Int. j. inf. tecnol. (June 2019) 11(2):265–275
https://doi.org/10.1007/s41870-019-00282-1

ORIGINAL RESEARCH

Load prediction analysis based on virtual machine execution time using optimal sequencing algorithm in cloud federated environment

S. K. Sonkar[1] · **M. U. Kharat**[2]

Received: 6 December 2017 / Accepted: 12 January 2019 / Published online: 2 February 2019

Abstract Virtual machine (VM) prediction and an effective resource management are the attractive areas in the cloud environment. VM prediction is an important task to execute the jobs for delay minimization and unnecessary states avoidance. Cloud computing attracted towards the increase in a number of applications that run on remote servers in parallel manner. Increase in parallelism reduces the CPU utilization adversely. Hence, the proper VM prediction and management are necessary stages in provisioning scheme. Also time required for allocating jobs is more in existing algorithms due to the number of computations involved. Therefore a novel algorithm is required to improve the performance of the job allocation with makespan reduction. In this paper the new algorithm is proposed that includes the VM capacity and execution time for load prediction and performance improvement purpose. Our proposed research work utilizes the VM clustering and optimization algorithms to improve job sequencing performance. The cost computation prior to clustering includes the VM capacity as a major factor. Clustering of VM with high-cost and isolation of low-cost and high-cost clusters reduces the searching time of VM and solve the imbalance state problem in traditional methods. The optimization algorithm with suitable initialization function reduces the time and steps for selection of VM for suitable job. The proposed model outperformance is established by the selected parameters.

Keywords Cloud environment · VM capacity · VM prediction · Resource management · Execution time

1 Introduction

In cloud computing applications, delivered as service over the Internet and that is provided with the help of hardware and system software of the datacenter (SaaS). Cloud computing is also called as utility computing, the service being sold. Amazon Web Services, Microsoft Azure, Google App Engine are few examples of the cloud service provider. Using cloud, end- user can access data anytime, anywhere [1, 2].

One of the best feature of cloud computing is resource elasticity. Because of which business customer can scale up and down utilization of resources as per their need without investing amount in licensed software and infrastructure. There are three types of cloud service models used widely that are Software as a Service (SaaS) which provides all support online so does not need any installation from client sites, Infrastructure as a Service (IaaS)—it provides the infrastructure such as storage, network, CPUs on demand, these resources are provided on rent basis and Platform as a Service (PaaS)—which is set of tools and services developed to make coding and deploying those applications quickly. In literature four cloud deployment models are discussed that are private, public, community and Hybrid cloud [3].

One of the key feature of cloud computing is virtualization technology which abstracts the physical infrastructure through a virtual machine monitor (VMM) or

✉ S. K. Sonkar
sonkar83@gmail.com

M. U. Kharat
mukharat@rediffmail.com

[1] Department of Computer Engineering, KKWagh Institute of Engineering Education & Research, Nashik, India

[2] Department of Computer Engineering, MET Institute of Engineering, Nashik, India

hypervisor [4]. VMM maps virtual machine to physical resources. VMM allows several VM or guest operating systems to share a single Physical Machine (PM) securely and fairly.

Also virtualization helps to relocate VMs to a minimum number of PM's therefore the number of active PM's can be reduced. This approach is called server consolidation. When some servers gets multiple requests from users to access the services and all other servers in datacenters are having only few requests, it means system is not able to balance the load, such problem is said to be a load balancing service problem [5, 6]. The load balancing problem can be solved by scheduling requests in a sequence so that efficient resource utilization can be done. In this regards, a good task and VM scheduler can improve the performance of resource utilization and avoid the load imbalance issue [7]. Our proposed system can solve this issue and provide the better load stability.

Rest of the paper is organized as follows: Sect. 2 describes related work. Our contributions are presented in Sect. 3. Section 4 represents proposed system architecture along with different algorithm to handle the system efficiently. Section 5 describes experimental results and discussion and Sect. 6 briefs about conclusion and future work.

2 Related work

This section presents different resource allocation and scheduling techniques which helps to present our proposed work.

Sonkar et al. [8] studied the several resource allocation and scheduling techniques and proposed an optimal resource management strategy which improve the overall utilization of server resource and also avoids load imbalance issue. Providing good QoS to the user is a critical challenge for Cloud Based Companies because workload demand is variable with respect to time. Calheiros et al. [9] solved this issue by introducing autoregressive integrated moving average (ARIMA) model which assesses accuracy of future workload prediction by using real traces of requests of web servers. da Rosa Righi et al. [10] presents elasticity model for high performance computing application in the cloud called Auto-elasticity. Xiao [11] proposed a resource allocation system which avoids overload of the system while minimizing the number of server used and also introduced the skewness concept which measure uneven utilization of server.

Farahnakian et al. [12] presented a dynamic Virtual Machine (VM) consolidation method called Utilization Prediction aware VM Consolidation (UP-VMC). To consolidate VMs into the least number of active Physical Machines system considers both the current and future utilization of resources. Walsh et al. [13] presents a distributed architecture, executed a precise model of data center which demonstrates how utility functions can permit for collection of autonomic elements to persistently optimize the utilization of computational resources in a dynamic and heterogeneous environment. The Li [14] presented adaptive resource allocation strategy for preemptible jobs by considering updated actual task execution. In this case static resource allocation performs static task scheduling which is generated offline and this can be done using adaptive list scheduling and adaptive min–min scheduling strategy. Best fit and worst fit dynamic VM Scheduling techniques are proposed by Rathor et al. [15]. In best fit, VM finding the host which supports the full utilization of its resources. The binary search method is used to search best fit host which reduces the VM allocation time. In worst fit VM allocation strategy, if first PM does not have sufficient resources, then in this case system power ON the new PM and allocate the VM on that machine. In Cloud environment When the number of VM increase or decrease also requests increase or decrease for such a situation a VM scheduling algorithm Artificial Bee Colony Longest Job First (ABC_LJF) [16] is proposed. This technique is suitable to maintain system stability and scheduling to prevent system crash. Hu et al. [17] proposed the load balancing scheduling technique of VM resources on the Cloud computing environment by implementing a tree structure utilizing Genetic algorithm for scheduling. The system which can resolve the quandary of load imbalance in Cloud computing by considering preceding data and the current state of work in advance to the performance demeanor. Panchal et al. [18] has discussed VM Load Balancing Algorithm of Weighted Active Monitoring which can implement in Cloud SIM Tools. This Load Balancing algorithm is proposed for the Data center to efficaciously load balance the application requests among the existing virtual machines allocating a certain weight, for to accomplish improved performance parameters such as replication and Data processing time. In First come first serve (FCFS) Scheduling is one of the simplest scheduling algorithm proposed by the author Yuan et al. [19]. Jobs are executed on first in first out basis. The disadvantage of this method it is no preemptive, i.e., average waiting time is high. Round robin scheduling algorithm is a preemptive scheduling algorithm by Raj [20]. In which each task have a fixed time to execution. If one of the task is not finish in it quantum then it require to wait for their next turn. Context switching is used to remember the state of the preempted task. Jobs are given priorities to execute in Generalized Priority algorithm proposed by Cao et al. [21]. The VM's are also prioritized, the highest priority is given to that VM which executes higher Million Instruction per Second

(MIPS). This scheme is better than RR and FCFS scheduling techniques. Li et al. [22] presented Ena Cloud technique in which application is encapsulates on a VM. This techniques supports applications scheduling and live migration to reduce the number of active machines, so that it save energy. Li et al. proposed [23] a balanced system which handles the VM demands in real time and providing assurance for better resource utilization. The total active physical machines and their energy consumption can be reduced as outcome of high resource utilization. Beloglazov et al. proposed [24] an effective resource management strategy for to constantly combine VMs leveraging live relocation and power off idle nodes to decrease power consumption along with essential Quality of Service. Buyya et al. [25] focuses the improvement of dynamic resource provisioning and allocation systems that reflect the interaction among numerous data center infrastructures and comprehensively effort to increase data center energy efficiency and performance.

3 Contribution

Our contribution to this work is as follows:

- Proposed new algorithm which includes VM capacity and execution time for load prediction and performance improvement.
- Proposed VM clustering and optimization algorithm to improve job sequencing performance.
- VM clustering algorithm introduced here to reduces searching time of VM and solves the imbalance state problem in traditional methods.
- The optimization algorithm used here which reduces the time for selection of VM for suitable job.

4 System model

Based on the research gap, proposed system is design that analyzes load prediction algorithm based on execution time of each task on Virtual Machine and sequencing them to improve service response time using optimal sequencing algorithm.

The proposed work implementation will be based on the architecture as shown in Fig. 1.

4.1 System workflow

In our proposed work, we have taken input parameters from cloudsim tool and then applied federation → clustering → job scheduling → prioritization.

The system workflow is explained as follows:

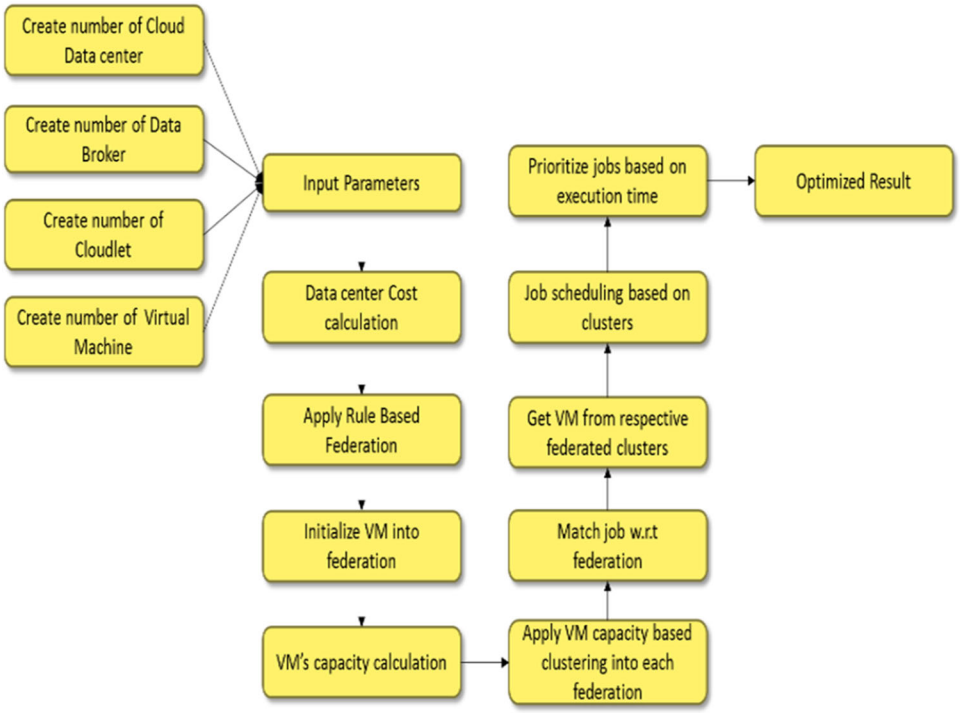
Load Prediction is done by applying optimal job and VM sequencing algorithm and the procedure for the same is done by retrieving inputs from cloudsim and stored into cloud database. Then VM capacity is calculated as (number of processing elements) × (million instruction per second) + bandwidth. Then created VM's are assigned to respective datacenter. Next step is to apply federation of the datacenter in which grouping the datacenters based on cost and Million Instruction per Second (MIPS). Further, based on the VM capacity clustering is applied into the federation. After performing the clustering, the jobs are initialized. Then based on job capacity, matched VM cluster is assigned. Next, after entering into VM cluster, the jobs are assigned to respective VM based on their available space and execution time. The Virtual machine execution time is calculated based on number of jobs are assigned to it.

In our system these jobs are executed by VM in a sequence. It means the jobs with lowest execution is executed first and job with higher execution time is executed subsequently. Like this our proposed system performs the optimal job sequencing algorithm. Further based on the total time required to execute all the jobs by VM, the VM's are also scheduled in sequence in VM cluster, i.e., VM which is having the less execution time having the priority first and then subsequent VM'S are scheduled for execution of jobs. Finally, load is predicted based on utilized capacity of VM and remaining capacity.

Following are the steps to implement sequencing algorithm.

1. Retrieve inputs from cloudsim tool and store it into database.
2. VM capacity is calculated based on VM capacity $=$ $(\text{nofpe} \times \text{MIPS}) + \text{bw}$
3. The created VM's belongs to respective datacenters, i.e., each and every datacenters have number of virtual machines. So, these virtual machines are categories based on their datacenters (DC).
4. Federation are implemented in this work. The federation is nothing but just grouping DC with respect to cost of the DC. Because, in real time scenario, sometimes users need services or Million instruction per second (MIPS) based on their required amount cost. So this work classifies the DC with respect to their cost and MIPS. In this work federation of DC are classified into following categories,
    - High MIPS with low cost
    - High MIPS with medium cost
    - Medium MIPS with medium cost
    - Low MIPS and medium cost

Fig. 1 System workflow



5. Due to the federation of DC Virtual machines also belongs to respective DC's federation group.
6. Here, we are applying clustering based on Virtual Machine's capacity in each federation groups. For example, if in federation1 there are 2 DC, DC1 have 10VM and DC2 have 7VM then these 17VM's belong to federation1. So, these 17VM's will be clustered based on their capacity.
7. After clustering of VM, the input jobs are initialized.
8. Based on the job's capacity, matched VM clusters are assigned. Then, after entering into the VM cluster job will be assigned to VM with respect to their available space and execution time.

Algorithm 1 - Clustering the Datacenters:
Calculate Mi of DC (Milli Instructions executed) for each Second
Calculate Cost/ 1 Mi(Cos Per Mips)
Calculate Cost/Mips= (Mips * Cost/Mi)
Dmips= Mips/3; (Mips by 3 (Low MIPS (LM), Medium MIPS (MM), High MIPS (HM)))
Dcost=Cost/3; (Low COST (LC), Medium COST (MC), High COST (HC))
Clustering the Datacenters:
Inputs:
DC-M=Mips of each Datacenter
DC-C=Cost of Each Datacenter
Output:
Clustered Data Centers
Start
Let DC_n be set of datacenters
Where n=1 to N. N=No of datacenters.
For (dc ← DC_i)
If (DC-M > HM && DC-C < LC)
Cluster1 ← dc
If (DC-M > HM &&DC-C<=MC)
Cluster2 ← dc
If (DC-M >= MM && DC-C<=MC)
Cluster3 ← dc
If (DC_M>=LM && DC_C<=MC)
Cluster4 ← dc
/*Calculate overall workload capacity of all 4 cluster datacenters*/
END

9. The Virtual Machine's execution time is calculated based on assigned job's execution time. For example if job1, job2, job3 are assigned to VM1 and their execution time such as 2, 3, 4 then the execution time for VM1 will be 2 + 3+4 = 9.
10. Based on execution time of Virtual machines, VM's are prioritized into each and every VM clusters. For example, VM1, VM2,VM3,VM4 into a belongs to same cluster then their execution time, respectively 8, 6, 7, 11 then it will be prioritized as VM2, VM3, VM1, VM4.
11. Then after assigning of jobs the utilized capacity is calculated to predict load applied on each Virtual Machine.

Thus, combining all above steps results in an efficient optimization algorithm.

4.2 Algorithms

The proposed system is implemented using following algorithmic strategies. Algorithm 1 is constructed for clustering the datacenters. Clustering of datacenter is performed based on datacenter cost and MIPS Parameter and outcome of this algorithm is system grouped datacenter into four clusters.

The algorithm 1 can be best explained as follows:

Initially each datacenter's calculated for the no of MIPS executed and the cost of the MIPS executed by each datacenter. A threshold value is set for the lower MIPS and upper MIPS. Also a threshold value is set for the lower and the upper limit for cost. Based on the threshold value MIPS and cost values are divided into three ranges namely Low, Medium, High. The Datacenters are Clustered based on the following order

Cluster 1: Maximum No of MIPS Executed in Minimum Cost → F1

Cluster 2: Maximum No of MIPS Executed in Medium Cost → F2

Cluster 3: Medium No of MIPS Executed in Medium Cost → F3

Cluster 4: Low No of MIPS executed in Medium cost → F4

For example a list has 5 datacenters having the MIPS value in List = 10, 20, 30, 40, 50;

So highest MIPS value is 50 so Max = 50; Max/3 = 50/3 = 16.66 (round off to 17) val1 = 17;

So result * 2; 17* 2 = 34; val2 = 34; val3 = max = 50;

Threshold values 17, 34, and 50

So take the list;

In Group1: (< val1)

Element = {10}; //Low MIPS

In group 2 (>=val1 && < val2)

Element = {20, 30}//Medium MIPS

In group 3: (>=val2)

Element = {40, 50}//High MIPS

As described in the above example, the created datacenters are listed in and grouped into three group vice, low, medium, high based on its MIPS rate.

The Same way the cost of the datacenters are also grouped into three groups are low medium and high.

Algorithm 2- VM placement:
Input: VMList
Output: VM Placements
Start
Load VMs to VMList along with Capacity
Load Datacenter Cluster List to DCList (cluste1, cluster2, cluster3, cluster4)
Let N= no of VM's to be placed
For each VM_i
If (Capacity (VM_i) <workload (cluster1))
List the physical machine in the datacenters → PMlist
Place VM_i in $PM_i$where (Capacity (VM_i) < Capacity (PM_i) in Dc_i in $Cluster(i)$)
Set VM_i status ← ready
Compute remaining Capacity ($Cluster_i$)
Update VMList and PMlist
Else
i=i+1
Move to place VM_i in next Cluster (i)
Update vmlist and hostlist
Continue above procedure
Place all the vm's in the respective physical machines
Until vmlist= empty
END

Algorithm3- Cluster VMs inside each Cluster of Datacenter:
Input: VMList, VmCapcity
Output: Clustering of VM's
Start
Sort VMs in each Datacenter-cluster (4 Clusters)
Compute Max= Max (Capacity of VM) in each cluster
Dmax=Max/3;
Fix Low (LT), medium (MT), high Threshold (HT) of Capacity (VM_i) in each cluster
For Each VM
If (Capacity (VM_i) > HT || Capacity (VM_i) =HT)
Put $VM_i \rightarrow F_1C_i$
Else if (Capacity (VM_i) >LT && (Capacity (VM_i) < MT || Capacity (VM_i) ==MT))
Put $VM_i \rightarrow F_1C_i$
Else if(Capacity (VM_i)< LT)
Put $VM_i \rightarrow F_1C_i$
/* As a general equation, $VM_i \rightarrow F_jC_i$ Where i refers vm cluster, j refers to datacenter cluster, This has to be applied for each datacenter cluster while placing the vm's .Group all VMs in each cluster to low, medium, high group based on capacity (vm) in each cluster */
END

The algorithm 2 is designed for VM Placements purpose, the VM can be placed on a Physical machine by checking the capacity of Physical machines in the clustered Datacenters.

The algorithm 3 is best explained with following example

As proposed system having 4 cluster of datacenter group, in each datacenter group, list the VM, take the capacity of VM in each cluster, and compute the maximum capacity of the VM.

Ex: if there are three group in datacenter F1, F2, F3, F4;

Consider F1 has 10 VM placed.

So compute capacity of the each VM, let VM capacity of each be in vmlist as

F1 = list {100,550,850,752,150,658, 200,300,400,685}
Maximum in list is 850;
Threshold is 850/3 = 283.3333 (round off to 284)
Threshold 1 = 284;
Threshold 2 = (284*2) = 568;
Threshold 3 = 850;
Condition 1: (< threshold 1)
F1_C1 = list {100,150,200}
Condition 2: (>=threshold1 && < threshold2)
F1_C2 = list {300,400,550}
Condition 2: (> threshold2 && <=maximum value)
F1_C3 = list {658,685,752,850}

Similarly doing this for all the federated group of datacenters.

Algorithm4- Job Allocation:
Input: Cloudlets or Jobs
Outputs: Prioritizing jobs
START
Choose Jobs to be allotted to the VMs.
Let N= no. of Jobs
Let JList be the List of Jobs
Let VList be the List of VM's in each Cluster
Let Glist be the VM's in each cluster Group (sorted in high, medium, low order of capacity)
For each Job_i in JList
If (Size (Job_i) <VM_i *in Glist*))
Allocate Job_i to VM_i in Glist
Else if
Move to next cluster
If all cluster is visited and Job_i is not allotted.
Place in waitlist
Continue until all job are allotted
Estimate the execution time (EET) of each job in each VM.
Prioritize the Jobs based on the $EET(Job_i)$
Compute Predicted Load of each VM
END

In Algorithm 4 the Job size are consider as a parameter and the conditional probability is calculated for all the jobs. Here the conditions are

1. job size $>$ 4000
2. job size $<$ 4000 and $\geq$2000
3. job size $<$ 2000

The probability value is calculated for the jobs that are satisfying the conditions.

As there are two conditions to compute P (a) and P (b);

Then the conditional probability will be P (a|b) = P (ab)/P (b) or simply P (a|b) = P (a) $\times$ P (b)/P (b)

$$P(a|b) = \frac{\mathrm{P(ab)}}{\mathrm{p(b)}} \text{Or simply} P(a|b) = \frac{\mathrm{P(a)} \times \mathrm{P(b)}}{\mathrm{p(b)}} = p(a). \tag{1}$$

So in our implementation, it is required to compute the execution time based on the job size.

System group the job size into $<$ 2000, 2000–4000, $>$ 4000;

So in the first case, and the third case p (a) is only provided

Case 1: probability of jobs with job size $<$ 2000;

$$\text{Therefore, p (job } < 2000) = \frac{\text{no of jobs } < 2000}{\textit{total no of jobs}}. \tag{2}$$

Case 3; probability of jobs $>$ 4000;

$$\text{Therefore, p (job } > 4000) = \frac{\text{no of jobs} > 4000}{\textit{total no of jobs}}. \tag{3}$$

But in case 2; there are 2 conditions; jobs $>$ 2000 and jobs $<$ 4000;

$$\text{Therefore; p (jobs } > 2000 \text{ and jobs } < 4000) = \frac{p(jobs > 2000) \times p(jobs < 4000)}{p(jobs < 4000)}. \tag{4}$$

Based on this conditional probability, the execution time of job is estimated with the three possible conditions.

Job are assigned to the VM's based on the broker associated to it. The broker can assign jobs to the VM's that are only allocated to them.

5 Performance analysis

In real Time the Datacenter capacity will be of huge size. It is not really possible to show creation of data centers and virtual machines. So the System is implemented using cloudsim tool having physical configuration of is i3 processor, 4 GB RAM and 1 TB HDD.

Figure 2 Illustrates total time needed to execute all the jobs by a VM, the VM's are also scheduled in sequence in VM cluster, i.e., VM which is having the less execution time having the priority first and then subsequent VM's are scheduled for execution of jobs. It is observed that within VM if there are two jobs for ex. job 19 having execution time 671 and job 0 having execution time 658, then by applying our proposed system these jobs are also sequenced that is job having less execution time, i.e., job 0 will be given first priority for execution then job 19 as shown in Fig. 2.

Then Load is predicted based on utilized capacity of VM and remaining capacity as shown in Fig. 3. The load of each virtual machines is calculated by extracting the total capacity and the utilized capacity of the each and every virtual machine, i.e., Load (L) = (total capacity − utilized capacity)/total capacity $\times$ 100.

Figure 4 shows the Graph for the first parameter, i.e., accuracy of VM placement in different federated cluster. It is observed that maximum 93% accuracy is achieved to place the VM's in Federated datacenter. When there are 55 virtual machines to be allocated to the federated datacenters (FDC), the best datacenter will be chosen by the VM placement algorithm. Thus, the placement accuracy is a measure of the correctly placed VM's in the best datacenters while comparing with the total number of virtual machines. This measure is computed for each and every federated datacenter.

Second parameter is load stabilization or balancing which is the measure of the rate of correctly migrated jobs to the best virtual machines while comparing with the total number of jobs which has to be migrated due to overload or under load. This measure is computed while varying the number of virtual machines. Load stabilization should be high. When it is low, it represents that, the proposed methodology is not able to stabilize or migrate the unallocated jobs effectively. In case, if the number of virtual machines are small, stabilization rate may be low. When the number of virtual machines are considerably large, the stabilization rate will be high and the proposed model will show the same result as shown in Fig. 5—if there are 2500 virtual machines then the load stability percentage is 25% which shows that as compared to other methods our system is outperforms.

Third parameter is response time. The Fig. 6 compares the response time taken by the VM'S. The response time is the measure which represents the time taken to complete the executions of the jobs which are allocated to the virtual machines. The datacenter can hold 'n' number of virtual machines placed in it. The response time increases, if the number of virtual machines increases. Since the increase in the number of virtual machines shows the increase in the availability of the resources for the execution of the jobs in the datacenter. As shown in Fig. 6 for 3500 Virtual machines only 25 ms are required for getting response, whereas for the same 3500 VM's using Greedy algorithm

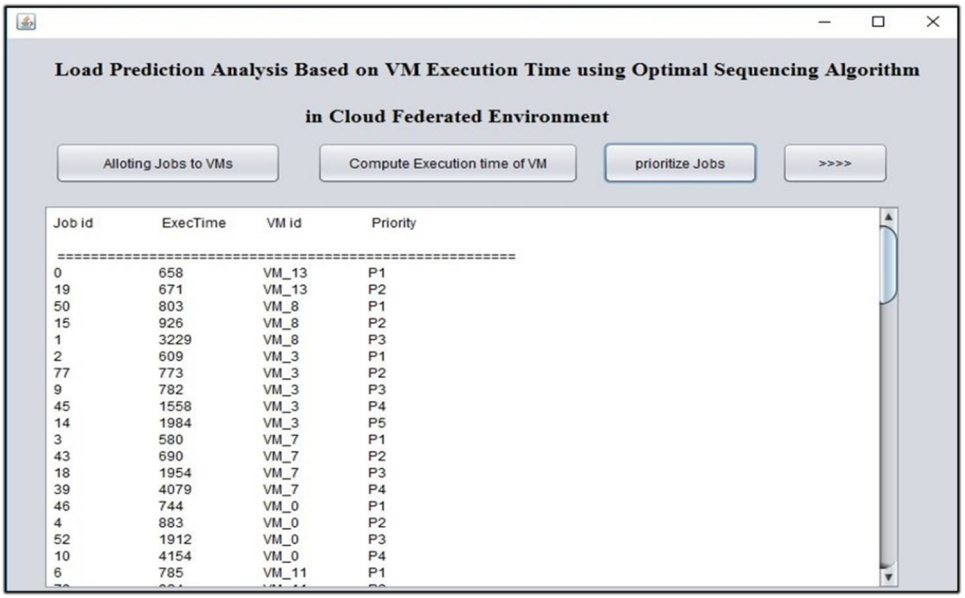


Fig. 2 Displaying priority of jobs

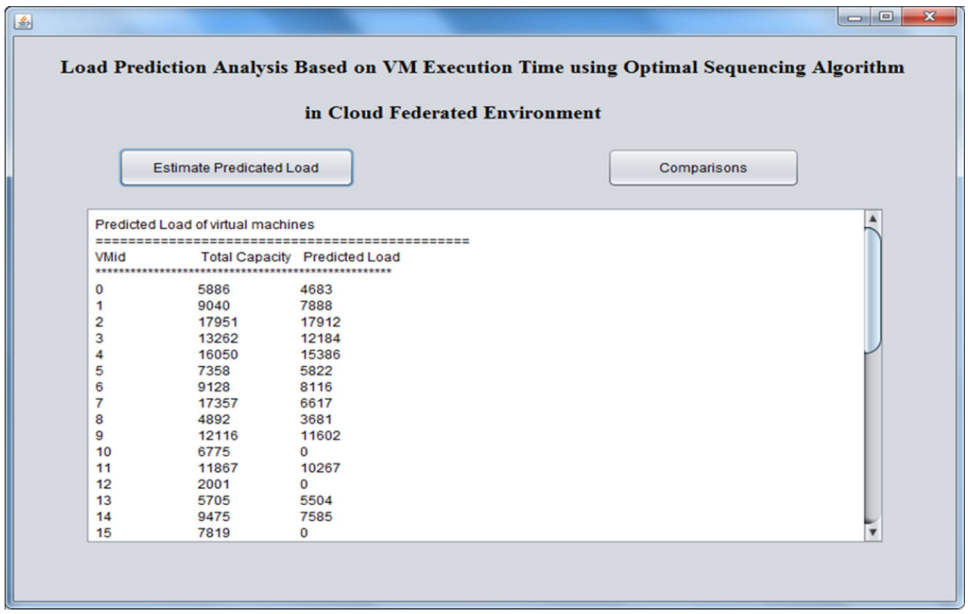


Fig. 3 Displaying predicted load

take 50 ms. As per the observation in Fig. 6 the proposed system is very effective as compared to all other systems.

Figure 7 shows the graph for response time when the jobs are increased. Job scheduling time increases with the increase in the number of jobs. But when simultaneously the virtual machines are increased, the job scheduling time decreases. Since when minimal number of virtual machine refers to job allocations, waiting time and execution time. It is observed that proposed system takes only 110 ms for 7000 jobs whereas ABC_FCFs takes 200 ms to respond the 7000 jobs. As we have compared the proposed system with three existing methods and it is concluded that proposed system outperforms as compared to the all other existing systems.

The Fig. 8 shows the reduction in response time of the virtual machine when the number of jobs needed to be executed are fixed and the number of virtual machines are increased in each case. It is pointed that only 40 ms are required as a response time for 21,000 virtual machines. So it takes very less time (as shown in Fig. 8 with pink line) to execute the jobs when more no of VM's are provided. The execution time is less because execution time refers to time which is taken to complete the allocated task. The execution time of our proposed framework is less as compared to other existing system because of our two major

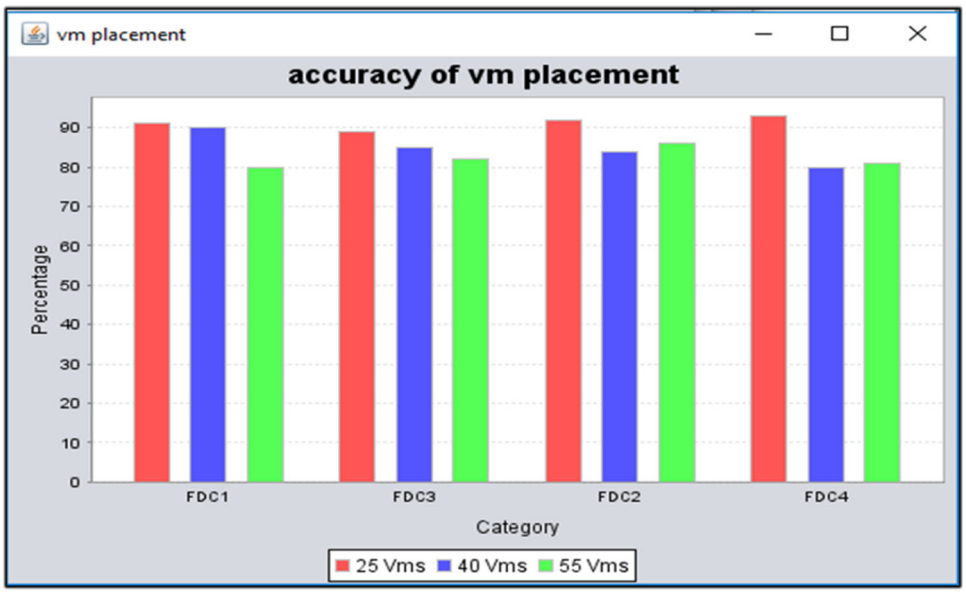


Fig. 4 Graph for accuracy of VM placement

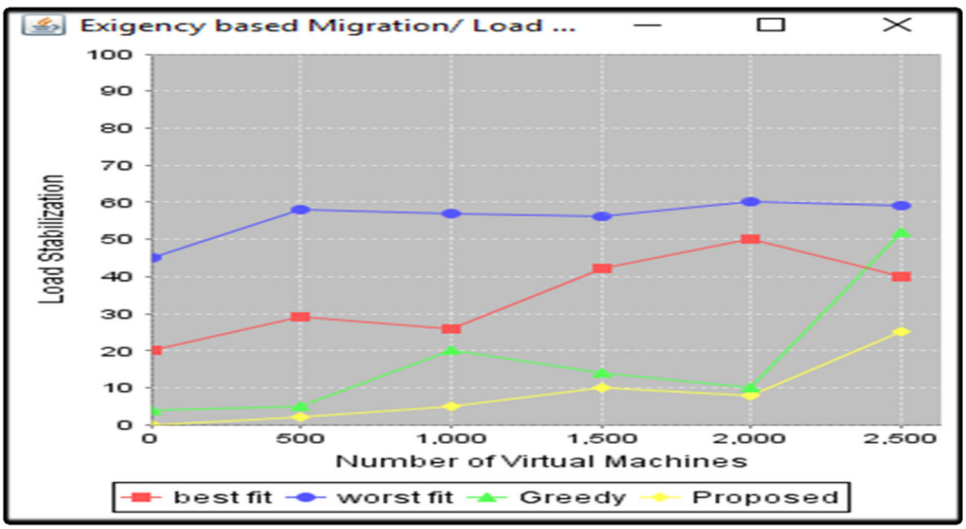


Fig. 5 Graph for load stabilization: no. of VM's vs load stabilization in %

contributions, First is the federations of the data center group's similar datacenters and helps the allocation of the similar jobs to a datacenter and second is our job scheduling model effectively predicts the best jobs to be placed to the best virtual machine for speedy execution.

6 Conclusion and future work

This paper presented a load prediction algorithm based on execution time of each task on Virtual Machine and sequencing them to improve service response time using optimal sequencing algorithm. Also this paper presented a VM Placement technique, which reduces time needed to place a virtual machine to physical machine- to respond the user's request. Finally, this paper discussed job priority algorithm based on execution time of job which improves the system performance. The proposed scheduling strategy was simulated using Cloudsim tools.

Our model outperformance is established by the parameters-high accuracy of virtual machine placement, high rate of load stabilization, minimal job scheduling time, and minimal response time. All these parametric responses shown that our proposed model outperforms.

As a future work, we are to planning to design a system which can reduce the energy consumption. In this view a novel energy efficient resource management model can be designed to handle resource scheduling and for the minimization of energy utilized by the cloud data centers for the computational work.

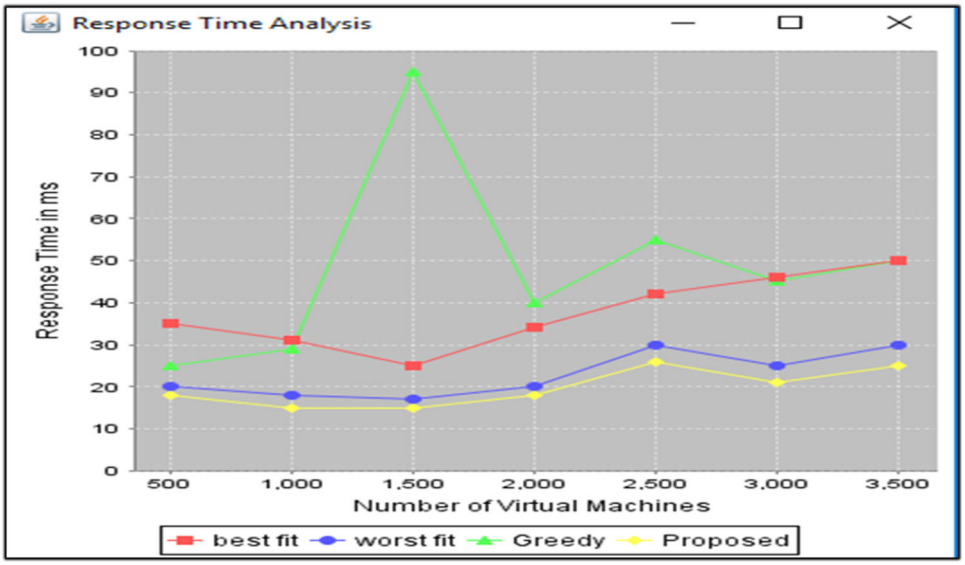


Fig. 6 Graph for number of VM's vs response time

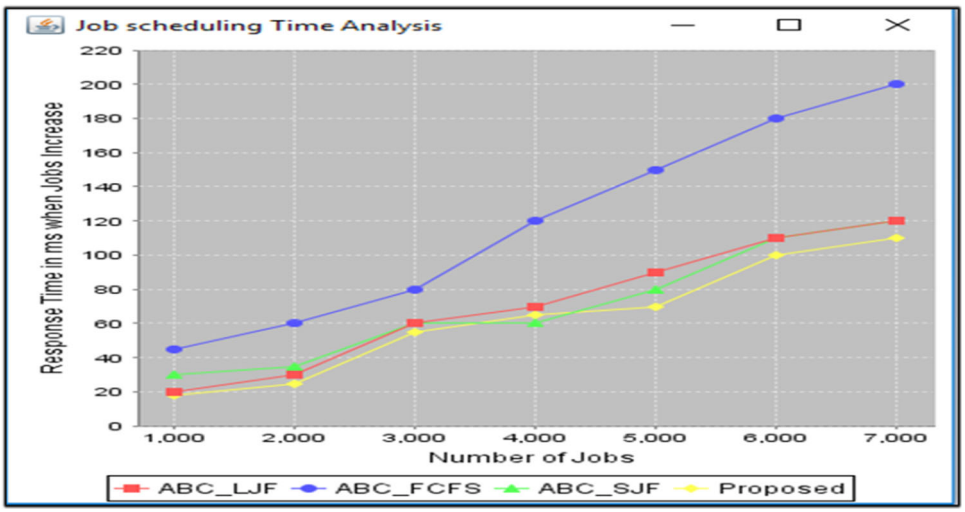


Fig. 7 Graph for increased number of job vs response time

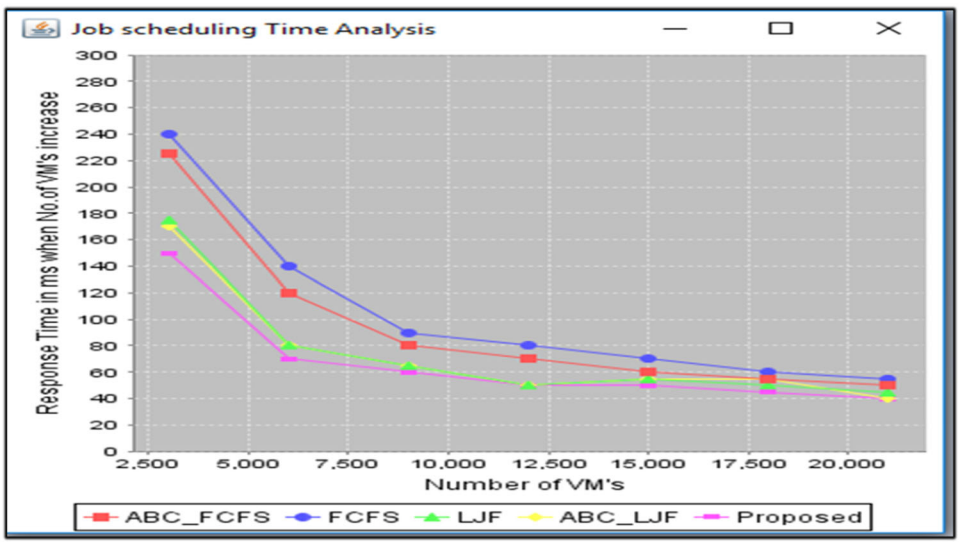


Fig. 8 Graph for increased no. of VM's vs response time

Acknowledgements This work was supported by BCUD SP Pune University (Grant no. 15ENG000654).

References

1. Armbrust M et al (2009) Above the clouds: a Berkeley view of cloud computing. University of California, Berkeley **(Tech. Rep.)**
2. Prasad AS, Rao S (2014) A mechanism design approach to resource procurement in cloud computing. IEEE Trans Comput 63(1):17–30
3. Bohli J-M, Gruschka N, Jensen M, Iacono LL, Marnau N (2013) Security and privacy-enhancing multicloud architectures. IEEE Trans Dependable Secure Comput 10(4):212–223
4. Chang E-H, Wang C-C, Liu C-T, Chen K-C, Chen C-H (2014) Virtualization technology for Tcp/Ip offload engine. IEEE Trans Cloud Comput 2(2):117–129
5. Xiao Z, Chen Q, Luo H (2014) Automatic scaling of internet applications for cloud computing services. IEEE Trans Comput 63(5):1111–1123
6. Zhang H, Jiang G, Yoshihira K, Chen H (2014) Proactive workload management in hybrid cloud computing. IEEE Trans Netw Serv Manag 11(1):90–100
7. James J, Verma B (2012) Efficient VM load balancing algorithm for a cloud computing environment. Int J Comput Sci Eng (IJCSE) 4(09):1658–1663 **(ISSN: 0975-3397)**
8. Sonkar SK, Kharat MU (2016) A review on resource allocation and VM scheduling techniques and a model for efficient resource management in cloud computing environment. IEEE Int Conf ICTBIG. https://doi.org/10.1109/ictbig.2016.7892646 **(ISBN: 978-1-5090-5519-9)**
9. Calheiros RN, Masoumi E, Ranjan R, Buyya R (2015) Workload prediction using ARIMA model and its impact on cloud applications QOS. IEEE Trans Cloud Comput 3(4):449–458
10. da Rosa Righi R, da Costa CA, de Bona LCE (2016) AutoElastic: automatic resource elasticity for high performance applications in the cloud. IEEE Trans Cloud Comput 4(1):6–19
11. Xiao Z, Song W, Chen Q (2013) Dynamic resource allocation using virtual machine in cloud computing environment. IEEE Trans Parallel Distrib Syst 24(6):1107–1117
12. Farahnakian F, Pahikkala T, Liljeberg P, Plosila J, Hieu NT, Tenhunen H (2016) Energy-aware VM consolidation in cloud data centers using utilization prediction model. IEEE Trans Cloud Comput. https://doi.org/10.1109/tcc.2016.2617374
13. Walsh WE, Tesauro G, Kephart JO, Das R (2004) Utility functions in autonomic systems. In: ICAC'04: proceedings of the first international conference on autonomic computing. IEEE Computer Society, pp 70–77
14. Li J, Qiu M, Niu J-W, Chen Y, Ming Z (2011) Adaptive resource allocation for preemptable jobs in cloud systems. In: 10th International conference on intelligent system design and application, pp 31–36
15. Rathor VS, Pateriya RK, Gupta RK (2014) An efficient virtual machine scheduling technique in cloud computing environment. IJCS 1(1):1–14 **(ISSN: 2287-8491)**
16. Kruekaew B, Kimpan W (2014) Virtual machine scheduling management on cloud computing using artificial bee colony. In: Proceedings of the international multiconference of engineers and computer scientists 2014, vol I, IMECS 2014, Hong Kong
17. Hu J, Gu J, Sun G, Zhao T (2010) A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In: 3rd IEEE international symposium on parallel architectures, algorithms and programming (PAAP), 2010, pp 89–96
18. Panchal B et al (2013) Dynamic VM allocation algorithm using clustering in cloud computing. Int J Adv Res Comput Sci Softw Eng 3(9):143–150
19. Yuan H, Li C, Du M (2014) Optimal virtual machine resources scheduling based on improved particle swarm optimization in cloud computing. J Softw 9(3):705–708
20. Raj G (2012) Effective cost mechanism for cloudlet retransmission and prioritized VM scheduling mechanism over broker virtual machine communication framework. Int J Cloud Comput Serv Archit 2(3):41–50
21. Cao Y, Ro C (2012) Adaptive scheduling for QoS-based virtual machine management in cloud computing. Int J Contents 8(4):7–11
22. Li B, Li J, Huai J, Wo T, Li Q, Zhong L (2009) EnaCloud: an energy-saving application live placement approach for cloud computing environments. In: IEEE international conference on cloud computing, pp 17–24. https://doi.org/10.1109/cloud.2009.72
23. Li X, Qian Z, Chi R, Zhang B, Lu S (2012) Balancing resource utilization for continuous virtual machine requests in clouds. In: Sixth international conference on innovative mobile and internet services in ubiquitous computing (IMIS), IEEE conference publication, pp 266–273. https://doi.org/10.1109/imis.2012.72
24. Beloglazov A, Buyya R (2010) Energy efficient allocation of virtual machines in cloud data centers. In: 10th IEEE international conference on cluster, cloud and grid computing (CCGrid) 2010, pp 577–578. https://doi.org/10.1109/CCGRID.2010.45
25. Buyya R, Beloglazov A, Abawajy J (2010) Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. In: International conference on parallel and distributed processing techniques and applications (PDPTA 2010), Las Vegas, USA