I.J. Modern Education and Computer Science, **2015, 3, 39-46**
Published Online March 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2015.03.06

An Efficient Virtual Machine Scheduling Technique in Cloud Computing Environment

Vijaypal S. Rathor
MANIT/Computer Science Engineering, Bhopal, 464051, India
Vijay.palrathor@gmail.com

R. K. Pateriya and Rajeev K. Gupta
MANIT/Computer Science Engineering, Bhopal, 464051, India
pateriyark@gmail.com, rajeevmanit1276@gmail.com

Abstract—Cloud is a collection of heterogeneous resources and requirements of these resources can change dynamically. Cloud providers are always interested in maximizing the resources utilization and the associated revenues, by trimming down energy consumption and operational expenses, while on the other hand cloud users are interested in minimizing response time and optimizing overall application throughput. In cloud environment to allocate the resources with minimum overhead time along with efficient utilization of available resources is very challenging task. The resources in cloud datacenter are allocated using a virtual machine (VM) scheduling technique. So there is a need of an efficient VM scheduling technique to maximize system performance and cost saving. In this paper two dynamic virtual machine scheduling techniques i.e. Best fit and Worst fit are proposed for reducing the response time along with efficient and balanced resource utilization. The proposed algorithms removes the limitations of the previously proposed Novel Vector based algorithm and minimizes the response time complexity in order of $O(\log n)$ and $O(1)$ using Best Fit and Worst Fit strategies respectively.

Index Terms—Cloud computing, load balancing, VM scheduling, response time, resource leak.

I. Introdution

The cloud computing technology is comparatively close to the present, acquiring primacy in 2007 as a means of depicting Internet-based distributed computing and its allied applications [1], So different professionals defined it in different way, Rajkumar Buyya defined it as follows: Cloud is a combination of parallel and distributed computing, which is consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and released [2]. Cloud Computing is a service delivery computing rather than product because in cloud computing the resources are stored on the remote location and these resources accessed via network as a services on the basis of pay as you go. Cloud computing is also called utility computing because user can access these services as storage and computing from anywhere without worrying about where these services are hosted [3].

Cloud computing framework involves service delivery model, deployment model, characteristics, infrastructure and resources as shown in figure 1. In the cloud environment an user can access three kinds of services i.e. Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [4], all these three services are described as a service delivery model of cloud computing. Where SaaS means applications are accessed as a service by client running on the cloud computing infrastructure hosted by the service providers. PaaS means platform or high–level integrated environment is accessed as service to design, build, run, test, deploy and update the applications created by client using development language and tool say Java, python, .net etc. provided by the service providers to the cloud infrastructure. The providers provide the processing power, storage, network and other basic computing resources as a IaaS, with which users can deploy and run any software including operating systems and applications.

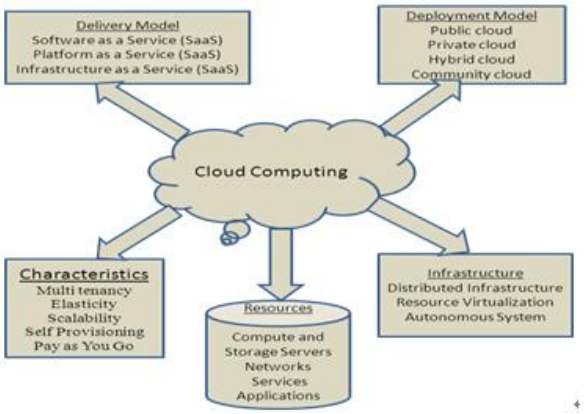

Fig. 1. Cloud Computing Frame work

The cloud computing services are deployed as private-owned by single organization and only available to that organization, public- run by third parties and available to all or publicly and hybrid-combination of private and public, known as deployment model [4].

Virtualization is the key feature of cloud computing [4]. It is a technique which allows multiple OS can run simultaneously on a single PM. This is implemented through the hypervisor. Virtualization coupled with migration ability make possible to consolidate the physical servers that gives the benefits in terms of reliability efficiency and cost and also improves system security. It also provides the effective resource management and load balancing through VM scheduling techniques. In the cloud environment number of user can request for the services simultaneously, so there is a need of scheduling mechanism that efficiently allocates the resources to the user. Two types of VM scheduling techniques are available in the cloud: static and dynamic. Static VM scheduling is based on the prior knowledge of the system, whereas dynamic scheduling depends on the current state of the system. User requirement is dynamic in nature, so the dynamic VM scheduling is better than the static VM scheduling but it has lot of over head. In the virtual machine placement problem consists of two steps. In the first step resource requirement of the VM is calculated and in the second step search the appropriate host, where VM can be placed. In this paper approach used for selecting the host for placing the VM is similar to the method proposed by the M. Mishra et al. [12], and the main focus is to minimizing searching time for the destination host.

A dynamic VM scheduling technique is proposed, which minimizes the resource allocation time of user request, provide efficient utilization of resources, load balancing and server consolidation. The experimental results show that the proposed technique provides efficient utilization of resources and equally distribute the load on the whole nodes in the cloud data center.

The work in this paper presents efficient approach for the following:

- **VM placement technique:** It provides the VM allocation strategy to provision the VM to PM.
- **Minimize the response time:** The user's request assigned to the appropriate host with almost no latency.
- **Efficient utilization of resources:** The resources are allocated in a manner that there should not be resource leak or nonuniform resource utilization. For a PM, if some type of resource is exhausted while other types of resource are still surplus, the surplus resource is wasted because the PM is already fully loaded, this is called resource leak [8].
- **Load balancing:** The resources are allocated in a manner that load on the PMs is equally distributed.
- **Server consolidation:** The VMs are placed in a manner that required number of PMs are minimized.

The rest of this paper is organized as follows. Section II describes related work of VM scheduling and load balancing. The section III and IV describe formulation of proposed Imbalance Level and Load measurement. The structure and the design of proposed model are introduced in Section V. Section VI presents experimental result and evaluation. Section VII gives conclusion and future work.

II. Related Work

The various virtual machine scheduling and migration methods have been proposed in the literature for minimizing the resource allocation time, efficient utilization of resources, load balancing and server consolidation. Time required to place a VM to the host is known as resource allocation time. K. Yang et al. [5] proposed a migration technique using classical time series model [6] prediction method for load balancing. W. Tian et al. [7] introduce a dynamic and integrated resource scheduling algorithm (DAIRS) for cloud datacenters which develops integrated measurement for the total imbalance level of a cloud datacenter and average imbalance level of each physical machine. X. Li et al. [8] proposed a balanced algorithm to reduce the energy consumption by providing balance resource utilization and avoiding resource leak. R. Buyya et al. [9] discusses energy aware resource allocation algorithm based on First Fit Decreasing technique (FFD) given in [10] to provide the server consolidation by giving the concept of upper and lower threshold. The fixed value for threshold is unsuitable for dynamic and unpredictable workload environment, so to solve this problem an adaptive threshold-based approach is proposed in [11]. M. Mishra et al. [12] introduces a VM placement technique based on the vector arithmetic given in [14] for load balancing and server consolidation.

III. Imbalance Level and Resource Leak Measurement

Xin Li et al. [8] gave the quantitative definition of resource leak. If there exists such R that $(\mu_R > \theta_\Lambda)$ and $(|max\{\mu_R\} - min\{\mu_R\}| \geq \theta_\Delta)$, where μ_R is resource utilization of PM, θ_Δ and θ_Λ are the two threshold to determine the moment when resource leak occurs. Here it is assumed that all the PM are of same capacity while in practical cloud environment the PM can be of different capacities so in this paper resource leak or imbalance level is defined in another way.

Let CPU, Memory and Bandwidth resources capacity of a physical machine are represented as C, M and B respectively. Assume that C_{max}, M_{max}, and B_{max} are the maximum available resource capacity of a PM and C_{rem}, M_{rem}, and B_{rem}, are the remaining or available capacity of each resource. The percentage remaining or available capacity of each resource is calculated as follows

$$C = \frac{\mathrm{C_{rem}}}{\mathrm{C_{max}}} * 100 \quad (1)$$

$$M = \frac{\mathrm{M_{rem}}}{\mathrm{M_{max}}} * 100 \quad (2)$$

$$B = \frac{\mathrm{B_{rem}}}{\mathrm{B_{max}}} * 100 \quad (3)$$

$$Avg = \frac{C+M+B}{3} \quad (4)$$

$$IB_{host} = \frac{1}{3}\left\{\frac{|C-Avg|}{Avg} + \frac{|M-Avg|}{Avg} + \frac{|B-Avg|}{Avg}\right\} \quad (5)$$

C, M, and B are the percentage remaining resource capacity of a host; Avg is the average remaining capacity of the host and IB_{host} is the imbalance level of the host.

IV. Load Measurement

In cloud computing the load balancing is very important task because in cloud environment number of physical machines are used to serve the users requests simultaneously. Due to improper scheduling, a situation may occur where some of the nodes are heavily loaded while other nodes are idle or doing very little work which reduces system performance. So load among the various physical machines must be equally distributed in other words, at any instant of time every node should do approximately the equal amount of work. The figure 2 describes the unbalance datacenter without appropriate scheduling and balance datacenter with appropriate scheduling.

The load can be the CPU load, memory capacity or network load. On the basis of these load different load measurements approaches are proposed in literature.

Wood et al. [13] proposed a load measurement as follows

$$V = \frac{1}{(1-CPU_u)(1-MEM_u)(1-NET_u)} \quad (6)$$

Where CPU_u, MEM_u and NET_u are the average CPU, memory and network utilization of a physical machine.

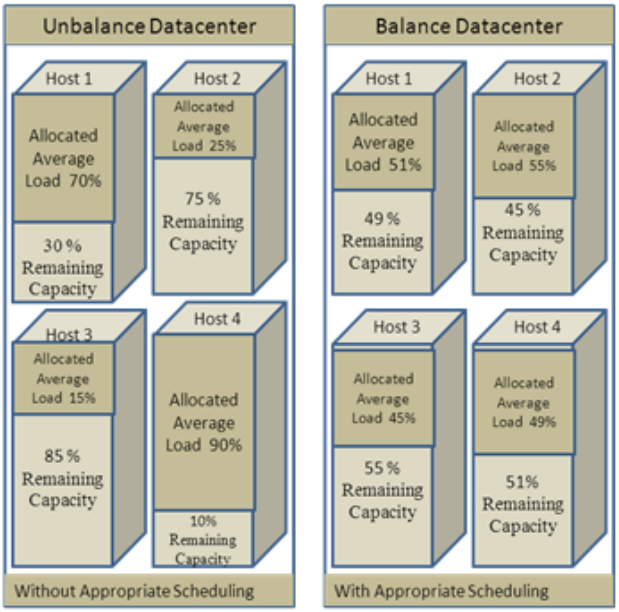

Fig. 2. Load Distribution in Cloud Datacenter

Wenhong Tian et al. [7] measured load of a single server i using integrated load imbalance value, which is defined as follows

$$\frac{(Avg_i-CPU_u^A)^2+(Avg_i-MEM_u^A)^2+(Avg_i-NET_u^A)^2}{3} \quad (7)$$

Where CPU_u^A is an averaged CPU utilization of all CPUs in cloud datacenter which is defined as given below:-

$$CPU_u^A = \frac{\sum_i^N CPU_i^U CPU_i^n}{\sum_i^N CPU_i^n} \quad (8)$$

And Avg_i is defined as follows

$$Avg_i = (CPU_i^U + MEM_i^U + NET_i^U)/3 \quad (9)$$

In previously proposed methods [13], [7] etc. load on PM is calculated on the basis of utilization, but in cloud data center the PMs can be of different capacity, a high capacity PM provides the more computing power than the less capacity PM. So load on the high capacity PM will be more than the less capacity PM. So here load of a physical machine is calculated in terms of average remaining capacity or available capacity, which shows the load balancing, means two PM are balance in load if the average remaining capacity or available capacity is same on both the PM. To measure load of a physical machine in terms of average remaining capacity of resources is calculated as given below.

$$Host_{cap} = \frac{K_1*C_{rem}+K_2*M_{rem}+K_3*B_{rem}}{3} \quad (10)$$

$$\text{where} \quad \sum_i^3 K_i = 1 \quad (11)$$

Here $Host_{cap}$ is the average remaining capacity of a PM, K_1, K_2 and K_3 are the constant, and C_{rem}, M_{rem} and B_{rem} are the remaining or available capacity or each resources of the PM.

V. Proposed Work

In this section a novel approach is proposed for VM placement which effectively solves the problems of minimizing the response time, load balancing, and balance resource utilization and server consolidation in cloud data center. When the cloud accept VM request with resource requirements R_{req}, a new VM will be created on a PM in real time. There are various policies to select the PM to host the new VM. In this paper two algorithms are introduced i.e. Best Fit and Worst Fit, based on two different policies. The best fit minimizes response time in order of $O(log\, n)$ where as worst fit technique minimizes it in order of $O(1)$.

In the Worst Fit load on PMs are equally loaded, but number of PMs required to host the VMs in worst fit can be more as compared to the best fit if all physical machines have equal remaining resource capacity, so it is tradeoff between the selections of these two.

A. Hosts Categorization in Cloud Datacenter

In these algorithms the hosts are classified according to their resource availability. If there are m types of resources, hosts can be divided into m! lists according to permutation. For example if three types of resources are there CPU (C), Memory (M) and Bandwidth (B), hosts are divided into 3! i.e. six lists in which each list contains particular type of host using the permutation and so on. Figure 3 shows the six resources availability lists based on three types of resources i.e. CPU, MEM, and BW.

For example, in first list have hosts which have resource availability in an order as $C \geq M \geq B$, second have $C \geq B \geq M$, third have $M \geq C \geq B$, fourth have $M \geq B \geq C$, fifth have $B \geq C \geq M$ and sixth have $B \geq M \geq C$. Here $C \geq M \geq B$ means, all hosts belong to this list, have CPU availability is greater than or equal to Memory availability and memory availability is greater than or equal to bandwidth. This list having resource capacity in order $C \geq M \geq B$ is complementary to the list having resource capacity in order $B \geq M \geq C$ and so on. Now these lists are sorted according to proposed best fit or worst fit algorithm and then VM is provisioned. The Algorithm 1 shows list creation process in datacenter.

Algorithm 1: Datacenter Creation Algorithm

Number of Host = N

If Type of resources =3 (CPU, MEM and BW) then Number of $\boldsymbol{Host\_List} = \mathbf{6}$.

[1] Create 6 empty $\boldsymbol{Host\_List}$ according to type of resources
[2] for $\boldsymbol{i} = \mathbf{1}$ to N
[3] Create the Host[i] and initialize it
[4] Add the Host[i] into the $\boldsymbol{Host\_List}$ as
[5] if Host[i] resources have $\boldsymbol{CPU \geq MEM \geq BW}$ then add it into the $\boldsymbol{Host\_List}$ which have resources capacity as $\boldsymbol{CPU \geq MEM \geq BW}$.
[6] And so on.
[7] End if
[8] End for
[9] Sort all $\boldsymbol{Host\_lists}$ in ascending order (for Best Fit) or in descending (for Worst Fit) order of resources availability.

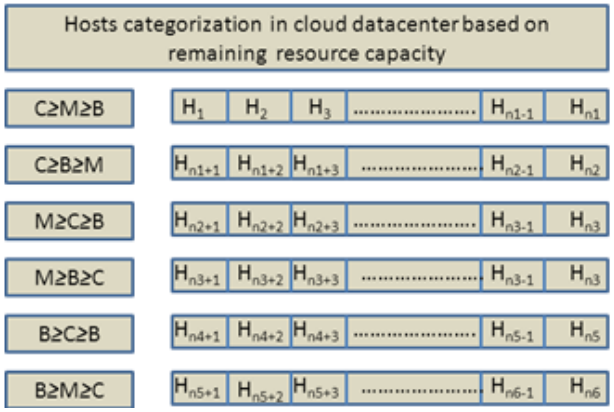

Fig. 3. Host List Creation in Datacenter

B. Working Flow Graph of Proposed Model

The figure 4 shows the working flow diagram of the proposed algorithm. Let us assume that when request ($C \geq M \geq B$) of VM (newly created or already running) for VM placement arrives from the VM pool then VM scheduler place this VM on the satisfying host in the list in which all host have remaining resource capacity in order as $C \geq M \geq B$. In other words if a VM having CPU requirement greater than or equal to Memory requirement and Memory requirement greater than or equal to Bandwidth is placed on the satisfying host in the list, in which all host have remaining resource capacity of CPU greater than or equal to Memory and Memory greater than or equal to Bandwidth.

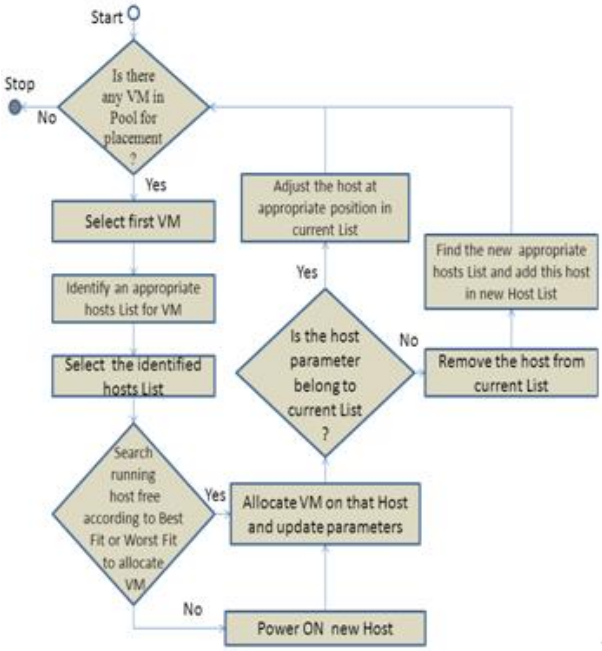

Fig. 4. Flow graph of proposed model

The satisfying host in the selected list is searched according to the two different strategy based on the Best Fit or Worst Fit algorithms. If satisfying running host is not available to allocate the VM, power ON the new host and allocate VM on that host and update the host parameters. If the host remaining capacity of each resource type does not belong to the current list, remove the host from the current list, find the new appropriate list, add the host at appropriate place in the new list.

C. Best Fit Strategy

In the Best Fit algorithm hosts in each list in datacenter are sorted in ascending order according to remaining capacity of resources. When request of a new VM or already running VM for VM placement arrives at the cloud data center, VM scheduler find the appropriate list and apply the binary search on the selected list to find host that is the best fit in remaining resource capacity than the VM requirement capacity in all dimensions. If no such host is available, Power ON the new physical machine and assign the VM on that PM. The Algorithm 2 shows VM allocation using the Best Fit strategy.

Algorithm 2: Best Fit Allocation Algorithm

Input: $\boldsymbol{n}$ number of Hosts, $\boldsymbol{6}$ number of $\boldsymbol{Host\_List}$, one $\boldsymbol{VM\_List}$ of $\boldsymbol{m}$ number of VMs.

[1] for each $\boldsymbol{VM}[\boldsymbol{i}]$ in $\boldsymbol{VM\_List}$
[2] Find an appropriate $\boldsymbol{Host\_List}$ as $\boldsymbol{Chosen\_Host\_List}$
[3] $\boldsymbol{Chosen\_Host} = \boldsymbol{Binary\_Search}(\boldsymbol{Chosen\_Host\_List}, \boldsymbol{VM}[\boldsymbol{i}], 0, \boldsymbol{sizeof}\ (\boldsymbol{Host\_List}) - \boldsymbol{1})$;
[4] Allocate VM on the Chosen_ Host
[5] Update following parameters.
a.) $\boldsymbol{Chosen\_Host\_CPU} = \boldsymbol{Chosen\_Host\_CPU} - \boldsymbol{VM\_CPU}$
b.) $\boldsymbol{Chosen\_Host\_MEM} = \boldsymbol{Chosen\_Host\_MEM} - \boldsymbol{VM\_MEM}$
c.) $\boldsymbol{Chosen\_Host\_BW} = \boldsymbol{Chosen\_Host\_BW} - \boldsymbol{VM\_BW}$
[6] if $\boldsymbol{Chosen\_Host}$ parameter does not belong to the current $\boldsymbol{Host\_List}$ then.
[7] Remove the $\boldsymbol{Chosen}_{\boldsymbol{host}}$ from the Current $\boldsymbol{Host\_List}$ and find the new $\boldsymbol{Host\_List}$ to which $\boldsymbol{Chosen\_Host}$ belong as a $\boldsymbol{Chosen\_Host\_List}$.
[8] Add the $\boldsymbol{Chosen\_Host}$ at appropriate position in the new $\boldsymbol{Chosen\_Host\_List}$
[9] else Adjust the $\boldsymbol{Chosen\_Host}$ in current $\boldsymbol{Chosen\_Host\_List}$ at appropriate position.
[10] end if
[11] end for

The Best Fit allocation algorithm calls the binary search procedure for searching the best fit satisfying host. The binary search procedure return the best fit host in remaining capacity for VM requirement capacity to the Best Fit algorithm as $Chosen\_Host$. Finding best fit host for VM reduces the required number of physical machine by fully utilization of the resources. The binary search strategy for searching the best fit host will reduce the allocation time of one VM in order of $O(logn)$. So response time complexity will be $O(logn)$.

The worst case time complexity of Best Fit algorithm for m number of VMs will be $O(m\ logn)$, where n is the number physical machines. The procedure for binary search is shown in Algorithm 3. Binary search procedure takes , a sorted host list, a VM, starting index and the end index of a sorted host list as input in which host will be searched and return the $Chosen\_Host$.

D. Worst Fit Strategy

In case of Worst Fit algorithm the hosts of each list in datacenter are sorted in descending order of remaining capacity of resources. When request (new VM or already Running VM) for VM placement arrive at the cloud data center, VM scheduler find the appropriate list and place this VM on to the first satisfying host of selected list. The Algorithm 4 shows VM allocation using the Worst Fit algorithm. If first host does not satisfy the resource requirement in all dimensions, power ON the new PM, allocate the VM on the new PM.

The update procedure is same for both the algorithms

Algorithm 3: Binary Search Algorithm

[1] Procedure Host $\boldsymbol{Binary\_Search}$(Host_List, $\boldsymbol{VM}[\boldsymbol{i}]$, low, high)
[2] if $(\boldsymbol{low} > \boldsymbol{high})$ then
[3] Power ON a new Host and return new Host.
[4] end if
[5] $\boldsymbol{mid} = (\boldsymbol{low} + \boldsymbol{high})/\boldsymbol{2}$;
[6] if$(\boldsymbol{VM}[\boldsymbol{i}]\boldsymbol{_CPU} \leq \boldsymbol{Host_List}[\boldsymbol{mid}]\boldsymbol{_CPU}$&& $\boldsymbol{VM}[\boldsymbol{i}]\boldsymbol{\_MEM} \leq \boldsymbol{Host\_List}[\boldsymbol{mid}]\boldsymbol{\_MEM}$&& $\boldsymbol{VM}[\boldsymbol{i}]\boldsymbol{\_BW} \leq \boldsymbol{Host\_List}[\boldsymbol{mid}]\boldsymbol{\_BW}$) then
[7] if$(\boldsymbol{low} == \boldsymbol{high})$ then
[8] return $\boldsymbol{Host\_List}[\boldsymbol{mid}]$;
[9] else return procedure $\boldsymbol{Binary\_Search}(\boldsymbol{Host\_List}, \boldsymbol{VM}[\boldsymbol{i}]$, low, mid);
[10] end if
[11] else return procedure $\boldsymbol{Binary\_Search}(\boldsymbol{Host\_List}, \boldsymbol{VM}[\boldsymbol{i}], \boldsymbol{mid} + \boldsymbol{1}, \boldsymbol{high})$;
[12] end if
[13] end procedure

Algorithm 4: Worst Fit Allocation Algorithm

Input: $\boldsymbol{n}$ number of Hosts, 6 number of $\boldsymbol{Host\_List}$, One $\boldsymbol{VM\_List}$ of *m* number of VMs.

[1] for each $\boldsymbol{VM}[\boldsymbol{i}]$ in $\boldsymbol{VM\_List}$
[2] Find an appropriate $\boldsymbol{Host\_List}$ as $\boldsymbol{Chosen\_Host\_List}$.
[3] if first host of $\boldsymbol{Chosen\_Host\_List}$ satisfy the VM[i] requirement then
[4] Select the first host of $\boldsymbol{Chosen\_Host\_List}$ as a $\boldsymbol{Chosen\_Host}$.
[5] else Power ON new Host as a $\boldsymbol{Chosen\_Host}$.
[6] end if
[7] Allocate VM on the Chosen_ Host.
[8] Update the following parameters.
a.) $\boldsymbol{Chosen\_Host\_CPU} = \boldsymbol{Chosen\_Host\_CPU} - \boldsymbol{VM\_CPU}$
b.) $\boldsymbol{Chosen\_Host\_MEM} = \boldsymbol{Chosen\_Host\_MEM} - \boldsymbol{VM\_MEM}$
c.) $\boldsymbol{Chosen\_Host\_BW} = \boldsymbol{Chosen\_Host\_BW} - \boldsymbol{VM\_BW}$
[9] if $\boldsymbol{Chosen\_Host}$ parameter does not belong to the current $\boldsymbol{Chosen\_Host\_List}$ then
[10] Remove the $\boldsymbol{Chosen}_{\boldsymbol{host}}$ from the Current Host_List and find the new $\boldsymbol{Host\_List}$ to which $\boldsymbol{Chosen\_Host}$ belong as a $\boldsymbol{Chosen\_Host\_List}$.
[11] Add the $\boldsymbol{Chosen_Hos}$t at appropriate position in the new $\boldsymbol{Chosen\_Host\_List}$.
[12] else Adjust the $\boldsymbol{Chosen\_Host}$ in current $\boldsymbol{Chosen\_Host\_List}$ at appropriate position.
[13] end if
[14] end for

which are given above in the flow graph working of the proposed model.

According to the algorithm 4 the worst case time complexity of West Fit algorithm for m number of VMs will be $O(m\ logn)$, where n is the number of physical

machines, but according to $Step3$ response time (time required to search the destination host) complexity for a VM will be constant i.e. $O(1)$.

VI. Evaluation And Simulation Results

The proposed algorithms are simulated using the CloudSim [15] [16] toolkit; it is a simulating environment for cloud computing applications. All results are collected using an Intel core i3 Windows PC with 1.9 GHz CPU, 4GB memory. Simulated results of Best Fit and Worst Fit algorithms are compared with two other algorithms (Greedy First Fit and Balance algorithm) which is included above. In CloudSim toolkit the scheduling policies can be applied at the two levels (Host level and VM level) [16].

The proposed scheduling policy is applied on the host level in cloud computing using the VmScheduler. So VMs are provisioned on to the physical machine to perform the operations required by the user. In the CloudSim simulator, at host level, VmScheduler uses two policies namely TimeShared and SpaceShared for resource allocation host level.

The figure 5 shows the evaluation structure of CloudSim toolkit for proposed work.

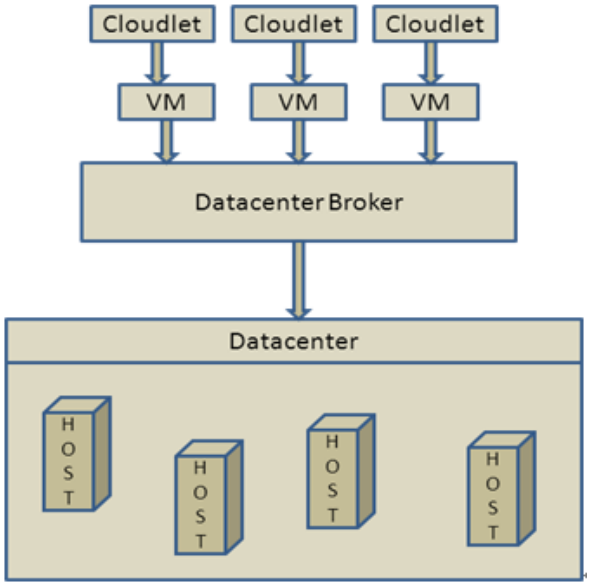

Fig. 5. Evaluation Structure for proposed work

The CloudletScheduler also divides resources among the Cloudlets running on virtual machines using two policies namely TimeShared and SpaceShared. As the proposed algorithm works on two different allocation strategies given above using SpaceShared policy for both VmScheduler and CloudletScheduler.

A. Experimental Results

The results shown in graphs are simulated by running 38 different size VMs on the 38 physical machines of different capacities. Experimental results show that the Worst Fit Strategy outperform the Best Fit strategy for all evaluated parameters in the case of physical machines of different capacities. In addition to this the proposed strategies provide good results in comparison to other traditional approaches such as Balance [8] and Greedy [17] for all evaluated parameters.

The experimental results for three evaluated parameters (Response Time, Imbalance level of physical machines and Load distribution on physical machines) are shown below in graphs.

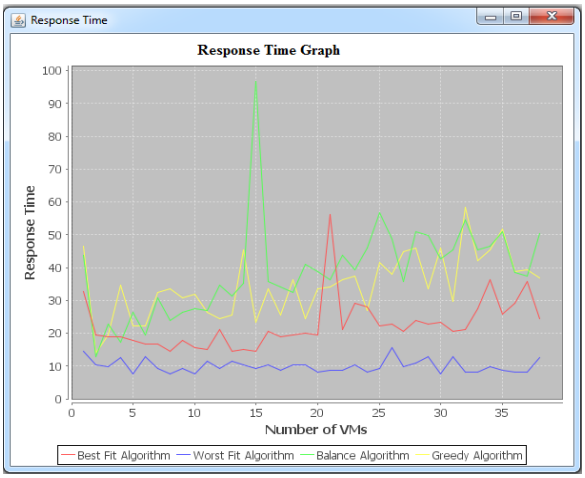

Fig. 6. Simulation Result for Response Time

The figure 6 shows the response time in millisecond of various algorithms. In the figure the Worst Fit algorithm takes very less time to allocate VMs on the physical machines in comparison with all other algorithms. Best Fit takes less time from the other two algorithms.

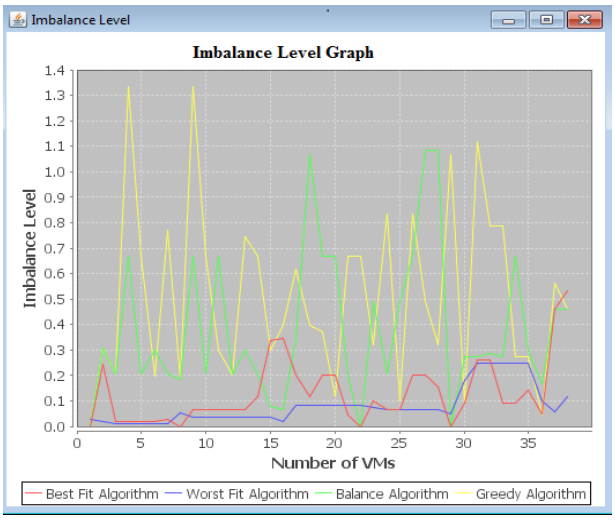

Fig. 7. Simulation Result for Imbalance Level

The figure 7 shows the imbalance level of various algorithms. The increase in imbalance level means the increase in resource leak. In the case of Worst Fit and Best Fit algorithms the PMs are utilizing resources in balance manner in comparison with other algorithms, but Worst fit outperform the Best Fit.

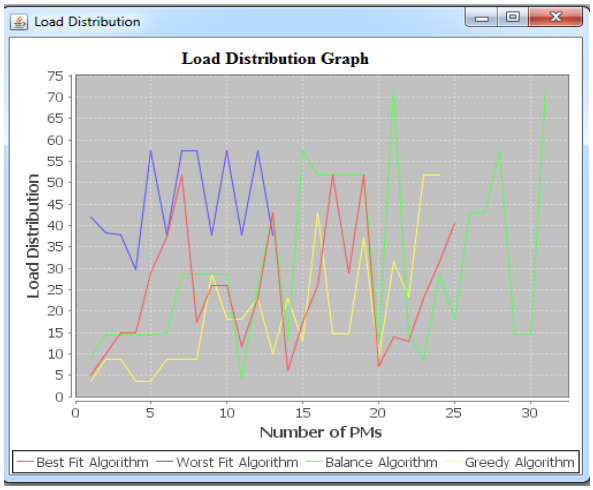

Fig. 8. Simulation Result for Load Distribution on PMs.

The figure 8 shows the load distribution (in terms of average remaining capacity of a physical machine) on physical machines of various algorithms. The load distribution shows the average remaining capacity of a physical machine, which is defined earlier. The graph presents that in the case of Worst Fit algorithm the load on all physical machines are equally distributed, and in Worst Fit strategy, to allocate the 35 VMs it required less PMs in comparison to other algorithms. So in case Worst Fit unutilized PMs can be turn OFF, which can provide the server consolidation.

B. Contributions of proposed work

The proposed algorithm removes the flaws present in Novel Vector Based approach [12] and gives a new approach to minimize the response time of user request.

Novel Vector Based approach has following flaws which are removed in proposed methods.

- In the novel approach VM is placed on the most complementary PM, complementary means a PM which has resource imbalance of same magnitude as VM in opposite direction. So there can be more than one PM with same magnitude which are complementary to the VM requirement. For example VM has the requirement vector $(2i + 4j + 6k)$, PM1 has the utilization vector $(9i + 6j + 3k)$ and PM2 has utilization vector $(12i + 8j + 4k)$. Both PM are best complementary to the VM, but PM2 will be good choice to place the VM that can reduce the required number of PM. This problem is not arises in the proposed algorithms.
- The novel approach is based on the $3-D$ vector. In which the projection of $3-D$ cube on the plane results in Planer Resource Hexagon (PRH). Opposite triangles of PRH are complementary to each other. This method is inappropriate in more than three resource dimensions scenarios. Whereas, proposed methodology used the concept of lists. These lists are created according to the permutation. For example if resources are of m dimensional, m! Lists are created.
- In novel approach the planer hexagon contain the PMs which have unequal capacity resource utilization vector. Virtual machine can have equal resources requirement in two or more dimensions (i.e. $C = M < I$ or $C = I < M$ or $I = M < C$ or I=M=C etc.). For example a virtual machine which has a resources requirement $CPU = MEM > IO$ should be placed on the best complementary PM which has resource utilization as $IO > MEM = CPU$. So this method is not appropriate in these scenarios. This problem is not arises in the proposed algorithms.

VII. Conclusion And Future Work

This paper describe the importance of efficient VM scheduling technique to provide the solution for the problems of VM placement, response time, balance resource utilization and load balancing. The proposed Best Fit and Worst Fit techniques are beneficial for cloud provider as well as cloud user for cost saving . The experimental results show that the Worst Fit technique is better than the Best Fit technique for different capacities hosts. But both techniques provide better improvements with other traditional techniques.

As proposed algorithms are evaluated using the CloudSim simulator toolkit. So run time challenges can be resolved by implementing the proposed algorithms in real world cloud environment. In the proposed scheduling algorithm VM migration strategy is not considered for under utilize host or over utilize host. The evaluation for the response time, balance utilization of resources of a host and load distribution on all the hosts is only based on the VM scheduling techniques. So using the migration strategy for under utilize host and over utilize host, which can provides the measurable improvements for load balancing and can also provides the server consolidation.

Acknowledgments

The Success of this research work would have been uncertain without the help and guidance of a dedicated group of people in our institute MANIT Bhopal. We would like to express our true and sincere acknowledgements as the appreciation for their contributions, encouragement and support. The researchers also wish to express gratitude and warmest appreciation to people, who, in any way have contributed and inspired the researchers.

References

[1] R. Buyya, J. Broberg, A. Goscinski, “Cloud Computing: Principle and Paradigms”, 1st ed., Hoboken: John Wiley & Sons, 2011.

[2] Michael Miller, “Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online”, 1st ed., USA: Que Publishing, 2008.

[3] Weiss. “Computing in the Clouds”, netWorker, 11(4): 16-25, ACM Press, New York, USA, Dec. 2007.

[4] T. Mather, S. Kumaraswamy, and S. Latif, “Cloud Security and Privacy”, 1st ed., USA: O’Reilly Media, 2009, pp. 11-25.

[5] K. Yang, J. Gu,T. Zhao and G. Sun , "An Optimized Control Strategy for Load Balancing based on Live Migration of Virtual Machine ", in Proc. 6th Annual China grid Conference (ChinaGrid), Liaoning: IEEE, 2011.
[6] W. Gersch and T. Brotherton, "AR model prediction of time series with trends and seasonalities: A contrast with Box-Jenkins modeling", in Proc. 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes, 1980, 19(1): 988-990.
[7] W. Tian, Y. Zhao, Y. Zhong, M. Xu and C. Jing, "A dynamic and integrated load-balancing scheduling algorithm for Cloud datacenters", in Proc. International Conference on Cloud Computing and Intelligence Systems (CCIS), Beijing : IEEE, 2011.
[8] X. Li, Z. Qian, R. Chi, B. Zhang, and S. Lu, "Balancing Resource Utilization for Continuous Virtual Machine Requests in Clouds", in Proc. Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Palermo: IEEE, 2012.
[9] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges", in proceedings International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, USA, July 12-15, 2010.
[10] M. Yue. "A simple proof of the inequality FFD (L)< 11/9 OPT (L)+ 1,for all l for the FFD bin-packing algorithm". Acta Mathematicae Applicatae Sinica (English Series), 7(4):321331, 1991.
[11] Beloglazov and R. Buyya, "Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers", in Proc. 8th International Workshop on Middleware for Grids, Clouds and e-Science, Ney York: ACM, 2010.
[12] M. Mishra and A. Sahoo, "On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach", in Proc. International Conference on Cloud Computing (CLOUD), Washington: IEEE, 2011.
[13] T.WOOD, P. Shenoy and A. Venkataramani, "Black-box and Gray-box Strategies for Virtual Machine Migration", in proceedings 4th USENIX conference on Networked systems design & implementation (NSDI), Berkeley: ACM, 2007.
[14] H. ZHENG, L. ZHOU, J. WU, "Design and Implementation of Load Balancing in Web Server Cluster System", Journal of Nanjing University of Aeronautics & Astronautics, Vol. 38 No. 3 Jun. 2006.
[15] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar AF De Rose, and Rajkumar Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software: Practice and Experience, 41(1):23{50, 2011.
[16] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, and Rajkumar Buyya, "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications", in: Proceedings 24th International Conference on Advanced Information Networking and Applications (AI NA), 2010.
[17] Subramanian S, Nitish Krishna G, Kiran Kumar M, Sreesh P4and G R Karpagam, "An Adaptive Algorithm For Dynamic Priority Based Virtual Machine Scheduling In Cloud", International Journal of Computer Science Issues (IJCSI), Vol. 9, Issue 6, No 2, November 2012.

Authors' Profiles

Vijaypal S. Rathor has completed his M.Tech. in Computer Science & Engineering from Maulana Azad National Institute of Technology (MANIT), Bhopal, India. He has completed his Bachelor of Engineering in Information Technology from SATI, Vidisha affiliated to RGPV, Bhopal India. He has over 2 years of teaching experience at his credit. His research interest includes Cloud Computing, Big Data and Security.

Dr. R. K. Pateriya is an Associate Professor in the Department of Computer Science and Engineering at Maulana Azad National Institute of Technology, Bhopal, India. He is a member of the IEEE. His current research interests include Cloud and Distributed Computing, E-Commerce, Security etc. He has authored and published over 100 research Papers.

Rajeev K. Gupta is a PhD candidate in the Department of Computer Science and Engineering at Maulana Azad National Institute of Technology, Bhopal, India. He has completed his Bachelor degrees in CSE at SATI, Vidisha and Master degree in CSE at MANIT, Bhopal. His research area is Cloud Computing, Distributed computing and Grid computing.