Load Management Model for Cloud Computing Using Cloudsim

Priyansh Srivastava, Bhavesh Gohil, and Dhiren Patel

***Abstract*—Cloud computing is an upcoming trend in the field of distributed computing in recent years. In cloud, resources are provided as a service in the form of virtual machines to its clients based on demand, which may result in overutilization or underutilization of servers. Cloud must accommodate changing demands for different types of processing with heterogeneous workloads and time constraints. This can be achieved by making decisions of Virtual Machine (VM) allocation and migration request time at the run time. We have evaluated the Load Management model for cloud computing using CloudSim simulator and compared with the default existing VM-Allocation algorithm. The results depict that the model reduces the number of overutilized hosts significantly and also an improved task execution time, thus increasing the performance overall.**

***Index Terms*—Dynamic load balancing, virtual machines (VM), VM distribution, cloudsim.**

I. Introduction and Background

Cloud computing offers resource pooling, on demand computing, scalability, flexibility, pay-as-per-use, high availability and low capital infrastructure setup and management [1], [2]. As Cloud computing is growing rapidly and users are demanding more services, efficient cloud management has become an interesting and important research area. Cloud resources are required to be managed at runtime to ensure fulfillment of user level agreements, fault tolerance, efficient utilization of resources and power saving. Accommodating varying user demands and runtime management of workload necessitates the use of a dynamic load management system for performance improvement and reduction in management costs [3].

There is a need of a cloud load management model to manage cloud resources, fulfill user level agreements, fault tolerance, efficient resource utilization, power saving, accommodating varying user demands, performance improvement and to reduce management costs [4].

VM provisioning policy by default follows either a round robin approach or a greedy approach in public cloud infrastructures, both of which can either overload the resources or do not utilize the resources to its full capacity. Also, the load balancing algorithms should consider power saving and balancing based on historical data so that future requests can be managed properly.

Considering the drawbacks and strengths of existing approaches of load balancers and VM schedulers, load management model for cloud computing addresses the complete model of load prediction, VM assignments and dynamic load balancing considering power saving, future requests managements and minimizing response time [5].

The rest of the paper is organized as follows. Section II describes our proposed load management model [5]. It provides the logical view of the system and discusses the algorithms used in the approach. Section III presents the results of simulation of the model. Conclusions followed by references are at the end in Section IV and Section V respectively.

II. Proposed Model

The proposed model considers a cloud environment having heterogeneous physical machines of different capacities and VM requests of heterogeneous type of varying sizes.

The different modules in the proposed model are as follows [5]:

- Load collection module
- Load evaluation module
- Load prediction module
- VM distribution module
- Dynamic load balancing module
  a. VM selection
  b. Destination selection
- VM migration module

The Load Collection module runs at regular intervals updating the current load of all machines. On receiving the load metrics at regular interval; the load evaluation, load prediction and load balancing module run on central controller to ensure stable performance of the entire cloud. The load migration module runs based on the decision of dynamic load balancing module. The VM distribution module runs each time a new request for VMs is received (to decide the destination).

A. *Load Collection Module*

The load collection module runs on all live physical machines and sends the load of the machine defined by the CPU usage, RAM usage and network bandwidth utilizations and available free space details to the central controller along with the usage metrics of the VMs running on it at regular intervals. All the metrics sent are averaged over a predefined interval so as to nullify any effect of transient spikes in load.

Manuscript received January 12, 2016; revised March 15, 2017.

Priyansh Srivastava was with S.V. National Institute of Technology, Surat, Gujarat, India. He is now with the Novell, India (e-mail: ps31ps@gmail.com).

Bhavesh Gohil and Dhiren Patel are with the S.V. National Institute of Technology, Surat, Gujarat, India (e-mail: bng@ coed.svnit.ac.in, dhiren@ coed.svnit.ac.in).

B. Load Evaluation Module

It takes the decision, if a physical machine should be considered as overloaded or not. All three metrics sent by Load Collection Module viz; CPU utilization, RAM usage and Network utilization are considered and a score is formulated based on all the 3 metrics to evaluate a system against a threshold score. Wood *et al*. gives the score as the volume of load to be [6].

$$\text{Volume} = \frac{1}{1-CPU} * \frac{1}{1\text{-net}} * \frac{1}{1-\text{mem}} \quad (1)$$

where CPU(CPU), Network(net) and Memory(mem) are the utilizations on the physical machine in percentage.

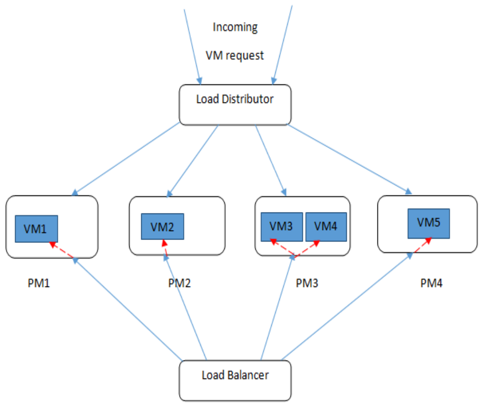

Fig. 1. Architecture of the proposed model [5].

C. Load Prediction Module

Load prediction is used to make a clever decision of VM placement by considering the current capacity of the physical machine, the expected load in the near future and the expected resource needs of the VM. It uses the seasonal exponential smoothing method because of its simplicity and good results [7]. *Single exponential smoothing can be expressed by:*

$$F_t = a * X_t + (1 - a) * F_{t-1} \quad (2)$$

where F_t is the predicted value at time t, F_{t-1} is the predicted value at time *(t-1)* and X_t is the current value. A weight, a, is taken to balance the current data and historical data [5]. The above *time series method of load prediction* is applied on the load metrics received by the Load collection module. The model describes a 3 element tuple which is used in the decision making for VM placement and Load Balancing and is updated at regular intervals when new metrics are received.

Virtual machine usage (VM) tuple- <C,M,N> where [5] -

C = max (predicted CPU usage, current CPU usage)

M = max (predicted memory usage, current memory usage)

N = max (predicted network usage, current network usage)

Physical machine free capacity (PM) tuple <C,M,N> where [5] -

C = min (predicted CPU free capacity, current CPU free capacity)

M = min (predicted memory free capacity, current memory free capacity)

N = min (predicted network free capacity, current network free capacity)

The VM usage tuple and PM free capacity tuple will

D. VM Distribution Module

This module runs a *VM Distribution Algorithm* as proposed in Load management model for cloud computing when an incoming VM request is received [5]. It uses *Best fit* to place VMs which is power efficient and can handle future VM request more efficiently. The algorithm takes into consideration all the 3 metrics which attribute to the VM tuple and the PM Tuple from Load Prediction Module and calculates a *fit score* to calculate the probability of a destination being selected.

$$Fit \quad Score = \frac{w1 * M + w2 * N}{w3 * (C - n)} \quad (3)$$

where n is the number of CPU requested in the new VM, and $w1$, $w2$ and $w3$ are the weights given to the three parameters to decide the fit value [5].

The algorithm first generates a candidate list which is a list of PMs (physical machines) which can accommodate the incoming VM request. Then, based on the score of fit value and the resulting probability distribution, it assigns a PM to the VM.

E. Dynamic Load Balancing Module

Dynamic Load Balancing module (also referred as Dynamic Load Balancer) then does the job of bringing stability to the system by migrating VMs from the overloaded PMs to other PMs [5]. The proposed Dynamic Load Balancer runs in two steps

1) VM selection

VM selection depends on two factors, first one is the active memory of the VM as it decides the migration time, the network bandwidth used in migration and CPU cycles used for migration. Second one is the amount of load the VM is putting on the PM. Based on the percentage CPU, memory and network utilization of the VM, Equation 1 of volume is used to check the load volume of a VM.

Considering both the above factors, of load volume and memory image, Wood *et al.* defines the VSR (Volume by size ratio) for VMs as [8]

$$VSR_{\text{score}} = \frac{\text{Volume}}{\text{MemorySize}} \quad (4)$$

Here, *VSR* denotes the load transferred per unit of memory. This is maximized to transfer maximum load from an overloaded machine and have minimum migration time in trade off [5].

2) Destination selection

The appropriate destination is selected based on the current and predicted usage of the PMs, the available capacity and best fit. Since migration incurs overhead on source and destination and the network, the algorithm tries to attain stability with minimum number of migrations. The first iteration tries to attain stability by migrating a single VM to a suitable destination and if still the system exists in an overloaded state, the algorithm proceeds to check for removal of multiple VMs. To limit number of migrations, only two VMs can be migrated from a single PM. The algorithm first generates a list of overloaded PMs and then forms a *move list* for all the overloaded PMs which consists of either one or two pairs of migrating VM and destination PMs for those VMs. In case, no destination PM is capable of holding the migrating VMs, a new PM is awakened. If for the overloaded PMs, the migrating VMs and the corresponding destinations are identified, the algorithm updates the free metric tuples of the source and destination PMs. After all the iterations if the system comes to a stable state with the calculated sources and destinations, the algorithm proceeds on placing the VMs to the destination PMs.

F. Migration Module

The migration module operates on the list of the VMs with the sources and destinations, executing migrations while minimizing the overall migration time and overhead. This is optimized by having an informed sequence of migration [9] and choosing the migration technique according to workload. Live migration has two major techniques- Precopy and Postcopy [10].

In case of more than one VMs migrating from a single host, the VM which is running workload which has a high rate of dirty pages should be migrated first as it causes large page faults in the host. These VMs tend to use a large amount of network bandwidth and hence will slow down the migration process for other VMs migrating before it [5].

III. Results

We used CloudSim simulator to simulate the load management model for cloud computing [5] and generate results. We performed the simulation: first with the existing allocation algorithm which allocates the incoming VM to the first host which has the required CPUs; and second with our proposed VM allocation algorithm and Dynamic Load Balancer. In both the cases, we have set the same parameters to perform the simulation:

- Number of Datacenters: 1
- Number of Hosts: 8 (2 hosts with 1 CPU, 4 hosts with 2 CPU, 2 hosts with 3 CPU). Each host has 2GB RAM, 1TB storage and 10000 kbps bandwidth.
- Number of VM`s: 10. Each VM has 512MB RAM, 1 CPU and 1000 kbps bandwidth.
- Number of Cloudlets: 50

The first result which we gathered was the number of hosts which were overutilised during the period of simulation. Here, Overutilization is measured collectively in terms of CPU, RAM and Network utilization of the host and not just the CPU utilization. The simulation involved execution of cloudlets which were given dynamically to the VMs running on hosts. Fig. 2 shows the number of hosts overutilised in our proposed model and Fig. 3 shows the same for the default model. We can see more peaks and more flat top edges in Fig. 3 which tells that more hosts were overutilised for a long time which affects the performance of the system. In Fig. 2, these are less as the load balancer decides on the basis of previous and predicted future usage of RAM, CPU and Network of the VM and migrates the VM to the best suitable host thus reducing host overutilization.

The second test we ran was to compare the execution time of the tasks (cloudlets). The cloudlets were submitted dynamically to the VMs and the time taken by all the cloudlets to complete was measured and average execution time of cloudlet was measured. Each cloudlet was of different size. We submitted a batch of 100, 250, 500, 750, 1000, 1250 cloudlets and calculated the average execution time. Fig. 4 shows the plot of Execution time of each batch of cloudlets in both the models. The average cloudlet execution time is visibly more in the default model than our proposed model due to the presence of an intelligent VM placement policy along with the dynamic load balancer.

To check and compare completion of no. of cloudlets (jobs) per time per hosts, bunch of cloudlets were submitted dynamically to the cloud and measured the no of completed cloudlets over regular interval with different no of hosts. Fig. 5 shows the plot of job throughput per hosts for both the models. The job throughput per host is visibly more in our proposed model than default one due to the presence of an intelligent VM placement policy along with the dynamic load balancer.

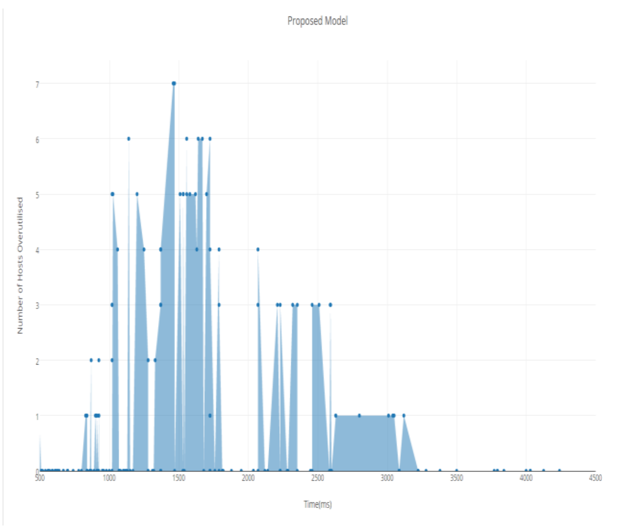

Fig. 2. Number of hosts overutilised during the simulation in our proposed model.

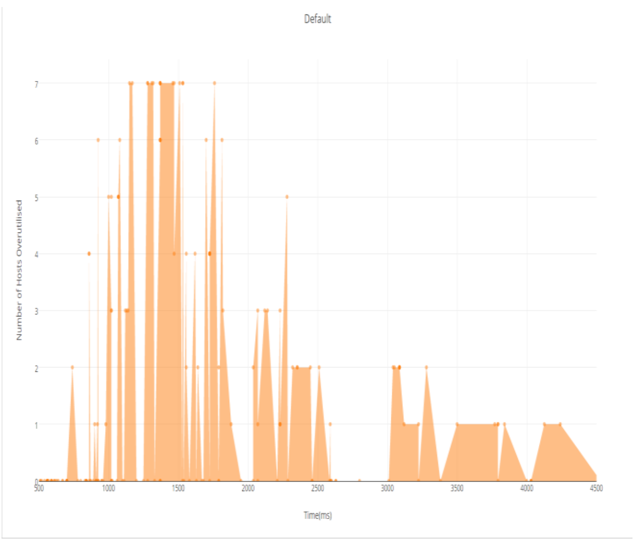

Fig. 3. Number of hosts overutilised during the simulation in the default model.

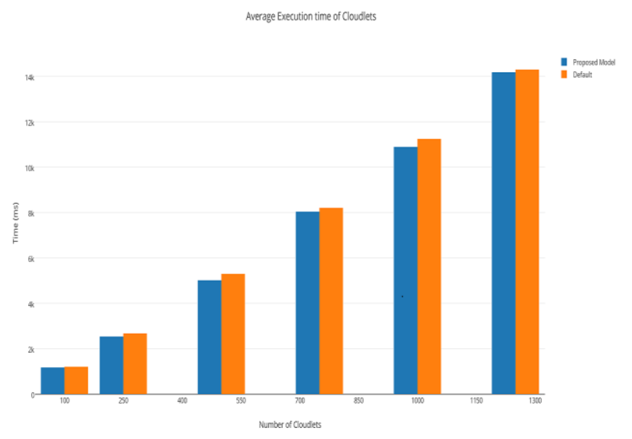

Fig. 4. Execution time of cloudlet batches in both the models.

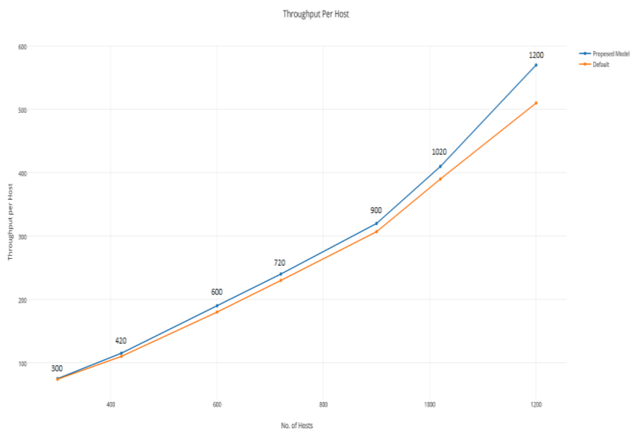

Fig. 5. Job throughput per hosts.

IV. Conclusions

Cloud systems need to manage load, and accommodate varying demands to ensure consistent performance along with efficient resource management. Simulations of our proposed framework indicate that it has relative good performance in makespan (average execution time of cloudlets/tasks), overutilization of hosts and job throughput per hosts.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7-18, Apr. 2010.

[2] A. T. Velte, T. J. Velte, and R. Elsenpeter, "Cloud computing: A practical approach," Tata Mcgraw-Hill Edition, 2010.

[3] A. M. Alakeel, "A guide to dynamic load balancing in distributed computer systems," *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 6, June 2010.

[4] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7-18, Apr. 2010.

[5] S. Mour, P. Srivastava, P. Patel, H. Ram, B. Gohil, and D. Patel, "Load management model for cloud computing," in *Proc. the 9th International Conference for Internet Technology and Secured Transactions(ICITST-2014)*, 2014.

[6] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. the 4th Usenix Conference on Networked Systems Design & Implementation*, 2007.

[7] B. Radojevic and M. Zagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments," in *Proc. 34th International Convention on MIPRO*, 2014.

[8] V. Sundaram, T. Wood, and P. J. Shenoy, "efficient data migration in self-managing storage systems," in *Proc. the 3rd International Conference on Autonomic Computing*, June 2006.

[9] X. Li, Q. He, J. Chen, K. Ye, and T. Yin, "Informed live migration strategies of virtual machines for cluster load balancing," in *Proc. the 8th Ifip International Conference on Network and Parallel Computing, Springer-Verlag, Npc'11,* 2011.

[10] K. Li, H. Zheng, and J. Wu, "Migration-based virtual machine placement in cloud systems," in *Proc. the 2nd IEEE International Conference on Cloud Networking (CloudNet),* November 2013.

Bhavesh Gohil is working as an assistant professor in the Department of Computer Engineering at S. V. National Institute of Technology (SVNIT), Surat. He carries more than 8 years of experience in Academic, Research and Industry. He is currently pursuing his Ph.D from the same institute.

His research interests include cloud computing and its issues and information security.

Priyansh Srivastava studied bachelor of technology in computer engineering from S. V. National Institute of Technology (SVNIT), Surat, India.

He is currently employed in Novell as a software engineer. His interests include mobile cloud computing and cloud security.

Dhiren Patel is a professor of Computer Engineering Department at S.V. National Institute of Technology, Surat. He carries more than 20 years of experience in Academics, Research Development of Secure ICT Infrastructure Design.

His research interests cover security and encryption systems, cloud computing and IoT, identity and access management, e-voting, advanced computer architecture etc. Besides numerous journal and conference articles, Prof. Dhiren has authored a book "Information Security: Theory Practice" published by Prentice Hall of India (PHI) in 2008. He is actively involved in Indo-UK, Indo-Norway, and Indo-Japan security research collaborations.