Indian Journal of Science and Technology, *Vol 8(17), DOI: 10.17485/ijst/2015/v8i17/59140, August 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Energy Efficient Multi Dimensional Host Load Aware Algorithm for Virtual Machine Placement and Optimization in Cloud Environment

T. Thiruvenkadam[1*] and P. Kamalakkannan[2]

[1]Department of Computer Science, K.S.Rangasamy College of Arts and Science, Tiruchengode - 637215, Tamil Nadu, India; mailone.thiru@gmail.com
[2]Department of Computer Science, Arignar Anna Govt., Arts College, Namakkal - 637002,Tamil Nadu, India; Kamal_karthi95@yahoo.co.in

Abstract

The effectiveness and elasticity of virtual machine placement has become a main concern in modern cloud computing environment. Mapping the virtual machines to the physical machines cluster is called the VM placement. In this paper we present an efficient hybrid genetic based host load aware algorithm for scheduling and optimization of virtual machines in a cluster of Physical hosts. We used two different techniques, first initial VM packing is done by checking the load of the physical host and the user constraints of the VMs. Second optimization of placed VMs is done by using a hybrid genetic algorithm based on fitness function. The presented algorithm is implemented in JAVA Net beans IDE, and Clouds simulator has been used for simulation to assess the execution and performance of our heuristics by comparison with algorithms first fit, best fit and round robin. The performance of the proposed algorithm was examined from both users and service provider's perception. The simulation results show that our proposed algorithm uses the less number of physical servers for placing a certain number of VMs which helps to improve the resource utilization rate. The response time of our algorithm is little bit more than the first fit algorithm because of its nature of allocating VMs is based on the user constraints and past usage history of the VMs. Elevated SLA satisfaction rate and inferior load imbalance rate was observed in results. Since we used a modified version of hybrid genetic algorithm for load optimization the percentage of VM migrations had been decreased through which we can achieve the better results for load balancing along with cost reduction. The results also show that our hybrid genetic based multi dimensional host load aware and user constraints based algorithm is applicable, valuable and reliable for implementation in real data center environments.

Keywords: Host Load Aware Algorithm, Load Monitoring, Load Rebalancing, Physical Machine Cluster, Virtual Machine, VM Scheduling

1. Introduction

Infrastructure-as-a-Service (IaaS) is the most fundamental use of cloud computing. The virtualization technology is the base to form an IaaS platform. This proposes the entire computing resources for deploying and executing applications, storing data, or accommodating a company's complete computing environment[1]. Virtualization technologies guarantee opportunities for cloud data centers to host applications on shared infrastructure. Data center expenses can be lessened by using Virtual Machines (VMs) Cloud data center providers can create a huge number of Virtual Machines (VMs) for different types of workload and specification requirements. [2]Each VM is configured with a certain amount of computing resources which is adequate with workload requirements. The cloud service providers can consolidate all the VMs into a few numbers of physical hosts, keeping in mind the end goal to lessen the aggregate number of obliged physical servers and abusing server capacities all the more completely,

*Author for correspondence

permitting cloud providers to spare cash on equipment and vitality costs. VM consolidation method is the key sympathy toward attaining economy of scale in a cloud data center domain[3]. The advent of virtualization technology enables the physical server consolidation in data centers which plays a vital role in minimizing the number of physical servers used and energy consumption also. Various approaches has been provided by the researchers for server consolidation in data centers but none of them have been considered all the aspects of the server consolidation which ensures the QOS as well as reduced cost for the datacenter administrators. Therefore a new algorithm is needed in order to provide better service to the cloud users and at the same time reducing the operational cost to the service provider[4]. Placing the VM in the appropriate host is necessary for ensuring the effective resource utilization and minimizing the datacenter cost as well as power. To address this problem in this paper we propose a new efficient hybrid genetic based host load aware algorithm for scheduling and optimization of virtual machines in a cluster of Physical hosts. We divide this problem into two following categories[5].

1.1 Initial Scheduling of VMs

The Virtual Machine allocation problem in a cloud infrastructure is investigated by many researchers in the past. But the majority of the presented mechanisms paid no attention to the ever changing load of the physical host and dynamic nature of the Virtual Machine deployment requests that frequently reaches the cloud provider infrastructure. Here we present an efficient hybrid host load aware algorithm for scheduling virtual machines to a cluster of Physical hosts. We developed the algorithm based on two different methods, first by checking the load of the physical host, the load factor of a physical host can be measured by the way of analyzing utilization level of the individual resources like CPU, Memory and Network bandwidth. Second by considering the past utilization activities of a VM to a physical host.

1.2 Ongoing Load Rebalancing or Optimization

Rebalancing of load in datacenter environment need live VM migrations but more number of frequently moved VMs between physical hosts causes increased network bandwidth utilization and datacenter cost hence the load rebalancing has to be achieved with minimum number of VM migrations in order to solve this issue we used a modified version of hybrid genetic algorithm for load optimization. The main contribution of this paper includes the introduction of virtualization technology, a new proposed algorithm for initial VM scheduling, ongoing load rebalancing or optimization and validation of the proposed algorithm on a simulated environment for its goals.

The rest of the paper is organized as follows: In Section II we describe the related work while in Section III placement problem under study has been explained, we present the design model to explain the proposed strategy in section IV The proposed algorithm for VM scheduling is discussed in section V. Load balancing and VM optimization based on genetic algorithm is presented in section VI. Section VII shows the experimental setup and results acquired by our technique compared with some of the existing strategy for optimal VM placement and optimization. Section VI concludes the paper and spotlights some possible future directions. Most of the IaaS cloud data centers uses virtualization technology since it provides a good flexibility in the provisioning and placement of servers and their associated workloads and cost savings[6,7] while this model provides a number of advantages, it is essential to administer the allocation of virtual machines to the physical hosts in the data center.

2. Related Work

Even though a lot of researchers have been studied this virtual machine mapping problem in the past we draw attention to some of the closest work in perspective of our point. In[8] the number of physical machines needed to deploy the requested virtual machine instances are reduced by combining time series forecasting techniques and bin packing heuristic but the model has not included the relationships between multiple resources, like CPU and I/O. In[9] the VM placement algorithms make use of the behavior of VMs to have some properties in general. In[10] for the placement of virtual machines to physical machines a two level control management system is used and it uses combinatory and multi-phase efficiency to solve potentially inconsistent scheduling constraints. In[11], VM scheduling constraints are considered as single dimension in a multidimensional Knapsack problem.

In[12], the VM scheduling policy is primarily dealt out from the viewpoint of network traffic and three common scheduling algorithms have been introduced for Cloud

computing and simulation results provided. In[13] the performing load balancing in data centers are intensively studied the heuristics has been used as a common approach among systems to enables the load balancing among physical servers. In[14] the performance variations have been identified and monitored in a physical server hosting VMs. A few simple VM placement algorithms like time-shared and space-shared were presented and compared in[15] and introduced a method to model and simulate Cloud computing environments, in which the algorithms can be implemented. In[16] pioneered methods for allocating and migrating virtual machines and proposed some migration techniques and algorithms based on the load imbalance level of the servers. [17]Evaluated most important load-balance scheduling algorithms for conventional Web servers. Vector Dot a novel load-balancing algorithm has been introduced in[18] to work with structured and multi-dimensional resources limitations by taking servers and storage of a Cloud into account. A countable measure of load imbalance on virtualized data center servers has been proposed in[19]. In[20] server consolidation was considered as a stochastic been packing problem and presented a VM sizing based algorithm which considers the cumulative resource demand of a host where the VM to be placed. An overloaded resource based VM placement approach has been presented in[21]. In our previous study[22] the comparison of various VM scheduling algorithm has been presented and demonstrated the necessity of new efficient placement VM placement algorithm. An algorithm for scheduling virtual machines have been presented in[23] based on user constraints and multi dimensional host load.

A genetic based simulated annealing algorithm for optimization of task scheduling in cloud computing has been proposed and implemented in[24]. This algorithm only considers the QOS necessities of various types of tasks. Some of the genetic operators that use the group-oriented structure lead the better results when compared to the non-grouping genetic based algorithms which are not use such grouping feature. In[25,26] they used the grouping based genetic algorithm to reach better results than conventional methods and universal heuristic algorithms.

3. Problem Formulation

The major principle of the IaaS cloud computing system is that its user can make use of the resources to have good performance and economic benefits. With the support of virtualization innovation the resources can be conveyed to the users in the form of virtual machines hence an efficient virtual machine allocation policy and management process is required to avoid under utilization or over utilization of the physical machines which may affect the quality of services of the IaaS cloud. The under utilization of servers is a well known expenditure concern in cloud management. Low utilization of server resources leads to the usage of more physical machines, increasing expenses for machine power and capital and operational expenses for cooling systems. Moreover, surplus machines require more carbon footprint. The over utilization of physical servers results in violating the SLA and quality of service constraints. Efficient allocation of Virtual machine instance request will meet client requirements, improve the resource utilization, increases the overall performance of the cloud computing environment and also decreases the number physical machines used. Therefore an efficient VM scheduling and ongoing load monitoring and optimization in IaaS is an important cloud computing problem to resolve.

4. System Architecture

To address the VM scheduling and ongoing load optimization problem we have proposed a multi dimensional physical host load aware scheduling and hybrid genetic based optimization algorithm and we implemented this heuristics in JAVA using Net beans IDE.

Figure 1 shows the framework model in which the proposed algorithm is implemented. Here the physical clusters can be formed by adding a set of physical servers each server contributing its own share of resources such as CPU cores, main memory, disk capacity and network bandwidth. The users can create virtual machine instances by giving their requirements for running the applications and the VM requests are submitted by the users to the computing system. As the submitted VMs enter to the cloud they are wait for their turn in the stack. The VM requests can be handled by the virtual machine scheduler and it finds the appropriate physical machine by estimating the VM size and checking for the availability and capacity of the physical machine when it finds the appropriate physical machine the VM scheduler immediately allocates the identified physical machine to the virtual machine instance request in queue and the required resource can be allocated to the virtual machine. The proposed algorithm uses the index table for efficient VM placement with the properties shown in Table 1.

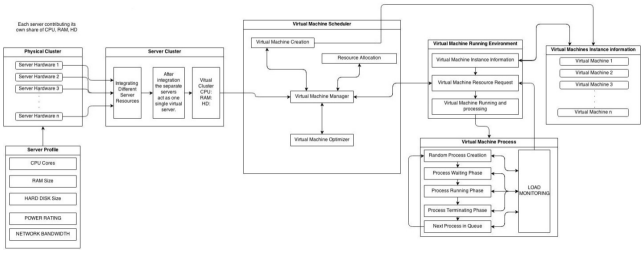

Figure 1. Framework model for VM placement in a cluster of physical machines.

Table 1. Properties required for the index table of physical machine and physical machine cluster

S.No	Physical Machine	Physical Machine Cluster
1	Total number of VMs placed	Total number of PMs
2	Total number of VMs in each type (CPU intensive, RAM intensive, N/W intensive)	Total number of PMs exhausted
3	The percentage of load of the PM in each resource type individually	The cumulative percentage of the load of the entire PMs
4	Total number of CPU cores utilized and available	The list of PMs which can be used to place the CPU intensive VMs
5	Total amount of RAM utilized and available	The list of PMs which can be used to place the memory intensive VMs
6	Amount of N/W bandwidth utilized and available	The list of PMs which can be used to place the N/W Bandwidth intensive VMs

Rebalancing of load in this environment is handled by virtual machine optimizer we used a modified version of genetic algorithm for load optimization.

5. Model Definition

This is a simple and efficient method that uses the load factor of the physical machine and also VM constraints given by the user about the VM resource requirement. It also identifies the overloaded physical machine and selects the VM to migrate based on the past behavior of the VM and picks the appropriate PM based on its resource utilization rate. Then it discovers the under utilized PMs and migrates the VMs running on it to some other suitable PMs, and turn it off in view of energy saving. Since accurately forecasting the resource requirement and behavior of the VM is not possible our algorithm utilizes the user deployed resource details of workload of the VM and considers the load factor of the physical machine as well as physical machine cluster to identify the appropriate PM for the given VM request. We use bin packing heuristic combined with three different algorithms to minimize the number of Physical machines required to place a set of VMs, quick and correct placement of VMs , maintain balanced load among the servers, increase the resource utilization rate and importantly doing all these things without violating any SLA agreements.

N number of virtual machines with resource requirements VR (CPU, Memory, N/W Bandwidth) to be placed on a set of M physical machines with resource capacities of PR (CPU, Memory, N/W Bandwidth) grouped in K number of physical machine cluster.

Consider *PM* as a set of all the physical machines in the entire system, where PM = {PM_1, PM_2, PM_3 … PM_m}. m is total number of the physical machines and an individual physical machine can be denoted as *PM i*, where i denote the physical machine number and range of i is (1 <= i <= m). Similarly, the set of VMs on the physical machine i, can be {VMi1, VMi2.… VMin} here n is the number of VMs on the physical server i. If we want to deploy VM j on the PMi then the load of the CPU, RAM and bandwidth has to be calculated individually. The CPU load of the PMi at the time interval ts is denoted as follows

$$\mathrm{PMi(cpu, ts)} = \sum_{j=1}^{n} \mathrm{VMij(cpu, ts)} \quad (1)$$

The amount of RAM utilized by all the VMs of PMi at the time interval ts can be denoted as follows,

$$\mathrm{PMi(ram,\ ts)} = \sum_{j=1}^{n} \mathrm{VMij(ram,\ ts)} \quad (2)$$

The amount of Network Bandwidth utilized by all the VMs of PMi at the time interval ts can be denoted as follows

$$\mathrm{PMi(nbw,\ ts)} = \sum_{j=1}^{n} \mathrm{VMij(nbw,\ ts)} \quad (3)$$

Where PMi represents the i^{th} physical machine of the Physical Machine Cluster k, VMij represents j^{th} virtual machine of the PMi and cpu, ram and nbw denotes the

amount of CPU, RAM and Network Bandwidth utilized by all the VMs of the PMi respectively.

Hence derived from (1),(2) and (3) the weighted average load of the Physical Machine Cluster k at time interval ts can be denoted as follows

$$PMCk(WL, ts) = \sum_{i=1}^{m} PMi\,(WL, ts) \quad (4)$$

Where PMCk represents the k^{th} physical machine cluster of the datacenter, WL represents the weighted load of physical machine cluster at time interval ts and PMi represents the i^{th} physical machine of the Physical Machine Cluster k

At any time interval the total VM load of a PM should not exceed the host capacity

$$\sum PMi\ W_{resource\ usage}\ (ts) \leq TH\ value \leq \sum PMi\ W_{resource\ capacity} \quad (5)$$

Where resource € {CPU, RAM, Network Bandwidth} and $W_{resource}$ is the weight associated with each resource TH value is the threshold value set by the administrator if the load goes beyond this value the host can be considered as overloaded host and the selected VMs has to be migrated to other appropriate physical machines.

6. Algorithm Design for the Process of Virtual Machine Allocation

In this process the objective is to place the VMs in PMs in a way that the total number of PMs required to place all the VMs is decreased. So we considered this a multi potential bin packing problem since this is a NP-hard problem, we provide a heuristic based on multiple policy. In the earlier stages of allocation most of the PMs are under utilized or not used so our heuristics works as like the first fit scheduler which is a simplest one to implement and which increases the response time of VM placement. As the number of VM grows in the datacenter the utilization level of PM is also being considered by our heuristic which really helps in maintaining the balanced load among servers. Towards the closing stages the heuristic works according to the nature of the VMs workload that is gathered from the user provided hints which helps in avoiding the bottleneck of a particular resource as well as avoiding the violence of any SLA agreements. The algorithm which is used to achieve these things is given below.

Algorithm 1: Dynamic VM placement

Step1: The VM requests given by the user at the time ti is considered for allocation and scans the values of number of CPU cores, amount of RAM and amount of N/W bandwidth required.

Step2: In this algorithm the scheduler maintains an index table for physical clusters and physical machines as well as their states whether available or busy.

Step 3: The scheduler scans the index table of the physical cluster for the load below 50 %, from top until the first available physical cluster is found or the index table is scanned fully.

Step 4: If the physical cluster is found then scan the index table of physical machines for the load below 50 % in all three major resources, from the top until the first physical machine is found.

Step 5: When found return the ID of the physical machine to the main controller

Step 6: Assign the VM to the identified PM.

Step 7: Update the index table of the PM and Physical cluster.

Step 9: Go to the step 1

Step 8: If not found then scheduler scans the index table of the physical cluster for the load below 70 %, from top until the first available physical cluster is found or the index table is scanned fully.

Step 9: If the physical cluster is found scan the index table of the PMs based on the requirements of the requested VM.

Step 10: If the requested VM is a CPU intensive then scan the PM index table for the amount of CPU utilized is below 70 %, from the top until the first physical machine is found.

Step 11: When found return the ID of the physical machine to the main controller

Step 12: Assign the VM to the identified PM.

Step 13: Update the index table of the PM and Physical cluster and go to the step 1

Step 14: If the requested VM is a memory intensive then scan the PM index table for the amount of RAM utilized is below 70%, from the top until the first physical machine is found.

Step 15: When found return the ID of the physical machine to the main controller

Step 16: Assign the VM to the identified PM.

Step 17: Update the index table of the PM and Physical cluster and go to the step 1

Step 18: If the requested VM is a network intensive then scan the PM index table for the amount of network bandwidth utilized is below 70%, from the top until the first physical machine is found.

Step 19: When found return the ID of the physical machine to the main controller

Step 20: Assign the VM to the identified PM.

Step 21: Update the index table of the PM and Physical cluster and go to the step 1

Step 22: If Physical Cluster is not found. The scheduler scans the index table for the load below 80 %, from top until the first available physical cluster is found or the index table is scanned fully

Step 23: If found scan the index table of the PMs based on the requirement of the requested VM.

Step 24: If the requested VM is a CPU intensive then scan the PM index table for the least number of CPU cores utilized from the top until the first physical machine is found.

Step 25: If found check the host has enough CPU cores to fulfill the VMs CPU requirement and will not surpass 90% of load after placing the new VM, then return the ID of the physical machine to the main controller.

Step 26: Assign the VM to the identified PM.

Step 27: Update the index table of the PM and Physical cluster and go to the step 1.

Step 28: Else go to step 22

Step 29: If the requested VM is a memory intensive then scan the PM index table for the least amount of RAM utilized from the top until the first physical machine is found.

Step 30: If host has enough RAM to fulfill the VMs memory requirement and will not surpass 90% of load after placing the new VM, then return the ID of the physical machine to the main controller.

Step 31: Assign the VM to the identified PM.

Step 32: Update the index table of the PM and Physical cluster and go to the step 1.

Step 33: Else go to step 22

Step 34: If the requested VM is a network intensive then scan the PM index table for the least amount of network bandwidth utilized from the top until the first physical machine is found.

Step 35: If host has enough bandwidth to fulfill the VMs bandwidth requirement and will not surpass 90% of load after placing the new VM, then return the ID of the physical machine to the main controller.

Step 36: Assign the VM to the identified PM.

Step 37: Update the index table of the PM and Physical cluster and go to the step 1.

Step 38: Else go to step 22

7. Load Balancing among Physical Servers

Since virtual machine workloads frequently change eventually, the well primary placement choices is not sufficient to maintain the balanced load. So it is essential to dynamically rework placements to make QOS constraints are to be satisfied while change in the data center load. Maintaining balanced load among server requires more number of VM migrations which leads to increase the operational cost of the service provider so VMs should be rearranged in a way such that the number of VM migrations should be minimized while satisfying resource utilization and load balance. In this type of multifaceted problems, even the most prominent algorithms can't realize all the associations between VMs, physical servers, and physical clusters to lead the most finely optimized solution. In order to achieve this goal a new grouping based genetic algorithm is proposed and we believe that our new algorithm is useful for this kind of complex optimization problem.

7.1 Grouping Genetic based Algorithm Design for Load Balancing among Physical Servers

Genetic algorithm is a better searching technique for VMs mapping problem because of its enhanced optimization ability and parallelism advantages to solve complex problems.

The common steps of the Genetic algorithm are summarized as follows:

- Creation of an initial population
- The following steps are repeated until it reaches the stopping condition
- Select chromosome pairs for mating
- perform cross-over to generate new offspring's
- Calculate the fitness value of new offspring's
- Create a new population

7.2 Creation of an Initial Population

Genetic algorithm is executed in parallel on a set of selected physical servers. So creating Initial populations plays an important role[27] in genetic algorithm so we develop a novel algorithm to generate initial population. In solution space for these physical hosts Selection process chooses the solution vectors according to the probability which is proportional to the fitness value. Then the algorithm crosses the chosen product vectors and performs mutation operation on the crossed product vectors based on the fitness value. The algorithm continues the same stage until it reaches out the terminating situation, followed by the crossover and mutation process.

Steps for selecting initial Population

Step 1: Check the PM load against threshold value.

Step 2: If any PM resource utilization surpasses the threshold value that can be considered as an overloaded host

Step 3: Select the overloaded servers and sort those PMs based on their resource utilization value.

7.3 Fitness Function

The fitness value plays an important role in any individuals output. It is the evaluation methodology of the dominance of an individual in the population. The performance of an individual can be determined by its fitness value. The performance of an individual can be considered as better when the fitness value is high. The existence or termination of an individual is completely based on the fitness value. Therefore, the fitness function is an essential part of the Genetic Algorithm. The objective function can be defined as follows when there is m host in the physical cluster k and m is the number of VM in each host.

$$\mathrm{PMi}(Rcpu, ts) = PMi(Tcpu, ts) - \sum_{j=1}^{m} \mathrm{VMil}(Dcpu, ts) \quad (6)$$

Where PMi(Rcpu,ts) represents the remaining CPU of i^{th} PM at the time slot ts ,T cpu represents the total CPU capacity of i^{th} PM and VMij(Dcpu,ts) represents the demanded CPU of the j^{th} VM of the i^{th} Physical host at the time slot ts.

$$PMi(Rram, ts) = PMi(Tram, ts) - \sum_{j=1}^{m} VMij(Dram, ts) \quad (7)$$

Where PMi (Rram,ts) represents the remaining RAM of i^{th} PM at the time slot ts ,Tram represents the total RAM capacity of i^{th} PM and VMij (Dram,ts) represents the demanded RAM of the j^{th} VM of the i^{th} Physical host at the time slot ts.

$$PMi(Rnbw, ts) = PMi(Tnbw, ts) - \sum_{j=1}^{m} VMij(Dnbw, ts) \quad (8)$$

Where PMi (Rnbw,ts) represents the remaining Network Bandwidth of i^{th} PM at the time slot ts, Tnbw represents the total Network Bandwidth capacity of i^{th} PM and VMij (Dnbw,ts) represents the demanded Network Bandwidth of the j^{th} VM of the i^{th} Physical host at the time slot ts.

$$PMCk\ \mu Rcpu = \sum_{x=1}^{m} \frac{PMi\ Rcpu}{m} \quad (9)$$

$$PMCk\ \ \mu Rram = \sum_{x=1}^{m} \frac{PMi\ Rram}{m} \quad (10)$$

$$PMCk\ \mu Rnbw = \sum_{x=1}^{m} \frac{PMi\ Rnbw}{m} \quad (11)$$

Where PMCk μRcpu , PMCk μRram and PMCk μRnbw represents the k^{th} physical cluster's mean value of CPU, RAM and Network Bandwidth respectively.

In our proposed algorithm we consider four objectives in packing and optimizing the virtual machines in a data center: minimizing the total revenues, reducing the power consumption cost, reducing the cost of migration, increasing the total revenues and also reducing the SLA violation rate. These diverse objectives can be accomplished by evaluating the following fitness function described in equation 12 while allocating the VMs

$$\begin{aligned} \text{minimize } & \left(\sqrt{\frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{PMiR\ cpu} - PMCk\ \mu R\ cpu)^2} \right) \\ & + \left(\sqrt{\frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{PMi\ ram} - PMCk\ \mu R\ ram)^2} \right) \\ & + \left(\sqrt{\frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{PMiR\ nbw} - PMCk\ \mu R\ nbw)^2} \right) \end{aligned} \quad (12)$$

The objective function of our algorithm wants to minimize the standard deviation of the remaining CPU, RAM and Network Bandwidth in each host. As we consider that the load of the entire physical cluster instead of taking into consideration of the total number of virtual machines in each physical host as a load balance metric we

developed an objective function that tries to balance the consumption of CPU, RAM and Network Bandwidth on each host, in view of a heterogeneous environment, which consists of different hosts with different configurations.

7.4 Crossover Operator

Genetic algorithms crossover operator used to combine the qualities of different individuals in the population with the intention of creating a new generation. Hypothetically the new child will have good qualities from both parents and optimistically has better fitness. Any two parents have been chosen with probability relative to the fitness of the individual. Most of the times, the individuals with high fitness value will reproduce with higher probability than the individuals with lower fitness value, we followed a method which is similar to the one illustrated in[28] for the implementation process of the crossover operator. In our methodology all of the servers from both parents are integrated and the servers are sorted based on the fitness.

The servers with less remaining capacity of all the individual resources are at the front of the list, whereas the servers with more remaining capacity are placed at the end of the list. Then our algorithm analytically chooses the servers which has less remaining capacity and remains them together in the same group. During this process whenever a selected server contains any VM that belongs to a server that has been chosen previously, then that server is a superfluous and can be removed in order to avoid duplication. But this process will create a list of servers that may not include all VMs. These VMs which are outstanding that have not been integrated in any server will be used to reinserted in to other servers based on the algorithm 1.

7.5 Mutation Process

Mutation operator in our algorithm comprises three alternatives. First, choice of mutation process removes the VMs of randomly selected servers and the removed VMs consequently reinserted into the other servers which are in the new population based on our algorithm 1. Second, two randomly chosen VMs of existing packing order are interchanged between servers. In this process we assure that the algorithm never interchanges two VMs that came from the same server. As a third option, one VM is shifted to a different server to generate a new packing order.

Based on the information provided by the monitoring driver the second and third genetic operator works on the packing order list, to increase the performance of the ordering genetic process. Finally, for all the above genetic operators the mutation process is done on the VMs with probability inversely proportional to the fitness value of the server that the VMs originally come from. VMs placed in servers with lesser fitness value are mutated more frequently than VMs placed in servers with higher fitness value, in order to guarantee that the organization of enhanced server is retained. Presently new children will be an element of the next generation so we need to choose one solution from the next generation of solution. Whenever the exit criteria are satisfied then this algorithm is stopped and returns servers which has the highest fitness evaluation value.

8. Performance Evaluation

8.1 Experimental Setup

The presented algorithm is implemented in JAVA Net beans IDE. Then we use Cloud Sim simulator for simulation to assess the execution and performance of our heuristics with some of the existing scheduling algorithm in terms of Response Time, Load Balancing among servers, Reasonable Resource Utilization, energy consumption, Minimum number of active PMs and Higher profit by reducing the number of migrations. The performances of the proposed algorithm were examined from both users and service provider's perception.

Since it is difficult to access the real data centers or cloud infrastructures we used simulation based evaluation which can be easily reproducible to compare the performance of the proposed algorithm with the following existing works which is currently used by the majority of the cloud service providers: 1) First Fit Algorithm 2) Round Robin Scheduling Algorithm 3) Best Fit Algorithm. The simulated cloud environment contains a cluster of heterogeneous PMs the total resource capacity of PMs is expressed in percentage and randomly generated VM resource demand includes the number of CPU cores, amount of RAM and required network bandwidth.

8.2 Analysis

The investigations are done to analyze the effect our proposed algorithm in number of physical servers required to place a certain number of VMs, overall resource

utilization rate of all the active servers, allocation time, load balancing, percentage of migration and percentage of SLA violations. The simulation results show that our proposed algorithm can use the less number of physical servers for placing a certain number of VMs which helps to improve the resource utilization rate. The response time of our algorithm is little bit more than the first fit algorithm because of its nature of allocating VMs is based on the user constraints and past usage history of the VMs. Higher SLA satisfaction rate and lower load imbalance rate can be observed in results which also show that our multi dimensional host load aware and user constraints based algorithm is applicable, valuable and reliable for implementation in real virtualized environments.

Rebalancing of load in datacenter environment need live VM migrations but more number of frequently moved VMs between physical hosts causes increased datacenter cost hence the load rebalancing has to be achieved with minimum number of VM migrations in order to solve this issue we used a modified version of genetic algorithm for load optimization. Our results show that the percentage of VM migrations had been decreased through which we can achieve the better results for load balancing along with cost reduction.

In the following figures, Figure 2 shows the number of physical servers utilized by the scheduler to place the set of VM request without violating any SLA. Here our proposed host load aware user hint based algorithm and first fit algorithm uses comparatively same number of physical hosts for placing the set of VMs. The number of servers used by the proposed algorithm is minimized when compared to the round robin and best fit algorithm.

Though the numbers of servers used by the first fit and proposed algorithms are comparatively stable from Figure 3 we can see that the resource utilization rate of

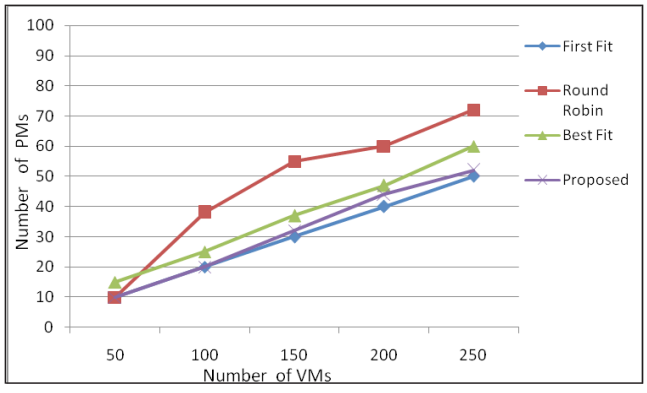

Figure 2. Comparision of the number of Physical Servers.

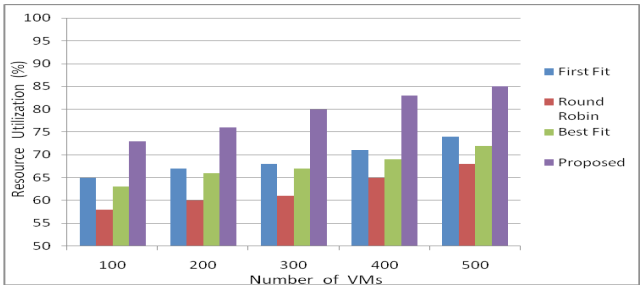

Figure 3. Comparision of the overall resource utilization rate.

our algorithm is appreciably outperforms the other three algorithms.

Figure 4 shows that the response time of all the algorithms are comparatively stable our algorithm takes little bit more time to allocate VMs than the first fit algorithm because of its nature of allocating VMs based on the user provided information and past usage history of the VMs

The analysis extremely examines the effect of load balancing by using the algorithm and the number of migration needed to achieve the load balanced environment subsequent to scheduling.

Figure 5 shows the percentage of load imbalance value in which our algorithm demonstrates that it gets better the

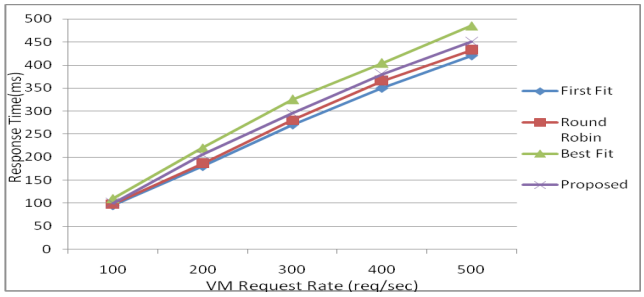

Figure 4. Comparision of the ResponseTime of different algorithms.

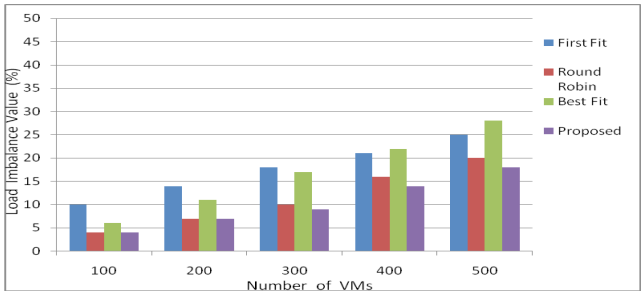

Figure 5. Comparision of the percentage of Load Imbalance Value.

way to obtain the load balancing of the data center than the three other approaches when the number of VMs to deploy is increased.

Our proposed algorithm is effective in improving the resource utilization rate and load balancing with the help of live migrations. But one of our major aims is increasing the total revenue which requires cutting down the VM migration cost which can be achieved by reducing the percentage of VM migration rate. We use migration rate as the estimation metric which is defined as the percentage of the migrated VMs to the total number of VM instances. We showed the results in the following Figure 6. The proposed algorithm decreases the migrating rate from about 18% - 20% to less than 13 % which leads to reduce the VM migration cost. Though the curve of our proposed algorithm indicates that only less number of VMs migrated from their original host to a new host we achieved the better resource utilization benefit and balanced load among the physical hosts.

From the below Figure 7 the low SLA violation rate is observed in the proposed algorithm because it uses the past behavior of the VM along with the user provided information and it maps the PM by considering the availability of the each key resource like CPU, RAM and network bandwidth individually.

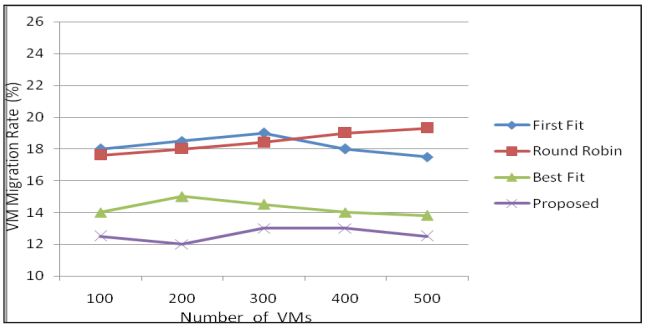

Figure 6. Comparision of the Percentage of VM Migarations for Load Balancing.

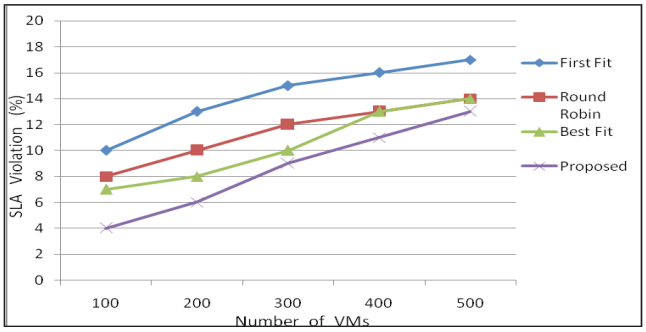

Figure 7. Comparision of the Percentage of VMs that violate their SLA.

9. Conclusion and Future Work

We presented a novel algorithm that considers user constraints of VM along with physical host load factor to address the problem of mapping the VMs into PMs such that the number physical host used is minimized, the over utilization and under utilization of the resources of a host can be identified and resolved at the same time without violating any SLA agreements. Since considered this as a multi potential bin packing problem, we combined three different heuristics which considers load factor of hosts along with user provided information at the various stages of placing the VMs in physical hosts. The proposed algorithm utilizes minimum number of physical servers for hosting the set of VMs, which also reduces the energy consumption of the datacenter and it achieved high resource utilization rate by the way of using minimal number of physical servers. Another considerable enhancement in our algorithm is less percentage of load imbalance value and the percentage of VMs that violate their SLA.

We plan to incorporate the proposed algorithm with an open source cloud platform and test its efficiency against real time environment in future. Also we would like to model the interconnection prerequisites that can correctly express the relationships between VMs consolidated in the same host which will be valuable for additional optimizations of VM scheduling in cloud infrastructure.

10. References

1. Chen K, Zheng WM. Cloud Computing: System instance and Current State. Journal of Software. 2009; 20(5):1337–48.
2. Xu ZW, Liao HM, et al., The Classification research of Network Computing System. Journal of Computing Machine. 2008; 18(9):1509–15.
3. Zhang GW, He R, Liu Y. The Evolution based on Cloud Model. Journal of Computing Machine. 2008; 7:1233–9.
4. Breitgand D, Epstein A. Sla-aware placement of multivirtual machine elastic services in compute clouds. '11 IFIP/ IEEE International Symposium on Integrated Network Management (IM); 2011. p. 161–8.
5. Meng X, Isci C, Kephart J, Zhang L, Bouillet E, Pendarakis D. Efficient resource provisioning in compute clouds via vm multiplexing. Proceeding of the 7th international conference on Autonomic computing; New York, NY, USA; 2010. p.11–20.

6. Hewlett Packard Web Site, Introducing Integrity Virtual Machines white paper. Available from: http://docs.hp.com/en/9987/Intro VM 2.1. 2011 Feb 26.
7. VMWareWeb Site, Server Consolidation, Containment Solutions Brief. Available from: http://www.vmware.com/pdf/server consolidation. 2011 Mar 15.
8. Bobroff N, Kochut A, Beaty K. Dynamic Placement of Virtual Machines for Managing SLA Violations. IM '07. 10th IFIP/IEEE International Symposium on Integrated Network Management; 2007 May 21. p. 119–28.
9. Sindelar M, Sitaraman RK, Shenoy P. Sharing-Aware Algorithms for Virtual Machine Colocation. Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures; San Jose, California, USA; 2011 Jun. p. 367–78.
10. Xu J, Fortes JAB. Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. Proceedings of the 2010 IEEE/ACM International Conference on Green Computing and Communications and International Conference on Cyber, Physical and Social Computing; Hangshou, PR of China; 2010 Dec. p. 179–88.
11. Singh A, Korupolu, M, Mohapatra D. Server-Storage Virtualization: Integration and Load Balancing in Data Centers. Proceedings of the 2008 ACM/IEEE conference on Super computing (SC'08); Austin, TX; 2008. p. 1–12.
12. Meng X, Pappas V, Zhang L. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. Proceedings of IEEE INFOCOM; San Diego, CA, USA; 2010 Mar. p. 1–9.
13. Kumar S, Talwar V, Kumar V, Ranganathan P, Schwan K. vManage Loosely Coupled Platform and Virtualization Management in Data Centers. Proceedings of the 6th international conference on Autonomic computing; ACM, Barcelona, Spain; 2009 Jun. p. 127–36.
14. Wood T, Shenoy P, Venkataramani A, Yousif M. Black-box and Gray-box Strategies for Virtual Machine Migration. Proc of the 4th USENIX Symposium on Networked Systems Design and Implementation; Cambridge, MA; 2007. p. 229–42.
15. Bobrof N, Kochut A, Beaty K. Dynamic Placement of Virtual Machines for Managing SLA Violations. Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management; Munich, Germany; May 2007 May. p. 119–28.
16. Tian W, Zhao Y, Zhong Y, Xu M, Jing C. A dynamic and integrated load-balancing scheduling algorithm for Cloud data centers. Proceedings International Conference on Cloud Computing and Intelligence Systems (CCIS); Beijing: IEEE; 2011. p. 311–5.
17. Zheng H, Zhou L, Wu J. Design and implementation of load balancing in web server cluster system. Journal of Nanjing University of Aeronautics & Astronautics. 2006 Jun; 38(3):347–51.
18. Singh A, Korupolu M, Mohapatra D. Server-storage virtualization: Integration and load balancing in data centers. Proceedings of the 2008 ACM/IEEE Conference on Super computing; 2008. p. 1–12
19. Arzuaga E, Kaeli DR. Quantifying load imbalance on virtualized enterprise servers. Proceedings of WOSP/SIPEW'10 San Jose, California, USA; 2010 Jan 28-30. p. 235–42.
20. Chen M, Zhang H, Su YY, Wang X, Jiang G, Yoshihira K. Effective VM sizing in virtualized data centers. Integrated Network Management (IM'11); 2011. p. 594–601.
21. Thiruvenkadam T, Karthikeyani V. An approach to virtual machine placement problem in a datacenter environment based on overloaded resource. International Journal of Computer Science and Mobile Computing. 2014 Jun; 3(6):837–42.
22. Thiruvenkadam T , Karthikeyani V. A Comparative study of VM Placement Algorithms in Cloud Computing Environment. International Journal of Advanced Computational Engineering and Networking. 2015 Jan; 3(1):18–21.
23. Thiruvenkadam T , Karthikeyani V. Multi Dimensional Host Load Aware and User Constraints Based Algorithm for Scheduling Virtual Machines. International Journal of Advancements in Computing Technology (IJACT). 2015 Jan; 7(1):56–66.
24. Guo G, Ting-Iei H, Shuai G. Genetic Simulated Annealing Algorithm for Task Scheduling based on Cloud Computing Environment. IEEE International Conference on Intelligent Computing and Integrated Systems (ICISS); Guilin; 2010. p. 60–3.
25. Wilcox D, McNabb A, Seppi K. Solving virtual machine packing with a reordering grouping genetic algorithm. Proceedings of the IEEE Congress on Evolutionary Computation (CEC); 2011. p. 362–9.
26. Goyal T, Agrawal A. Host scheduling algorithm using genetic algorithm in cloud computing environment. International Journal of Research in Engineering & Technology (IJRET). 2013 Jun; 1(1):7–12.
27. Srinivas M, Patnaik LM. Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. IEEE Transactions on Systems, Man and Cybernetics. 1994 Apr; 24(4):656–67.
28. Rohlfshagen P, Bullinaria J. A Genetic Algorithm with Exon Shuffling Crossover for Hard Bin Packing Problems. Proceedings of Genetic and Evolutionary Computation Conference; 2007. p.1365–71.