Asian Journal of Research in Computer Science

7(1): 1-13, 2021; Article no.AJRCOS.63913
ISSN: 2581-8260

A Threshold-based Tournament Resource Allocation in Cloud Computing Environment

I. P. Oladoja[1*], O. S. Adewale[1], S. A. Oluwadare[1] and E. O. Oyekanmi[2]

[1]Department of Computer Science, the Federal University of Technology Akure, Nigeria.
[2]Department of Mathematical Science, Achievers University Owo, Nigeria.

Authors' contributions

This work was carried out in collaboration among all authors. Author IPO designed the study, performed the statistical analysis, wrote the protocol and wrote the first draft of the manuscript. Author OSA supervised the study, validated the data and edited the manuscript draft. Author SAO co-supervised the study and also edited the manuscript draft. Author EOO was also involved in the statistical analysis, design of study and literature survey. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/AJRCOS/2021/v7i130169
Editor(s):
(1) Dr. G. Sudheer, GVP College of Engineering for Women, India.
Reviewers:
(1) Radael de Souza Parolin, Federal University of Pampa, Brazil.
(2) A. Ponshanmugakumar, Sri Sairam Institute of Technology, India.
Complete Peer review History: http://www.sdiarticle4.com/review-history/63913

Original Research Article

Received 06 October 2020
Accepted 12 December 2020
Published 07 January 2021

ABSTRACT

Cloud computing environments provide an apparition of infinite computing resources to cloud users so that they can increase or decrease resource consumption rate according to their demands. In the Cloud, computing resources need to be allocated and scheduled in a way that providers can achieve high resource utilization and users can meet their applications' performance requirements with minimum expenditure. Due to these different intentions, there is the need to develop a scheduling algorithm to outperform appropriate allocation of tasks on resources. The paper focuses on the resource optimization using a threshold-based tournament selection probability for virtual machines used in the execution of tasks. The proposed approach was designed to create metatask and the proposed algorithm used was Median-Based improved Max-Min algorithm. The experimental results showed that the algorithm had better performance in terms of makespan, utilization of resources and throughput. The load balance of tasks was also fairly distributed on the two datacenters.

**Corresponding author: E-mail: writepep@yahoo.com, perpetual.oladoja@gmail.com;*

Keywords: Cloud computing; genetic algorithm; resource allocation; threshold-based tournament.

1. INTRODUCTION

The concept of cloud computing has been around since the early 1950s, but the term was not coined then. It was addressed as "Time sharing systems". During the period of 1960-1990, different experts talked about the era of cloud computing in various books or quotes. In the early 1990s, the telecommunications companies began to offer Virtual Private Networks (VPNs), instead of dedicated connections, which were standard in Quality of Service (QoS) but were comparatively cheaper. This move, aided the advent of cloud computing which was introduced around the year 2002 by Amazon. This, organization can be considered as one of the pioneers in this field, with their Amazon Web Services (AWS) and Elastic Compute Cloud (EC2) [1].

A Cloud is a distributed system, which consists of a parallel collection of interconnected and virtualized computers. These computers are provisioned and seen as one or more integrated computing resources based on service-level agreements acknowledged through discussions between the service provider and consumers. The computing resources, are virtualized and allocated as services from providers to users and this can be allocated dynamically upon the requirements and preferences of the consumers [2].

In Cloud Computing Environment, Tasks are sent to the data center broker (DCB) by the Users. The broker is responsible for scheduling tasks on Virtual machines (VM) and also stands as an intermediary between the Cloud Users and cloud Providers. A data center is a virtual Infrastructure for encasing resources and it consists of a number of Hosts. The proffered tasks are scheduled according to the scheduling policies used by the DCB. The DCB communicates directly with the Cloud controller and tasks are assigned to Virtual machines in the Host [3].

There are different types of Scheduling according to different policies for example, Immediate and batch scheduling, centralize and distributed scheduling. Task scheduling is a process of selecting the best resource accessible for tasks execution. Task scheduling Algorithms aims at minimizing the completion time of tasks and also maximizes resource utilization, to meet user requirements [4].

In cloud computing, tasks need to be executed by the resources, to achieve high performances, optimal completion time, reduce response time and effective resources utilization. Premised on these aforementioned objectives, there is need to develop and propose a scheduling algorithm that will be used by task scheduler to appropriately allocate tasks to resources.

2. LITERAURE REVIEW

Saraswathi et al. [5] proposed a dynamic resource allocation scheme in cloud computing, where an effective and dynamic utilization of the resources in cloud can be use to balance the load and avoid situations like the systems running slow. The study focused on allocation of VM to the user, based on analyzing the characteristics of the job. The limitation of the work is that the proposed approach left some jobs idle for a long time, due to the time taken to create new virtual machines and it was not implemented in a real time cloud environment.

In Hamed et al. [6], a hybrid genetic algorithm with a knowledge-based operator for solving the job shop scheduling problems was proposed. The study minimized makespan to solve the problem more effectively and an operation-based representation was used to enable the construction of feasible schedules. The study recommended developing new operators that further increase the population diversity of the algorithm and modeling an operator to measure the population diversity.

In Priya and Babub N.K., [7], a resource scheduling algorithm with load balancing for cloud service provisioning was proposed. The objective of the work was to introduce an integrated resource scheduling and load balancing algorithm for efficient cloud service provisioning. The method constructed a fuzzy-based multidimensional resource scheduling and queuing network (F-MRSQN) was used for efficient scheduling of resources and optimizing the load for each cloud user requests, with the efficient evolution of data center. The effectiveness of F-MRSQN method was estimated by attaining simulation results for testing the average success rate and resource scheduling efficiency and response time. The

results showed that F-MRSQN method provided better performance with an improvement of average success ratio by 9% and reduced the response time by 20% compared to existing methods. The work did not investigate privacy-aware efficient resource scheduling with load balancing of intermediate data and information in cloud or by taking privacy preserving as a metrics with other metrics.

Wang et al., [8] worked on improving task scheduling with parallelism awareness in heterogeneous computational environments. The paper focused on the problem of executing tasks with deadline constraints with parallelism awareness where the parallel degree of each task can be tuned between one or more cores of the server assigned during execution. The problem was first modelled as an optimization problem thereby maximizing the overall utilization of servers. A scheduling method with parallelism awareness (SPA) was proposed, where each core iteratively allocates much resources to task with the earliest deadline on a server and thereby reducing the number of decision variables. The paper introduced a method of calculating the start time and the finish time for each task into the optimization problem and transforming the problem into binary programming. The issue was then identified in polynomial time, based on existing task scheduling methods. The study demonstrated a great performance improvement in resource utilization, task violations, finish time, and energy efficiency, when executing tasks in a heterogeneous computational system using SPA. The study was limited by the fact that scheduling designs policies with parallelism awareness, to improve the efficiency and effectiveness of computational systems executing tasks online was not considered.

In Krishnaveni and Prakash [9], an execution time based sufferage algorithm for static task scheduling in cloud was proposed. The research focused developing an efficient algorithm named Execution Time Based Sufferage Algorithm (ETSA) that takes into account, the parameters makespan and also the resource utilization in scheduling the tasks. The scheduling was vital in attaining a high-performance schedule in a heterogeneous-computing system. Existing scheduling algorithms such as Min-Min, Sufferage and Enhanced Min-Min, focused only on reducing the makespan but failed to consider the other parameters like resource utilization and load balance. The work was implemented in Java with Eclipse IDE and a set of Expected Time to Compute (ETC) matrices was used in the proposed algorithm. The ETSA delivers better makespan and resource utilization than the other existing algorithms. Proposed ETSA, compares the Sufferage Value (SV) of each task with EXSV and then take the decision to give out the tasks to the resource. It also tries to decrease the makespan with a balanced load across the resource. It gives better result in terms of makespan and resource utilization with a balanced load when compared with existing Min-Min, Enhanced Min-Min, and Sufferage. The limitation was that the algorithm was not applied in actual cloud computing environment (CloudSim) and other parameters such as computational cost, storage cost, and deadline of the tasks was not considered.

In order to obviate the challenges identified in the previous studies, the present research aimed at designing a threshold-base tournament selection for resource allocation, improve the load balance and resource utilization, thereby maximizing the throughput.

3. SCHEDULING MODEL

The Cloud manager occasionally collects information about resources availability and the price for each resource in the database. It obtains this information from the different cloud providers and acts as a pricing interface between them and the users, thereby updating the database when new information is available. The architecture has two main actors: the broker and the user of the cloud. The former adjusts the configuration options (available clouds, resource types from each cloud, pricing information, etc.) before the execution begins; while the latter receives information from the broker and specifies a new service to deploy among available clouds, describing it through a service description file. The broker is the one who deploys the services among the available cloud providers. One of the main components of the broker architecture in the cloud is the scheduler, which is responsible for independently making scheduling decisions based on dynamic pricing schemes, dynamic user demands, and different resource types performance. The Scheduler is responsible for making placement decision and can also be configured to work with different scheduling policies based on different optimization criteria, such as service cost, service performance, etc [10].

A data center can manage several hosts which in turn manages virtual machines (VMs) during their life cycles. The datacenters must have some characteristics and this characteristic must be basically for the host so that each datacenter may have some host. Host is a CloudSim component that represents a physical computing server in a Cloud: it is assigned a pre-configured processing capability (expressed in millions of instructions per second—MIPS), memory, storage, and a provisioning policy for allocating processing cores to VMs.

The datacenter broker has some characteristics and also have some tasks which is called the cloudlet(s) in CloudSim frame work. There may be one cloudlet or set of cloudlets which will be submitted to the broker and once the broker has the details it then directly interacts with the datacenters and assign this cloudlet(s) to some VM(s) which runs on the host. The Host component implements interfaces that support modeling and simulation of both single-core and multi-core nodes. The host can have some set of VM and these VM must also have the hardware configurations of the host. In the CloudSim, VM allocation policy deals with datacenters, while VM Scheduler policy deals with host. All processing done inside the CloudSim can either be time-shared or space-shared, Maheswaran et al. [11].

In Fig. 1, the relationships between the datacenter, host, broker and virtual machines are shown:

Fig. 1. shows the data flow of the cloud simulator and the activities of the simulator are represented in the following equations:

$$Let\ Q = \{U_T\} \quad (1)$$

where U_T is user submitted tasks and T is from 1,2, 3…n

Let B = Broker
Let D = Datacenters
Let S= Services needed by users

$$Then\ S = B \circ D\ \ i.e. a\ composition\ of\ the\ relations\ B\ and\ D \quad (2)$$

$$Let\ U = Q \bullet S \quad i.e. the\ operations\ of\ elements\ Q\ and\ S \quad (3)$$

This research used a space-shared policy which was done inside the broker and this decides which VM get the tasks. The equations (4, 5 and 6) were used to show the relationship between the host and Virtual machines.

Consider *H* as a set of hosts in the entire system, where

$$H = \{h_1, h_2, h_3, \dots, h_N\} \quad (4)$$

N is the total number of the hosts and an individual host can be denoted as h_i, where i denote the host number and range from 1 to *N*. Similarly, a set of VMs on each Host h_i can be represented as

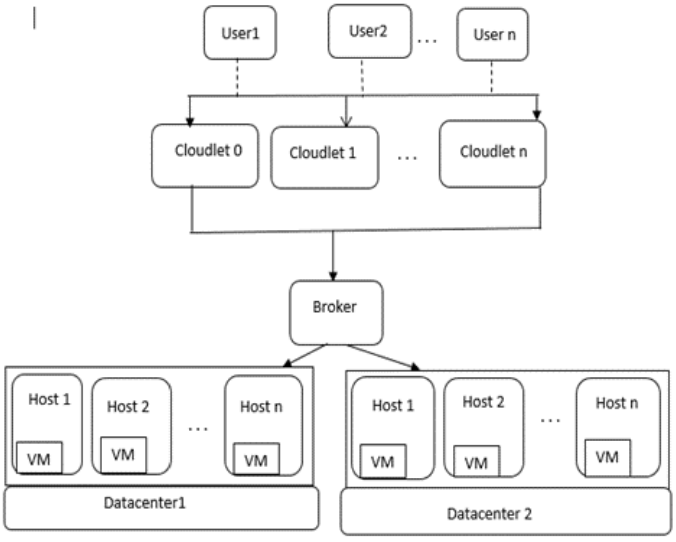


Fig. 1. Cloud Simulator data flow

$$V = \{v_1, v_2, v_3, \dots, v_m\} \quad (5)$$

Here, *m* is the total number of VMs on the physical server i. If VM *V* is deployed on the present system, then a solution set denoted by

$$S = \{s_1, s_2, s_3, \dots \dots, s_N\} \quad (6)$$

and this represents the mapping solution after VM *V* is assigned to each of the Hosts. When the *V* is arranged with the Host *H,* the result is the mapping structure denoted as S.

VM has instructions (I), Size (S), RAM (M), Processor (P) and Bandwidth (B) as its components. Mathematically, this can be represented as follows:

$$\mathrm{VM} = \{\mathrm{I}, \mathrm{S}, \mathrm{M}, \mathrm{P}, \mathrm{B}\} \quad (7)$$

$$\forall\, VM\ \exists\ I |\ I = \{i_1, i_2, \dots, i_n\} \quad (8)$$

$$I_{max} = Max\{i_1, i_2, \dots, i_n\} \quad (9a)$$

$$I_{min} = Min\{i_1, i_2, \dots, i_n\} \quad (9b)$$

Jobs coming into the queue will be in sequence of 10, 30, nth term and this is done by using the arithmetic progression method. In mathematics, an **arithmetic progression** (AP) or **arithmetic sequence** is a sequence of numbers such that the difference between the consecutive terms is constant. If the initial term of an arithmetic progression is a_1 and the common difference of successive members is *d,* then the *n*th term of the sequence a_n is given by:

$$a_n = a_1 + (n - m)d \quad (10)$$

where a_1 is the first term of an arithmetic progression which is (10), a_n is the nth term of an arithmetic progression, *d* is the difference between terms of the arithmetic progression but in this equation *d* is (2*10), *n* is the number of terms in the arithmetic progression and *m* = 1.

4. SCHEDULING ALGORITHM

All tasks in this paper are computational ones, only the Meta task is considered, and the tasks are independent of each other. Each Request (Cloudlet) has size. In order to efficiently implement the Genetic Algorithm for the allocation of VM, this work used a reduction method by reducing the request size by the value of the highest instruction size from the virtual machines (resources) that will be used in the processing.

Therefore, if R_s represent the request's size, the reduced form is termed as

$$R = roundup(\frac{R_S}{I_{max}} * 10) \quad (11)$$

$$Gene\_1 = BIN\ (R) \quad (12)$$

Each virtual machine's Mips is also converted into binary form

$$Gene\_2 = BIN\ (I/I_{min}) \quad (13)$$

Chromosome is formed by combining request in binary form (Gene_1) with binary converted, randomly picked virtual machine.

where BIN is procedure for binary conversion. $Gene\_1$ bits are set as the solution (target) which serves as the fitness criterion for final best-fit chromosome. In other words, once the genetic algorithm gets same fitness value with Gene_1, the iteration is stopped. This research used a datacenter with 4 hosts with 10 cloudlets or requests and adopted the size ranges from 300 MB to 23000 MB [12]. Table 1. shows the configuration of the datacenter which consist of virtual machines with their configurations as follows:

Table 1. Virtual machine configuration

VM(ID:0)	VM(ID:1)	VM (ID: 2)	VM (ID: 3)
Mips= 250	Mips =500	Mips =1000	Mips =5000
Size = 10000MB	Size =10000MB	Size =10000MB	Size =10000MB
Ram =512MB	Ram =512MB	Ram =512MB	Ram =512MB
Bandwidth=1000kbps	Bandwidth=1000kbps	Bandwidth=1000kbps	Bandwidth=1000kbps
Pes number =1	Pes number =1	Pes number =1	Pes number =1

Source: Neha, 2014.

A binary coding with 6-bit strings (S) is used for each variable (Gene (G)).

$$G_i^L \leq G_i \leq G_i^U \quad (14)$$

G_i^L is the lower limit which can be represented with 6-bit string as 000000 while the upper limit gene that could be generated is 111111. The combination of the two makes 12-bits of a chromosome, $i \leq 10$, 10 is the population size of the individual to be reproduced or generated. This size is used to give enough search space for possible solutions (called individuals) and move towards the optimal one. 10 individuals are generated randomly. For instance, assuming population size is 5, and the cloudlet of size 22626 MB is to be executed, using equations 11 and 12. The binary form of the cloudlet will be 101101. An example of the generated population is shown in Fig. 2.

In this paper a threshold-based tournament selection was used. The selection involves randomly picking two individuals from the population and staging a tournament to determine which one gets selected. To get a parent, samples are made from the population in k times (with replacement), where k is the tournament size and 5 is assigned to it in irrespective of the population size.

The two individuals entered into crossing process using selection probability in equation 15 for the best-fit probability. A floating-point random value is generated, if the value is greater or equal to the selection probability (in equation 15), the fitter candidate is selected, otherwise if the value is equal to the second value, that is in equation 16 then the weaker candidate is chosen. The probability parameter provides a convenient mechanism for adjusting crossover process. In practise it is always set to be greater than 0.5 in order to favour fitter candidates. In this paper, the selection probability p was set to be:

$$a = \frac{\sum_{i=0}^{n} S_j \times 2^j}{64} \quad (15)$$

$$b = P \times (1 - P) \quad (16)$$

The selection probability (P):

$$P = \begin{cases} a \; iff \; a \geq 0.5 \\ b \; iff \; a \geq b \end{cases} \quad (17)$$

where n is 6 which signifies the 6-bit Strings (S) and S_j is the string bit at index $_j$. The tournament was extended to involve second individual to cater for the case when threshold limit of the first best is exceeded and the second value is therefore taken to be $p * (1 - p)$ [13]. This procedure is repeated until all the population members have been exhausted. The Best-fit value for crossing was 0.7 while 0.2 was taken as the second-fit value for crossing. Once the random number generated is greater or equal to best-fit value, the value at the ith random of individual-1 is used to replace the index value of the new solution after crossing. If the second fit value is the same with the random number generated, the value at the ith index of the individual- 2 is used to replace the corresponding ith index value in the new solution. At each tournament crossover, the final solutions are displayed. These are then mutated based on mutation rate of 0.015. Mutation is an operator used in genetic algorithm and its function is to maintain diversity from one generation of population of genetic algorithm chromosomes to the next and this involves altering one or more gene values in a chromosome from its initial state. According to Paulo Gaspar [14], using larger mutation rates prevents the genetic algorithm from converging quickly in order to find an optimal solution. Using small mutation rate leads quickly to good results, 0.015 was therefore used. From the result in Fig. 4, the target solution is a combination of gene_1 and gene_2 with gene_1 (which is the first 6 bits string) as the request or job size binary value and gene_2 (which is the last 6 bits string) as the virtual machine's binary value is then set to the lower limit value (that is, 000000).

5. ALGORITHM FOR META-TASK COMPUTATION

The approach used in achieving this, is by calculating the median of the average value of the completion time of the resources in each datacenter used in this paper. The minimum out of the two datacenters is calculated and the corresponding resource (virtual machine) was used to process the meta-task. With this approach, it was discovered that there is a close-form load balancing with respect to the makespan metric in the two datacenters.

Median-based meta-task scheduling algorithms

1. Let R represent resource
2. Let D represent Datacenter

3. Let C represent Completion time
4. Let J represent the Job/Cloudlet processed
5. R_i is the resource at index i where $i = \{1,2,3,4\}$
6. D_j is the datacenter at index j where $j = \{1,2\}$
7. $C_{i,j}$ is the completion time of resource at index i in datacenter j
8. $n_{i,j}$ is the number of cloudlets processed by resource at index i in datacenter j
9. Calculate the Average completion time by a resource is given by $\frac{C_{i,j}}{n_{i,j}}$
10. the median is found from the set of average values of the completion time across the number of resources used at each datacenter by ordering the set from lowest to highest and finding the exact middle.
11. Get the minimum out of the two middle numbers from the respective datacenter is calculated
12. Use the resource where this middle number (value) can first be found in the corresponding datacenter is chosen as the best-fit resource that will process the meta-task.

6. SIMULATION EXPERIMENT AND RESULT ANALYSIS

Fig. 3. shows the crossover computation of the two individuals at each tournament. The outcome of the crossover on the two individuals can either affects the New-solution or leave the solution intact, (i.e., nothing happens to it). The New-solution is gotten by merging the 6-bit string of cloudlet size and VM but the VM bits will be in lower bits (000000) all through. The condition for the change on new solution is based on the selection probability which was discussed in equation (15,16 and 17). The set of Final solutions at each tournament will then proceeded for mutation.

Fig. 4 shows the mutation computation of the final solutions from the crossover in Fig. 3. The mutation uses a conditional value of 0.015 adopted from Paulo Gaspa [14]. This value was said to be highest, but in this paper, the condition was taken to be lesser or equal to the value (0.015). At index 9 for a final solution of "011100100000", mutation occurred with a value of 0.006 which is less than 0.015. A random bit-string is then generated which is either 1 or 0 to replace the bit value at that index. In this case the bit-string generated are both 0s and are used to replace the value at each index. At the end of the whole tournament which involves series of crossover and mutation, the last value of the mutation is then taken as the best solution. It should be noted that the solution was the merging of gene_1 and gene_2 which in order words are cloudlet size and virtual machine respectively. The 6-bit string for virtual machine is the last 6 bits for the New-solution are reconverted to decimal number (base 10). If the value at the end of the conversion is zero (0), then no virtual machine is found for the processing of the cloudlet and therefore such cloudlet will be dispatched to meta-task queue else the cloudlet is dispatched to the available virtual machine for processing.

Samples of Generated Population

001100100001
001100100000
001100100010
011100100000
001100101010

Fig. 2. Examples of generated populations

Table 2. Simulation of meta-tasks on VM with minimum completion time

Cloudlet ID	Status	Data CenterID	VMID	Time	CloudletLength	Start Time	FinishTime
0	SUCCESS	2	1	6	3100	5	11
15	SUCCESS	2	1	5	2350	11	16
27	SUCCESS	2	1	3	1800	16	19
39	SUCCESS	2	1	3	1200	19	22
49	SUCCESS	2	1	2	1000	22	24

Fig. 3. Crossover implementation

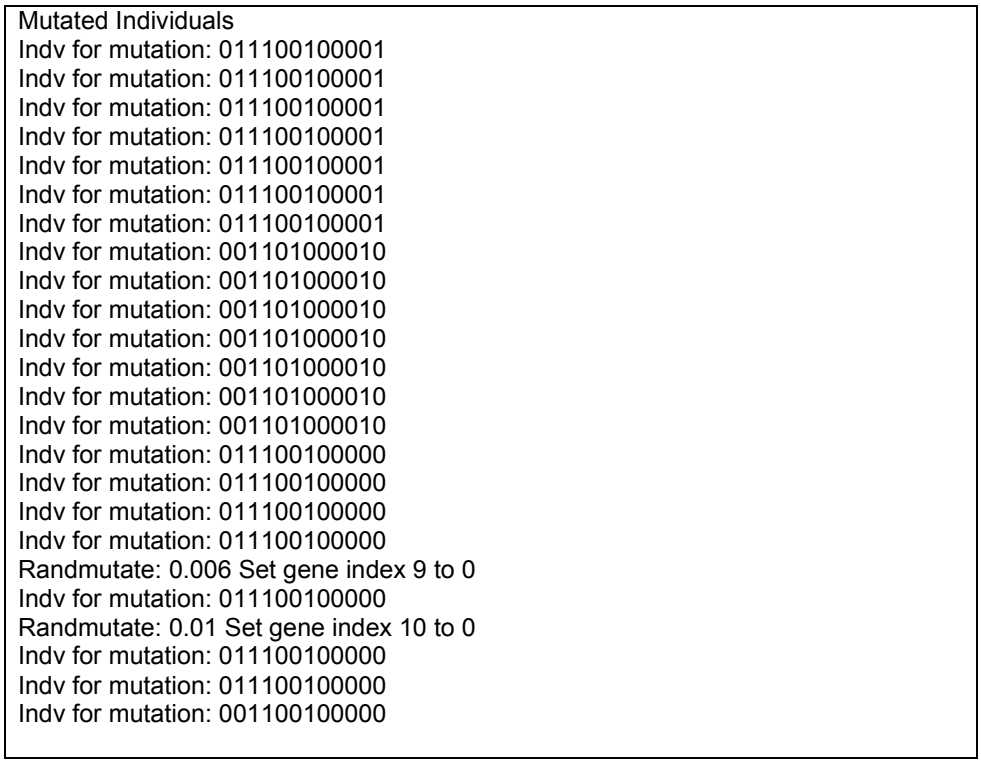


Fig. 4. Mutation implementation

Table 3. Calculation of VM with minimum completion time using median-based improved max-min

VM	No of Job done in DC 1	LFT of VM in DC 1	AVG of LFT of VM in DC 1	No of Job done in DC 2	LFT of VM in DC 2	AVG of LFT of VM in DC 2
0	6	68	11.3	5	35	7.0
1	1	23	23.0	2	5	2.5
2	8	46	5.8	6	20	3.3
3	10	177	17.7	7	39	5.4

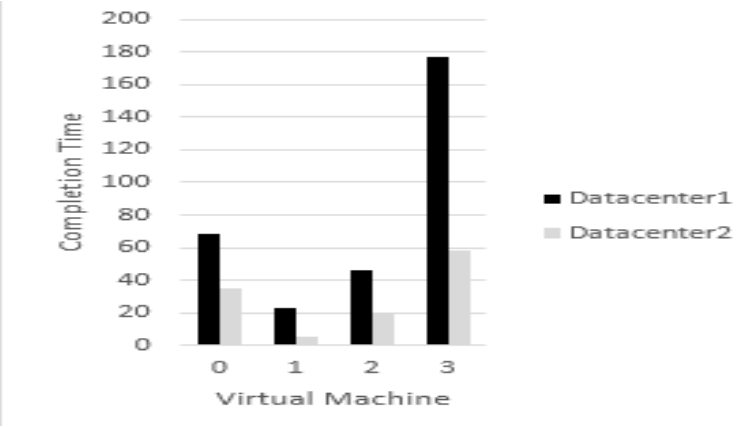
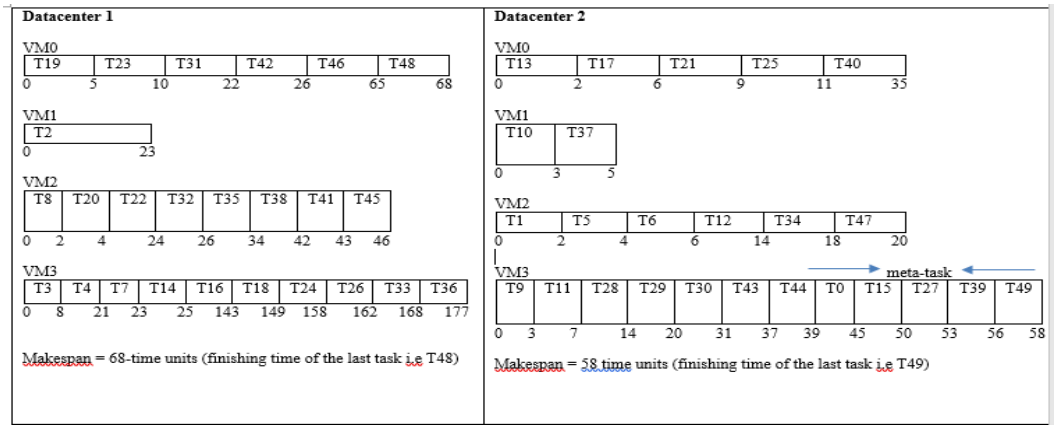


Fig. 5. Completion time of VM in datacenter 1 and 2 using median based-improved max-min

Table 4. Gant chart for scheduled tasks on datacenters 1 and 2 using Median-based Improved Max-Min algorithm (MIMM)



The Graph of completion time of virtual machine in datacenter 1and 2 after meta-tasks has been executed using Median Based-Improved Max-Min method is shown in Fig. 5.

Calculations of the desired metrics

The cloudsim 3.0 version has a pre-defined cost value for resources used. These are stated as follows:

the cost of using processing in VM resource is 3.0

the cost of using memory in VM resource is 0.05

the cost of using storage in VM resource is 0.001

In total the cost is 3.051 and it is uniform across the two datacenters

i.) $\boldsymbol{Economic\ Cost} = 3.051 * 68 + 3.051 * 23 + 3.051 * 46 + 3.051 * 177$

$= 207.47 + 70.17 + 140.35 + 540.03$
$= 958$ in data center 1

ii.) $\boldsymbol{Economic\ Cost} = 3.051 * 35 + 3.051 * 5 + 3.051 * 20 + 3.051 * 39$
$= 106.79 + 15.08 + 61.02 + 118.99$
$= 303$ in data center 2.

i.) $\boldsymbol{Resource\ utilization} = \frac{\sum_{i=1}^{n} Time\ taken\ by\ resource\ i\ to\ finish\ all\ jobs}{Makespan \times n}$ %

where *n* is the number of resources (VM).

For datacenter 1

RU = (68+23+46+177) / (68 * 4) = 314/272

= 1.15%

For datacenter 2

RU = (35+5+20+39) / (58 *4) = 99/232
= 0.43%

ii.) Throughput

For median-based improved max-min algorithm

Throughput = $25/68 = 0.37 jobs/secs$

Therefore, the throughput for datacenter 1= 0.37jobs/secs.

For datacenter 2 the throughput

= 24/58 = 0.43jobs /secs.

Table 5. Comparison of cost in both datacenters

Datacenter 1	Cost	Datacenter 2	Cost
VM0	207.47	VM0	106.77
VM1	70.17	VM1	15.08
VM2	140.35	VM2	61.02
VM3	540.03	VM3	118.99

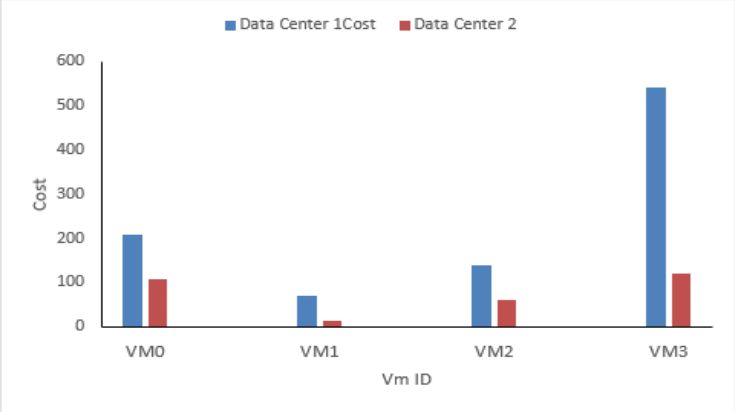


Fig. 6. Graph of against VMs

Comparison with other Algorithms:

Table 6. Comparison of Makespans in Datacenter 1 and 2

Algorithms	Datacenters	
	DC1	DC2
IMM	68	24
ACO	65	20
MIMM	68	58

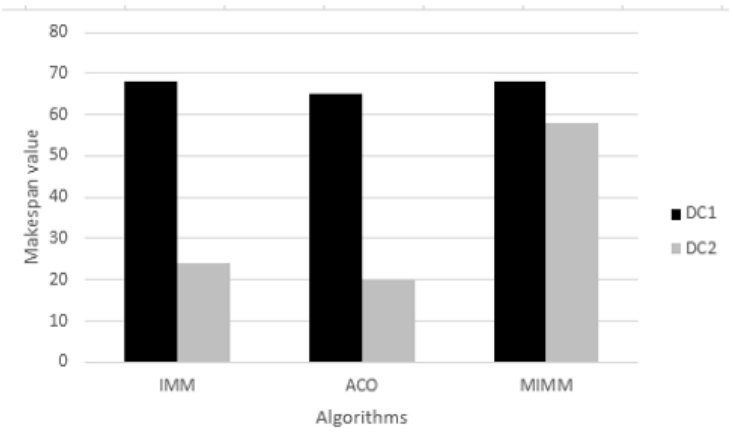


Fig. 7. The graph of the makespans

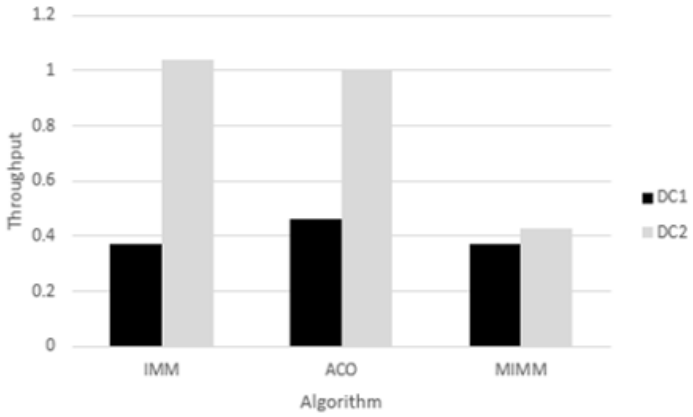


Fig. 8. Graph of throughput against the different allgorithms

Table 7. Comparison of throughput in datacenter 1 and 2

Algorithms	Datacenter	
	DC1	DC2
IMM	0.37	1.04
ACO	0.46	1
MIMM	0.37	0.43

Table 6, shows the makespans of both datacenter 1 and 2. IMM depicts the improved max-min, ACO depicts the ant colony optimization while MIMM is the median-based improved max min.

From Fig. 7, it shows that there is fairness in the distribution of cloudlets using the median-based improved max-min.

Table 7 shows the comparison of the different algorithms used to compare with the Median-based improved Max-Min algorithm used in this paper.

The calculations of the throughput of the different algorithms is plotted in the graph above to show the total number of jobs completing execution per unit time.

7. CONCLUSION

This paper focused on the resource scheduling challenges and load balancing that cloud computing is facing today. Several resource scheduling algorithms were compared with respect to the cloud workload as an answer for the dynamic scalability of resources. The

simulation result showed that the median-based improved Max-Min scheduling maximizes the makespan with load balancing and guarantees the high system availability in system performance. The median-based improved Max-Min algorithm was compared with improved max-min and ACO. Experimental results show that the new algorithm has a better quality of system load balancing and the utilization of system resources than others.

The experimental results gathered through cloudSim 3.0 clearly demonstrated that the proposed framework has better performance in terms of throughput, distribution and utilization as compared to existing scheduling algorithms. The results also showed the fairness in the utilization of the resources used in the two datacenters and the cost of resources decreased with increasing number of cloudlets.

For future work, scheduling algorithms that inspect the dynamic behavior of the resources and algorithms that allow tasks to be preempted according to a given priority and dynamically adapt the scheduling algorithm will be considered. Secured optimal resource allocation algorithms and frame work could be explored to strengthen the cloud computing paradigm.

Also, devising a strategy of dynamic resource allocation which will reduce the overall energy consumption rate of the datacenters in the Cloud could be considered.

Furthermore, the Vm Scheduler policy and cloudlets policy can also be alternated.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Atrey A, Jain N, Iyengar N, Ch. SN. A Study on Green Cloud Computing, IJGDC. 2013;93-102.
2. Shubhangi, Mehrotra. Resource allocation and scheduling in the cloud. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS). 2012;1(1).
   Available: www.ijettcs.org
3. Himani, Harmanbir SS. Comparative analysis of scheduling algorithms of cloudsim in cloud computing. International Journal of Computer Applications. 2014;97(16):0975-8887.
4. Chawda P, Chakraborty PS. An improved min-min task scheduling algorithm for load balancing cloud computing. International Journal on Recent and Innovation Trends in Computing and Communication. 2016;4:60-64.
5. Saraswathi AT, Kalaashri YRA, Padmavathi S. Dynamic Resource Allocation Scheme in Cloud Computing, Elsevier Procedia Computer Science. 2015;47:30–36.
6. Hamed Piroozfard, Kuan Yew Wong, Adnan Hassan. A hybrid genetic algorithm with a knowledge-based operator for solving the job shop scheduling problems. Journal of Optimization; 2016.
7. Priya, Babub NK. Moving average fuzzy resource scheduling for virtualized cloud data services. Journal of Computer Standards and Interfaces. 2018;50:251-257.
8. Bo Wang et al. Improving task scheduling with parallelism awareness in heterogeneous computational environments. Future Generation Computer System. 2019;94:419-429.
9. Krishnaveni H, Sinthu Janita Prakash V. Execution time based sufferage algorithm for static task scheduling in cloud. Advances in Big Data and Cloud Computing, Advances in Intelligent Systems and Computing. 2019;750.
   Available:https://doi.org/10.1007/978-981-13-1882-5_5.
10. Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, C´esar AF. De Rose, Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Published online 24 August 2010 in Wiley Online Library (wileyonlinelibrary.com); 2010.
    DOI: 10.1002/spe.995.
11. Maheswaran M, Ali sh, Jay siegel H, Hensgen D, Freund RF. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of Parallel and Distributed Computing. 1999;59:107-131.
12. Neha Gupa, Parminder Singh. Load Balancing Using Genetic Algorithm in Mobile Cloud Computing. International

Journal of Innovations in Engineering and Technology (IJIET). 2014;4(1). ISSN: 2319-1058.

13. Daniel W. Dyer. Tournament Selection: Selection Strategies & Elitism; 2010. Copyright Accessed on 18-10-2018.Url: Available:https://watchmaker.Uncommons.Org/manual/ch03s04.Html.

14. Paulo Gaspar. Why is the mutation rate in genetic algorithms very small. Vishwakarma Institute; 2018. Available:https://www.Researchgate.Net/post/why_is_the_mutation_rate_in_genetic_algorithms_very_small.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sdiarticle4.com/review-history/63913