optimization algorithm (OLOA) to improves execution cost and makespan parameter. This algorithm is combination of lion optimization algorithm (LOA) and oppositional based learning (OBL). This hybrid version of algorithm provide solution for task scheduling optimization [22].

The above studies show that an adequate research was carried out to evaluate the performance of scheduling algorithms using the parameters like execution time, makespan, resource utilization etc. However, research on evaluation of algorithms considering degree of imbalance, execution cost and makespan time has not been adequately addressed. Therefore, this work considered method to reduce the performance parameters such as degree of imbalance, execution cost and makespan time. This paper proposes an algorithm which allocate VM to a best suitable host, based on availability of random access memory (RAM) and microprocessor without interlocked pipelined stages (MIPS) of host. In addition, proposed algorithm allocate task to a best suitable VM by considering balanced condition of VM. If VM is in an overloaded condition, task will be transferred to an underloaded VM. Thus, a newly proposed algorithm is given a name as a dynamic degree memory balanced allocation (D2MBA) algorithm. The proposed algorithm D2MBA has been simulated using a simulation tool CloudSim [23]. The proposed algorithm viz. D2MBA based is compared with the other load balancing algorithms viz. Round Robin (RR) and dynamic degree balance with CPU based (D2B_CPU based) [19]. The D2MBA algorithm shows an improved efficiency of system in terms of performance parameters such as degree of imbalance, execution cost and makespan time.

This paper is organized in the following way. Section 2 describe proposed algorithm which include functioning of algorithm. Section 3 explains experimental setup used for the algorithm. Section 4 explain mathematical model of system. Section 5 explains complexity and workflow analysis of proposed algorithm. Section 6 analyses the results and which is followed by conclusions.

2. PROPOSED METHOD

This section explains methodology used to develop proposed algorithm viz D2MBA algorithm. Figure 1 shows general architecture of load balancing. D2MBA based algorithm does an improvement over the D2B_CPU based algorithm by allocating VM to host whose RAM & MIPS is maximum. This modification in

allocation policy minimizes performance parameters like degree of imbalance, execution cost and makespan time. The proposed algorithm works in two phases, in the first phase VMs are allocated and in the second phase tasks are allocated. A detail working of the two phases of algorithm are explained in section 2.1. and section 2.2.

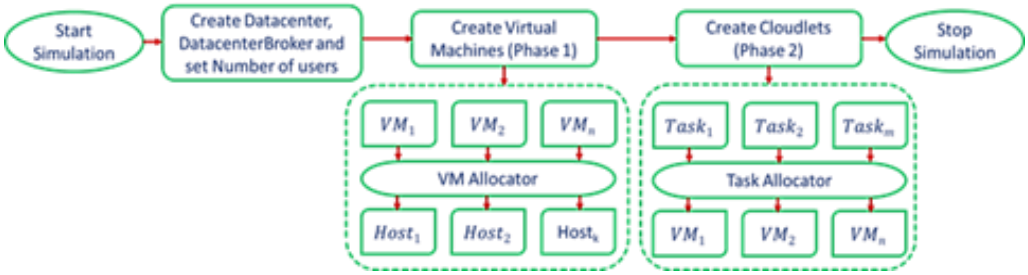

Figure 1. Overall proposed architecture

2.1. Phase 1: VM allocation

In this phase, VM is allocated to a host whose capacity is greater than the requirement of VM. Figure 2 shows the flow of allocation of VMs to hosts based on the available RAM & MIPS value of hosts. Steps involved in allocation of VMs to hosts are described as below. In the first step, list of VMs is given as an input to the algorithm. In the next step, algorithm initializes data structure to store data such as VMTable and MapTable. VMTable stores information about VM and its allocated host and MapTable store information about host and its available RAM & MIPS value. In the next step, algorithm finds required RAM, MIPS and bandwidth (BW) of VM to be allocated. Next, algorithm finds a suitable host for VM from MapTable by considering required values of RAM, MIPS and BW of VMs. An expression used to find out suitable host for VM to be allocated is as given shown in (1).

$$VM_j = \sum_{i=1}^{n} \left(Host_i \geq Requested_{RAM_j} \&\& Host_i \geq Requested_{MIPS_j} \&\& Host_i \geq Requested_{BW_j} \right) \quad (1)$$

where, VM_j is virtual machine to be allocated on $Host_i$, n is number of hosts.

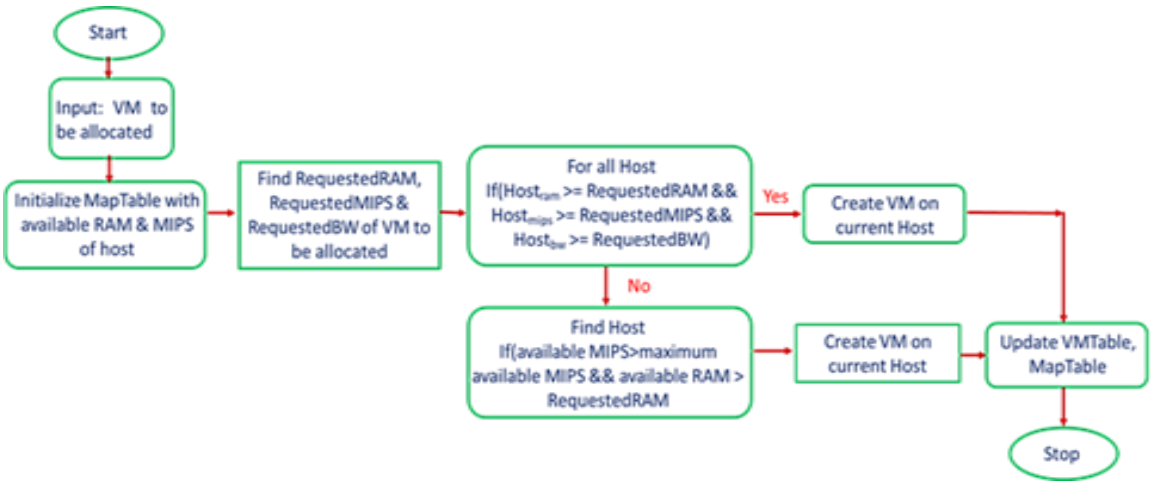

Figure 2. Flowchart showing VM allocation

In this step, algorithm finds host with maximum available values of $RAM, MIPS\ and\ BW$ of host from $MapTable$ and creates VM on a current host. Once VM is created on host, $VMTable$ and $MapTable$ is updated. If VM is not created successfully, then algorithm find a suitable host for VM by using expression given by (2) and create VM on current host. Once VM is created on current host, $VMTable$ and $MapTable$ is updated.

$$VM_j = \sum_{i=1}^{n} \left(Host_{Available_{MIPS}} > Max_{Available_{MIPS}} \;\&\&\; Host_{Available_{RAM}} > Requested_{RAM_j} \right) \quad (2)$$

Where, VM_j is virtual machine to be allocated on $Host_i$, n is number of hosts. Thus, the process of allocation of VM on host is completed. This phase of VM allocation is followed by task allocation phase.

2.2. Phase 2: Task allocation phase

The main aim of task allocation phase is to distribute the dynamic workload to all VMs in order to avoid underutilization or overutilization of resources. In this phase, condition of VM viz. underloaded or overloaded is considered to schedule the task. Here, task is scheduled on a suitable VM in order to reduce the performance parameters like degree of imbalance, execution cost and makespan time. Algorithm 1 describes the flow of task allocation is as given below.

Thus, unlike in the previous algorithms where, researchers allocated VMs on host based on number of processers in use, in the D2MBA algorithm, VM is allocated on host based on available RAM and MIPS available on host. Also, in D2MBA algorithm, workload is distributed dynamically by evaluating load on VM in order to reduce the performance parameters such as degree of imbalance, execution cost and makespan time. Therefore, the proposed algorithm is termed as a dynamic degree memory balanced allocation (D2MBA) algorithm. In the next section, simulation procedure used for the proposed algorithm D2MBA is explained in details.

Algorithm 1. Task allocation phase

Algorithm: Task Allocation
Input: Task T, Virtual Machines VMs
Output: Balances Task Allocations to Virtual Machines

1. Generate number of tasks and VMs
2. Find capacity and loads of all VMs
   Capacity of $V_m(C_i) = PE_{ni} * PE_{mpi} + VM_{bwi}$
   Where, PE_{ni} = No. of processor
   PE_{mpi} =Millions of instructions per second of all processor
   VM_{bwi} =Bandwidth of VM
   Capacity of all $VMs(C)$=$C_1 + C_2 + \cdots + C_n$
   Load on VM $(LVM_{i,t}) = \frac{N(T,t)}{S(VM_{i,t})}$
   Where, $LVM_{i,t}$ =Load of VM_i at time t,
   $N(T,t)$=No. of task at time t on service queue
   $S(VM_{i,t})$=Service rate of VM_i at time t
3. Determine if the system is balanced or not by checking the value of standard deviation "σ" with threshold value Ts (0-1)
4. Load is balanced only if load is smaller than maximum capacity
5. Find out sets of VMs viz. overloaded or underloaded, depending upon load on VMs
6. Sort overloaded VMs in a decreasing order and under loaded VMs in an increasing order
7. Find VMs to transfer task from an overloaded VMs.
   if ($CPU\ Value < Ts$)
   then $transfer\ task\ to\ underloaded\ VMs$
   if ($CPU\ Value == Ts$)
   then $transfer\ task\ to\ balanced\ VMs$
8. Find VMs to transfer task from an underloaded VMs.
   if ($CPU\ Value > Ts$)
   then $transfer\ task\ to\ overloaded\ VMs$
   if ($CPU\ Value == Ts$)
   then $transfer\ task\ to\ balanced\ VMs$
9. Update overloaded, underloaded and balanced set of VMs
10. Return result

3. EXPERIMENTAL SETUP

Simulation tool used for the experiment is a CloudSim [24] simulator. CloudSim simulator supports creation and allocation of VMs at two levels i.e at host level and at VM level. In this paper, CloudSim was used to model datacenter, hosts and VMs in order to experiment in the simulated cloud environment. Figure 3 shows the part of CloudSim simulator used in the simulation along with relationship of its components.

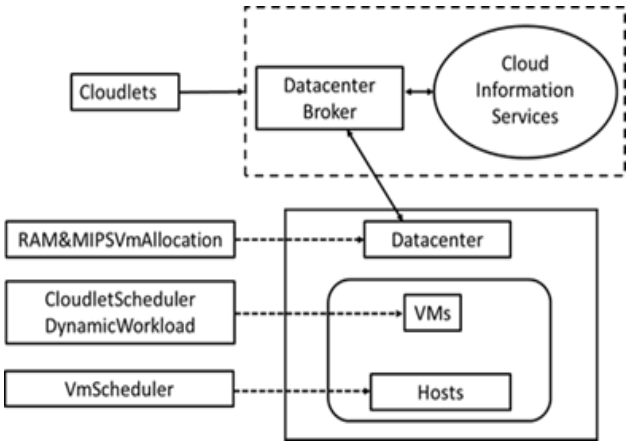

Figure 3. CloudSim and its components

The components of CloudSim used in this simulation and its functions are explained as:

- Cloud Information Service: This component registers datacenter entity and discovers the resources
- Datacenter: It models the core infrastructure level services (hardware), which is offered by cloud provider
- DatacenterBroker: It models the broker which is responsible for mediating negotiations between cloud provider and cloud user
- Host: It models a physical server
- VM: It models a virtual machine which is run on cloud host to deal with the cloudlets
- Cloudlet: It models the cloud-based application service
- VMScheduler: This is an abstract class implemented by host component that model the policies required to allocate processor core to VMs. It runs on every host in datacenter
- CloudletSchedulerDynamicWorkload: This class implements a policy of scheduling performed by VMs. This class inherits CloudSim "CloudletSchedulerTimeShared" class which allocate tasks to VMs for a fixed period of time [1].
- RAM&MIPSVmAllocation: This class allocates VMs to the hosts. It inherits CloudSim "VmAllocationPolicy" which is an abstract class. This class holds internal method of CloudSim which takes input as VMs to be allocated and choose ideal hosts based on RAM and MIPS values. The VM allocation of proposed algorithm is added in this class. This class holds two data structure i.e. VmTable which maps every VM to its allocated host and MapTable which store information of available RAM and MIPS of each host. Table 1 gives parameters and specifications of Virtual machine, datacenter and tasks used in the simulation in details.

Table 1. A set of parameters considered for analysis

Simulation Parameter	Value
Virtual Machines	
Total number of VMs	Varying
Processing speed (MIPS)	Random
Number of PE per VM	1-5 nos
RAM (MB)	Random
Bandwidth (Mbps)	Random
VM Manager	Xen
Operating system	Linux
Cloudlets	
Total number of tasks	60-80 nos
Length of task (MI)	Random
File size (MB)	300
Output size (MB)	300
Datacenter	
No. of datacenter	1
No. of hosts	2

4. MATHEMATICAL MODEL

Dynamic degree memory balanced allocation (D2MBA) algorithm for load balancing which allocate VM to a best suitable host, based on availability of RAM & MIPS of host. In addition, D2MBA algorithm

allocate task to a best suitable VM by considering a balanced condition of VM. Datacenter, host, virtual machine and tasks are the elements of D2MBA algorithm.

The tasks have considered are non-preemptive tasks i.e task cannot be interrupted. Let non-preemptive tasks T_{NPT}= $\{T_1, T_2 \ldots T_m\}$ be the set of m task which should be process on n virtual machine represented by VMs=$\{VM_1, VM_2, \ldots VM_n\}$. These VMs are assigned on suitable host based on availability of RAM & MIPS of hosts. Our aim is to improve performance of system by considering evaluation parameter such as degree of imbalance(DI), execution cost (EC) and makespan time. This evaluation parameter can be represented by P in model. So, proposed model can be represented as $VMs|T_{NPT}|P$.

5. COMPLEXITY AND WORKFLOW ANALYSIS OF PROPOSED ALGORITHM

This section explains complexity and workflow analysis of proposed D2MBA algorithm. The complexity of the scheduling algorithm may have some effect on the system. The algorithm's time complexity is related to the number of n virtual machines and the number of m tasks [25]. While D2MBA algorithm does not utilize priority method, its time complexity remains to $O(mn)$. For space complexity, task scheduling and VMs scheduling is both $O(1)$. So, the total space complexity is $O(1)$. The scheduling method in this paper is simple and does not involve differential or integral calculations. Therefore, the time complexity and space complexity are relatively low. In this paper, analysis part of algorithm is divided into two cases by including three performance parameters in each case, see Figure 4.

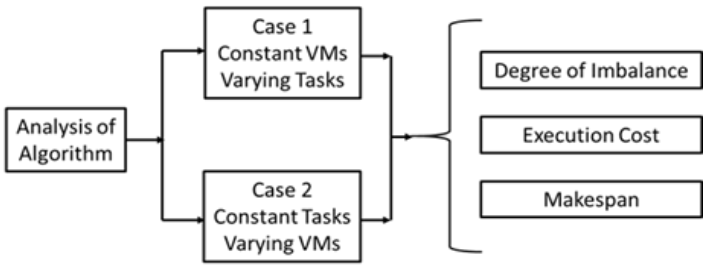

Figure 4. Workflow analysis of algorithm

In case 1, VM is kept constant (No. of VMs=50) and number of tasks varied from 100 to 1000. Whereas, in case 2, task is kept constant (No. of tasks=500) and number of VMs varied from 60 to 80. Three performance parameters viz degree of imbalance, execution cost and makespan time were used to evaluate the performance of algorithms (see Figure 4). Degree of imbalance is calculated using an expression given in (3) and execution cost is calculated using an expression given in (4). Whereas, makespan time is calculated using (5).

$$\text{Degree of Imbalance} = \frac{PT_{max} - PT_{min}}{PT_{Avg}} \quad (3)$$

where PT_{max} , PT_{min} and PT_{Avg} are maximum processing time, minimum processing time and average processing time among heterogeneous VMs respectively.

$$\text{Execution Cost} = \frac{\sum_{i=1}^{m} Task\ Length_i + \sum_{j=1}^{n} Cost\ per\ second_j}{\sum_{j=1}^{n} VM_{mips_j}} + processing\ cost \quad (4)$$

where m is number of tasks and n is number of VMs.

$$\text{Makespan} = \max_{1 \le i < m} \{CT_i\} \quad (5)$$

where, CT_i is completion time of task i. The two cases as defined above are explained in next section as follows.

6. RESULTS AND DISCUSSION

In this section, results of a newly proposed $D2MBA$ algorithm are presented. These results in terms of performance parameters are compared with the algorithms viz. Round Robin (RR) and dynamic degree balance CPU based (D2B_CPU). The parameters used to evaluate the performance of D2MBA algorithm with

the other two algorithms are degree of imbalance, execution cost and makespan time. The two cases as defined above (refer Figure 4) are explained in details as follows:

6.1. Case 1

Case 1: In this case, variation in the three performance parameters viz. execution cost, degree of imbalance and makespan time for all the algorithms viz. RR, D2B_CPU based and D2MBA algorithm (proposed algorithm) are compared and presented. Here, number of VMs were kept constant to 50 and number of tasks varied from 100, 300, 500, 700 and 1000. In this case, first result on execution cost is presented which is followed by results on degree imbalance and finally results on makespan time are presented.

a. Variations in execution cost

Variations in execution cost with an increase in the number of tasks were shown in the Table 2 and presented in Figure 5. Table 2 gives a value of the execution cost for all the three algorithms. It is observed that the D2MBA algorithm has lowest values for execution cost. From the Table 2, it is observed that, the proposed D2MBA algorithm reduces execution cost by an average 0.83% as compared to RR and 0.87% as compared to D2B_CPU based algorithm. From Figure 5, it is observed that, in case of RR and D2B_CPU based algorithm, execution cost remains constant with an increase in the number of tasks from 100 to 1000. Also, both the algorithms have more or less same execution costs for the tasks in the range from 100 to 1000 nos. In the case of D2MBA algorithm, with an increase in the number of tasks from 100 to 500, execution cost decreases by 0.87 $. However, with an increase in the number of tasks 500 to 1000, execution cost increases by 1.49 $. It is also observed that, all the three algorithms have same execution cost of 61.03 $ for the tasks in the range from 700 to 1000 nos. Thus, it is observed that the D2MBA algorithm has lowest value of execution cost for smaller number of tasks.

b. Variations in the degree of imbalance

In this section, results on variations in the degree of imbalance with an increase in the number of tasks were shown in Table 3 and presented in the Figure 6. Table 3 gives a value of the degree of imbalance for all the three algorithms. It is observed that the D2MBA algorithm has lowest values for degree of imbalance. From the Table 3, it is observed that, D2MBA algorithm reduces degree of imbalance by an average 91.68% as compared to RR and 43.35% as compared to D2B_CPU based algorithm. From Figure 6, it is observed that, in case of D2B_CPU based and D2MBA algorithms (proposed algorithm), degree of imbalance remains more or less constant with an increase in the number of tasks from 100 to 1000 nos. In case of RR algorithm, degree of imbalance is far higher than the other two algorithms viz.D2B_CPU based and D2MBA algorithm. Also, it is observed that at a lower number of tasks ranging from 100 to 300 nos., degree of imbalance decreases by 4.7. However, with further increase in the number of tasks from 300 to 1000, degree of imbalance remains more or less constant (avg. 2.4). Thus, it is observed that, D2MBA algorithm has lower degree of balance as compared with the other two algorithms.

Table 2. Execution cost on varying number of tasks

	Execution cost ($)		
No of Tasks	D2B_CPU	RR	D2MBA
100	60.92	61.05	60.4
300	61.08	60.86	60.04
500	60.83	61.11	59.53
700	60.88	61.17	60.71
1000	61.05	61.07	61.02

Table 3. Degree of imbalance on varying number of tasks

	Degree of imbalance		
No of Task	D2B_CPU	RR	D2MBA
100	1.36	7.46	0.89
300	0.53	2.72	0.29
500	0.33	2.63	0.22
700	0.30	2.51	0.14
1000	0.19	1.97	0.10

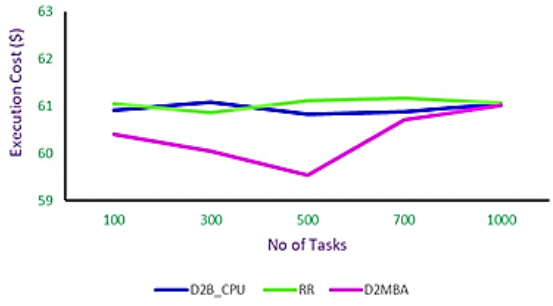

Figure 5. Variation in the execution cost ($) vs. number of tasks

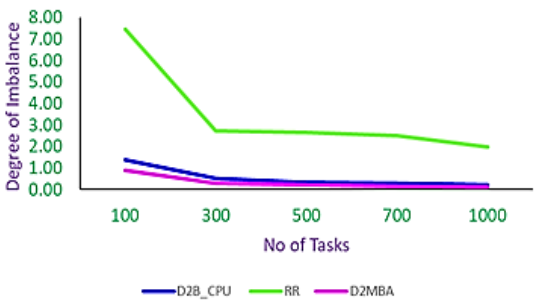

Figure 6. Variation in the degree of imbalance vs. number of tasks

c. Variations in the makespan time

In this section, results on variations in the makespan time with the changes in the number of tasks are presented in Table 4 and plotted in the Figure 7. Table 4 gives a value of the makespan time for all the three algorithms. It is observed that the D2MBA algorithm has lowest values for makespan time. From the Table 4, it is observed that, D2MBA algorithm reduces makespan time by an average 8.32% as compared to RR and 8.93% as compared to D2B_CPU based algorithm. From Figure 7, it is observed that, at 100 number of tasks, the makespan time for D2MBA algorithm is 301 ms, for RR algorithm is 430 ms and for D2B_CPU based algorithm is 439 ms. At a number of tasks in the range from 300 to 1000 nos., variation in the makespan time for all the three algorithms remains approximately constant. Thus, for a small number of tasks (100 nos.) the D2MBA algorithm has a reduction in makespan time by 30.01% as compared to Round Robin (RR) and by 31.53% as compared with dynamic degree balance with CPU based (D2B_CPU). However, for variation in the tasks in the range from 300 to 1000, the D2MBA algorithm has a reduction in makespan time by 2.9% as compared to Round Robin (RR) and by 3.28% as compared with dynamic degree balance with CPU based (D2B_CPU). Thus, for the number of tasks ranging from 100 to 1000, the D2MBA algorithm has lower makespan time as compared with the other two algorithms.

Table 4. Makespan time on varying number of tasks

	Makespan Time (ms)		
No of Tasks	D2B_CPU	D2MBA	RR
100	439.93	301.22	430.60
300	497.07	472.94	492.52
500	502.96	488.08	506.22
700	514.79	496.03	510.44
1000	516.20	507.52	513.78

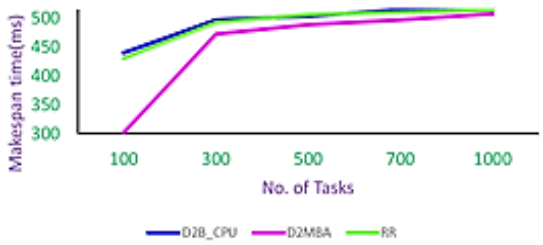

Figure 7. Variation in the makespan vs. number of tasks

6.2. Case 2

Case 2: In this case, results on variation in the performance parameters such as execution cost, degree of imbalance and makespan time for all the algorithms viz. RR, D2B_CPU based and D2MBA algorithm (proposed algorithm) are compared and presented. Here, numbers of tasks were kept constant to 500 and number of VMs varied from 60 to 80. In this case, first results on execution cost are presented which is followed by results on degree of imbalance and finally results on makespan time are presented.

a. Variations in execution cost

In this case, results on variations in the execution cost with an increase in the number of VMs is presented in Table 5 and plotted in Figure 8. Table 5 gives a value of the execution cost for all the three algorithms. It is observed that the D2MBA algorithm has lowest values for execution cost. From the Table 5, it is observed that, the proposed D2MBA algorithm reduces execution cost by an average 23.30% as compared to RR and 23.47% as compared to D2B_CPU based algorithm. From the Figure 8, it is observed that, in case of RR and D2B_CPU based algorithm, execution cost remains constant with an increase in the number of VMs from 60 to 80. Also, both the algorithms have similar execution costs. However, in the case of D2MBA algorithm, with an increase in the number of VMs from 60 to 70, execution cost decreases by 11.62$. Also, with an increase in the number of VMs from 70 to 80, execution cost further decreases by 4.83$. It is also observed that, at a small number of VMs i.e 60, execution cost of D2MBA algorithm is smaller by 5$. However, at a higher number of VMs i.e 80, execution cost of proposed algorithm is smaller by 21$. Thus, it is observed that the D2MBA algorithm has lowest value of execution cost for VMs ranging from 60 to 80.

b. Variations in the degree of imbalance

In this case, results on variations in the degree of imbalance with an increase in the number of VMs are shown in Table 6 and plotted in the Figure 9. Table 6 gives a value of the degree of imbalance for all the three algorithms. It is observed that the D2MBA algorithm has lowest values for degree of imbalance. From the Table 6, it is observed that, the proposed D2MBA algorithm reduces degree of imbalance by an average 87.50% as compared to RR algorithm and 30.07% as compared to D2B_CPU based algorithm. From Figure 9, it is observed that, in case of RR algorithm, degree of imbalance is far higher than the other two algorithms viz.D2B_CPU based and D2MBA algorithm. Also, with an increase in number of VMs, degree of imbalance of RR algorithm remains more or less constant at a higher value of 3.3. In case of D2MBA algorithm, at 60 VMs, degree of imbalance is the lowest one. In case of D2B_CPU based algorithm, at 60 VMs, degree of imbalance is higher than the D2MBA algorithm by 0.9. Also, it is observed that with an increase in the number of VMs from 60 to 70, degree of imbalance of D2B_CPU based algorithm decreases by 0.82. Further, with an increase in the number of VMs from 70 to 80, degree of imbalance of D2B_CPU based and D2MBA algorithm remains more or less constant and with similar values. Thus, for the number of VMs ranging from 60 to 80, D2MBA algorithm has lower degree of balance as compared with the other two algorithms.

Table 5. Execution cost on varying number of VM's

	Execution Cost ($)		
No of VM's	D2B_CPU	RR	D2MBA
60	60.93	60.73	55.9
70	60.76	60.59	44.28
80	60.71	60.71	39.45

Table 6. Degree of imbalance on varying number of VM's

	Degree of Imbalance		
No of VMs	D2B_CPU	RR	D2MBA
60	1.36	3.23	0.44
70	0.54	3.26	0.50
80	0.33	3.31	0.28

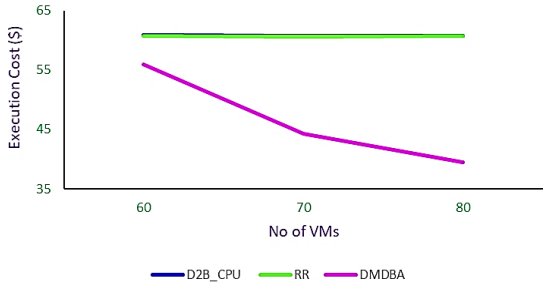

Figure 8. Variation in the execution cost ($) vs number of VMs

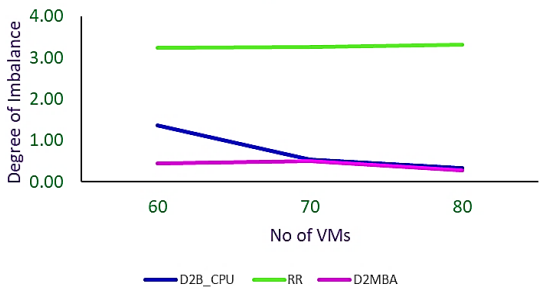

Figure 9. Variation in the Degree of imbalance vs. number of VMs

c. Variations in makespan time

In this case, results on variations in the makespan time with an increase in the number of VMs are shown in Table 7 and plotted in the Figure 10. Table 7 gives a value of the makespan time for all the three algorithms. It is observed that the D2MBA algorithm has lowest values for makespan time. From the Table 7, it is observed that, the proposed D2MBA algorithm reduces makespan time by an average 40.80% as compared to RR and 26.20% as compared to D2B_CPU based algorithm. From Figure 10, it is observed that, for VMs in the range from 60 to 80 nos., the D2MBA algorithm has lowest makespan time, RR algorithm has the highest makespan time, whereas the D2B_CPU based algorithm has makespan time in between the other two. Also, in case of RR algorithm, it is observed that, with an increase in the number VMs from 60 to 70, makespan time decreases by 83.34 ms. However, with a further increase in the number of VMs from 70 to 80, the makespan time remains more or less constant. In case of both D2B_CPU based and D2MBA algorithm, with an increase in the number of VMs from 60 to 80, makespan time is observed to decrease. Thus, for the number of VMs ranging from 60 to 80, the D2MBA algorithm has lower makespan time as compared with the other two algorithms.

Table 7. Makespan time on varying number of VM's

	Makespan Time (ms)		
No of VM's	D2B_CPU	D2MBA	RR
60	339.94	301.22	513.79
70	316.20	288.94	430.44
80	302.96	263.08	406.22

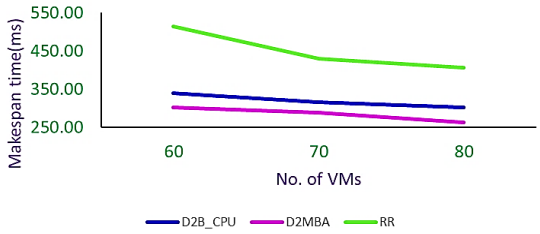

Figure 10. Variation in the makespan Vs number of VMs

7. CONCLUSION

This paper proposed a dynamic degree memory balanced allocation (D2MBA) algorithm for load balancing which allocate VM to a best suitable host, based on availability of RAM & MIPS of host. In addition, D2MBA algorithm allocate task to a best suitable VM by considering a balanced condition of VM. The performance parameters such as degree of imbalance (DI), execution cost (EC) and makespan time of D2MBA algorithm is compared with the other two algorithms viz. RR and D2B_CPU based. The simulations were performed by varying number of tasks and keeping number of VMs constant and vice versa. Following conclusions can be drawn from the analysis of results described in the above section. In both the cases, i.e., keeping VMs constant and varying tasks (Case 1) and keeping task constant and varying VMs (Case 2), it is observed that the D2MBA algorithm has a large reduction in the performance parameters such as execution cost and degree of imbalance as compared with RR and D2B_CPU based algorithms. Similarly, in both the cases, the D2MBA algorithm has reduction in makespan time as compared with RR and D2B_CPU based algorithms. Thus, the proposed Dynamic Degree Memory Balanced Allocation (D2MBA) algorithm has superior performance parameters as compared to RR and D2B_CPU based algorithms.

ACKNOWLEDGEMENTS

Authors thanks to Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology Chennai, India and Army Institute of Technology for providing an infrastructure to carry research on above mentioned topic.

REFERENCES

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, Hype and reality for delivering computing as the 5th utility," *Future Generation of Computer System*, vol. 25, no. 6, pp. 599-616, 2009, doi: https://doi.org/10.1016/j.future.2008.12.001.

[2] S. Afzal and G. kavitha, "Load balancing in cloud computing-A hierarchical taxonomical classification," *Journal of Cloud Computing*, vol. 8, no. 1, pp:1-24, 2019, doi: 10.1186/s13677-019-0146-7.

[3] B. P. Mulla, C. R. Krishna, and R. K. Tickoo, "Load balancing algorithm for efficient VM allocation in heterogeneous cloud," *International Journal of Computer Network and Communication*, vol 12, no 1, pp: 83-96, 2020, doi: 10.5121/ijcnc.2020.12106.

[4] S. K. Mishra, B. sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 2, pp: 149-158, 2020, doi: https://doi.org/10.1016/j.jksuci.2018.01.003.

[5] Y. A/P Parmesivan, S. Hasan, and A. Muhammed, "Performance Evaluation of load balancing algorithm for virtual machine in data centre in cloud computing," *International Journal of Engineering & Technology*, vol. 7, no. 4, pp: 386-390, 2018

[6] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Load balancing algorithms in cloud computing: a survey," *Journal of Network Computer Application*, vol. 80, pp. 50-71, 2017, https://doi.org/10.1016/j.jnca.2017.04.007.

[7] S. R. Gundu, C. A. Panem, and A. Thimmapuram, "Real-Time Cloud-Based load balance algorithms and an analysis," *SN Computer Science*, vol 1, pp: 1-9, 2020.

[8] A. Joshi, M. S. Devi, "A Survey of Job Scheduling Algorithms for Load Balancing in Hadoop Environment," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 16, pp. 5033-5046, 2018.

[9] G. Gopinath P P, and S. K. Vasudevan, "An in-depth analysis and study of load balancing techniques in the cloud computing environment," *Procedia Computer Science*, vol. 50, pp: 427-432, 2015, doi: https://doi.org/10.1016/j.procs.2015.04.009.

[10] M. A. Shahid, N. Islam, M. M. Alam, M. M. Su'ud and S. Musa, "A Comprehensive Study of Load Balancing Approaches in the Cloud Computing Environment and a Novel Fault Tolerance Approach," *IEEE Access*, vol. 8, pp. 130500-130526, 2020, doi: 10.1109/ACCESS.2020.3009184.

[11] J. Li, L.Feng, and S. Fang, "An greedy-based job scheduling algorithm in cloud computing," *Journal of Software*, vol. 9, no. 4, 2014, doi: 10.4304/jsw.9.4.921-925.

[12] B. Sahoo, D. Kumar, and S. K. Jena, "Analysing the Impact of Heterogeneity with Greedy Resource Allocation Algorithms for Dynamic Load Balancing in Heterogeneous Distributed Computing System," *International Journal of Computing Application*, vol. 62, no. 19, pp. 25-34, 2013, doi: 10.5120/10190-5070.
[13] Lakra and D. Yadav, "Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization," *Procedia Computer Science*, vol. 48, pp: 107-113, 2014, doi: https://doi.org/10.1016/j.procs.2015.04.158.
[14] X. Ren, R. Lin and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast," *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, 2011, pp. 220-224, doi: 10.1109/CCIS.2011.6045063.
[15] M. A. Tawfeek, A. El-Sisi, A. E. Keshk and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," *2013 8th International Conference on Computer Engineering & Systems (ICCES)*, 2013, pp. 64-69, doi: 10.1109/ICCES.2013.6707172.
[16] K. R. Remesh Babu, A. A. Joy and P. Samuel, "Load balancing of tasks in cloud computing environment based on bee colony algorithm," *2015 Fifth International Conference on Advances in Computing and Communications (ICACC)*, 2015, pp. 89-93, doi: 10.1109/ICACC.2015.47.
[17] Y. S. Sheeja and S. Jayalekshmi, "Cost effective load balancing based on honey bee behaviour in cloud environment," *2014 First International Conference on Computational Systems and Communications (ICCSC)*, 2014, pp. 214-219, doi: 10.1109/COMPSC.2014.7032650.
[18] K. R. Babu, and P. Samuel, "Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud," *Innovation in Bio-Inspired Computing and Applications, Advances in Intelligent Systems and Computing, Springer*, vol. 424, pp. 67-78, Dec 2015, doi: 10.1007/978-3-319-28031-8_6.
[19] A. Joshi and M. Shyamala Devi, "Dynamic Degree Balanced with CPU Based VM Allocation Policy for Load balancing," *Journal Of Information & Optimization Sciences*, vol. 41, no. 2, pp. 543-553, 2020, doi: https://doi.org/10.1080/02522667.2020.1724618.
[20] A.Joshi, and S. D. Munisamy, "Enhancement of Performance Parameter of Cloud using Dynamic Degree Balanced with Membership Value Algorithm," *International Journal of Advanced Research in Engineering and Technology*, vol. 11, no.8, pp. 664-676, August 2020, doi: 10.34218/IJARET.11.8.2020.065.
[21] P. Krishnadoss, and P. Jacob, "OCSA:Task Scheduling Algorithm in Cloud Computing Environment," *International Journal of Intelligent Engineering & Systems*, vol. 11, pp. 271-279, 2018, doi: 10.22266/ijies2018.0630.29.
[22] P. Krishnadoss, P. Jacob, "OLOA:Based Task Scheduling in heterogeneous Clouds," *International Journal of Intelligent Engineering & Systems*, vol. 12, no. 1, pp. 114-122, 2019, doi: 10.22266/ijies2019.0228.12.
[23] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and Experience*, vol. 41, no. 1, pp. 23-50, Jan. 2011, doi: 10.1002/spe.995.
[24] V. Velde and B. Rama, "Simulation of optimized load balancing and user job scheduling using CloudSim," *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2017, pp. 1379-1384, doi: 10.1109/RTEICT.2017.8256824.
[25] H. Saleh, H. Nashaat, W. Saber and H. M. Harb, "IPSO Task Scheduling Algorithm for Large Scale Data in Cloud Computing Environment," *IEEE Access*, vol. 7, pp. 5412-5420, 2019, doi: 10.1109/ACCESS.2018.2890067.

BIOGRAPHIES OF AUTHORS

Aparna Shashikant Joshi is a research scholar at Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology Chennai, India. Author working as an Assistant Professor at Department of Information Technology, Army Institute of Technology, Pune. Author's area of interests is cloud computing.

Dr. Shayamala Devi Munisamy is working as a Professor at Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology Chennai, India. Her area of interests is cloud computing, Machine learning and image processing.