Asian Journal of Information Technology 15 (14): 2306-2312, 2016
ISSN: 1682-3915

Optimization of Virtual Machine Placement

A.R. Ragavi Priyadharshini, K. Koushika and P. Prakash
Department of Computer Science and Engineering, Amrita Vishwa Vidyapeetham,
641 112 Coimbatore, India

Abstract: Cloud computing is an internet based computing, where shared resources and information are provided to computers or other devices on demand. It has now become a highly demanded service or utility due to the advantages of high computing power, cheap cost of services, high performance, scalability, accessibility as well as availability. Cloud computing of course has some pitfalls which need to be given proper attention to make cloud computing services more reliable and user friendly. Some of the major drawbacks are underestimating cloud sprawl, failing to monitor performance, perils of platform lock-in mismanaged performance guarantees, data jurisdiction, data storage area, possible down time, inflexibility, etc. In this study, we have studied about the power consumption so as to improve the efficiency of a computing system. Here, we detect the requests and then the placement of virtual machine takes place, thereby improving the performance of the computing system.

Key words: Cloud computing, power, performance, VM request and VM placement, information

INTRODUCTION

Computer performance is characterized by the amount of useful work accomplished by a computer system or computer network compared to the time and resources used. Accompanied by improving the performance of a computing system, its power efficiency is also addressed as a major concern. Though performance of a computing system is enhanced per watt, the total current drawn by the computing systems has not come down. For an instance, the cost of power consumed will be relatively more than that of the hardware cost which is a major distress to the entrepreneurs. This scenario is again worse quality if in clusters or data centre like large scale systems.

The so called power consumption depends on various factors like, hardware efficiency, infrastructure, resource management, nature and type of applications running in the system. Also if the number of computing components increases, its cooling mechanism takes a lot of effort. Along with this cooler systems, power distribution infrastructures must be cared a lot which includes, UPS and power delivery units (Barroso, 2005).

Certain servers are hard to cool down because, there will be densely seated components preventing the space to let air flow between them. This puts the servers in an excited state and its performance might come down. Yet there are servers to aid, like blade servers which can do higher computations with components which demand less space. Though the space complexity seems reduced, the power demand is still a problem, wanting >4000 Watt for the entire system.

Literature survey: So from 1992, energy efficient green computing methods were found. This credit goes to the energy star, a program which was launched by US Environmental Protection Agency. Energy star was mainly created to reduce the greenhouse gas emissions and also to find and promote energy efficient products. One such technique discussed (Flinn and Satyanarayanan, 2004) is that it deployed the 'sleep' mode in electronic devices. As we know, this had a great welcome and almost all electronic gadgets come with this 'sleep' mode feature which is a state that does not consume more power.

Then, came the term 'green-computing' to denote power-efficient methods of computing. As years flew away, the computing need increased. The resource required increased which directly impacted on the power consumption. So about, after a decade, a newer version of it was introduced (Buyya *et al*., 2009) to make sure optimal performance is done which consumed less power, even in a very big tired system.

Apart from energy star, many companies took initiatives to develop the standardized methods and techniques which help in reduced power consumption, also to limit the carbon emission in the atmosphere.

Corresponding Author: A.R. Ragavi Priyadharshini, Department of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, 641 112 Coimbatore, India

Some of them include, the green grid, ECO_2 cloud, ECO_4 cloud and green electronics council, green computing impact organization, Inc., FIT4Green, climate savers computing initiative and international professional practise partnership. All these initiatives as mentioned (Beloglazov, 2013; Feller *et al*., 2012) were deployed with correspondence with membership from large companies like DELL, Microsoft, Intel, IBM, Sun Microsystems, HP, etc.

It is necessary to manage the resources under each cluster to match the demand in order to maximize total returns by minimizing the power consumption cost. Researchers of the study (Prakash *et al*., 2014) have minimized by applying minimal virtual design, live migration and variable resource management. But, the traditional way of scheduling doesn't meet their expected requirements. So, they introduced the distributive power migration and management algorithm for cloud environment that uses the resources in an effective and efficient manner ensuring animal use of power. The result indicates that the algorithm reduces up to 30% of the power consumption to execute services.

The power controller gets the server details and sends the identified server to the PIC microcontroller. The PIC microcontroller shuts up or down the server by passing the respective signal to the specific server. By this method, every incoming virtual machine is allocated a server by an arbitrary order defined by the user. Thus, a server cannot be left alone unless its maximum capacity is reached and no server can be left idle or without running at maximum potency. In algorithm (Prakash *et al*., 2014), it was implemented using a 12 v power supply, a controller section into which the input is fed from a USB and the output section. The dynamic power migration technique was highly satisfactory but could always be improved further with zigbee, bluetooth, etc.

MATERIALS AND METHODS

The state of the art of cloud computing: The process of optimizing the virtual machines starts with understanding the scenario. The current network will be checked for any under-load or over-load. If any, the corresponding requests will be sent to the respective components so that they will be handled and acted upon as required.

Handling an under-load request will be as follows: the local manger will study the scenario and finds if there are any host under-loaded. This will be found with the help of the under-load detection algorithm.

If an under-load is detected, the local manager sends an under-load request to the global manager, stating the name of the under-loaded host. The global manager calls the API and gets the list of VM allocated to the under-loaded host.

On getting the VM list, global manager triggers the VM placement algorithm with the following details, list of VM, their resource usage and state of the host taken from the database as arguments. Then, everything works according to the algorithm and the global manager will ensure if the migration is fully done.

Once the allocated hosts and the under-loaded host are popped out and migrated, the host becomes idle. So, the global manager puts this node into sleep mode. The host over-load request will be handled as follows.

The local manger will study the scenario and it finds if there are any host over-loaded. This will be found with the help of the over-load detection algorithm.

If any over-load is detected, the local manager sends an over-load request to the global manager, stating the name of the over-loaded host, along with the list of UUIDs of the virtual machines.

Which virtual machine should be popped out so that, the load can be balanced will be decided by the configured VM selection algorithm. On receiving the global manager triggers the configuration VM placement algorithm, for which the argument list will be as follows, list of VMs, other system information.

On the event off putting the popped-out virtual machines, if they are re-mapped to currently idle servers, then those servers will be activated by the Wake-on-LAN technology as addressed (Jung *et al*., 2010). In both under-load and over-load request the requests will be served at a single resource which are accessed by the 'put' method of the Hypertext Transfer Protocol (HTTP) (Fig. 1 and 2).

Host under load detection: The algorithm stated below states that: select the newly CPU utilization measurement and with the help of this measurement find and calculate the mean of x.

Make a comparison with the threshold value which you have already set as static with the measurement which you have calculated from mean time which is addressed (Cleveland and Loader, 1996).

If the outcome of this is very low when compared to the threshold value then the CPU utilization is underperformed (Algorithm A).

Algorithm A; underperformed:

```
Input: threshold, x, consumption
Output: Whether the node is underperformed
  If Consumption Is Not Emptythen
        Consumption?last_x values of_consumption
            meanConsumption?sum(utilization)/len(utilization)
        Return (meanConsumption≤threshold)
    Return false
```

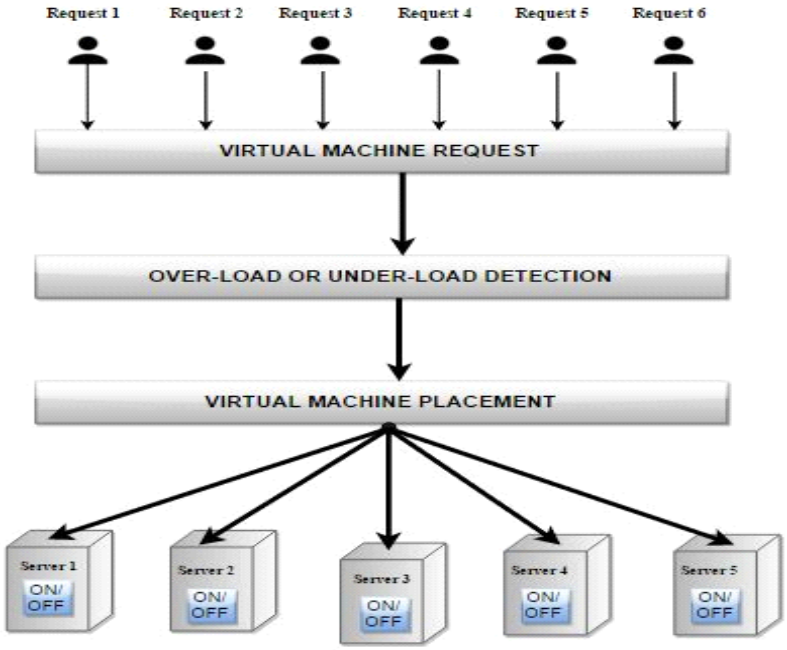

Fig. 1: Process of optimization

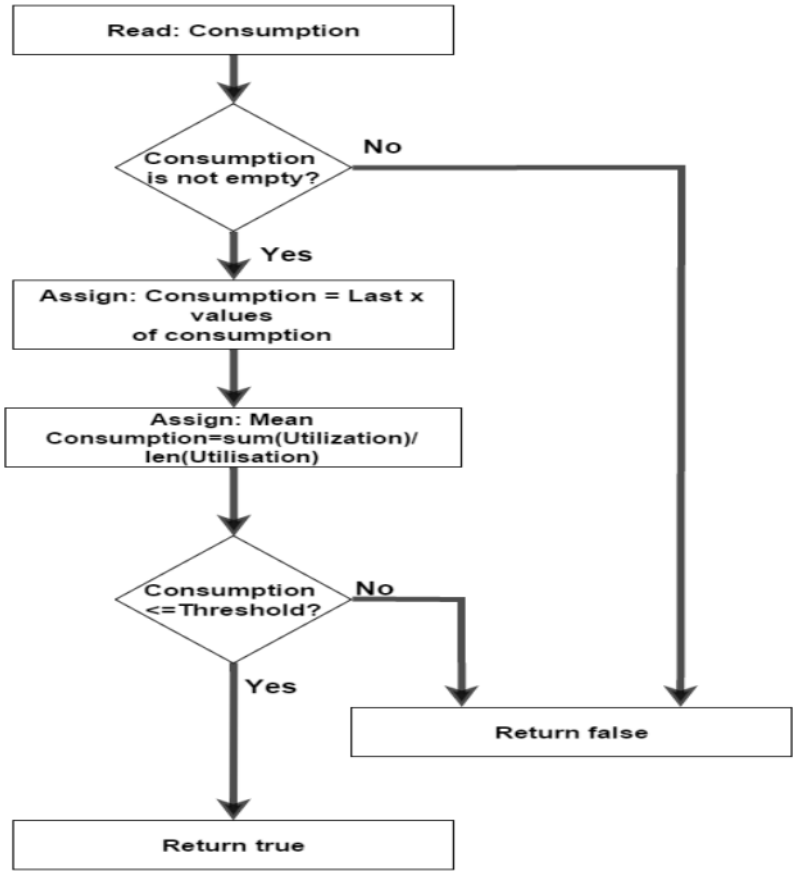

Fig. 2: VM detection

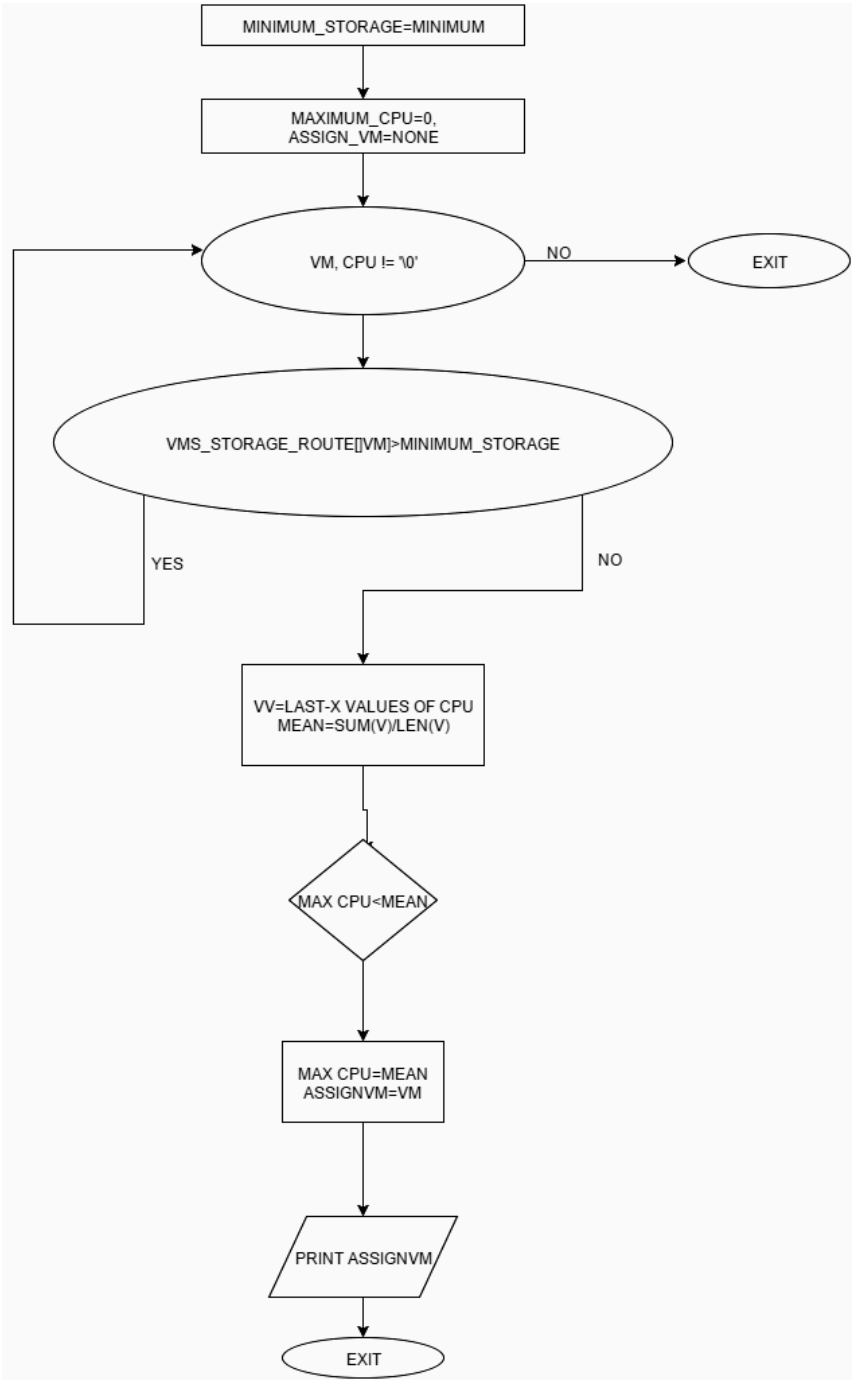

Fig. 3: VM selection

VM selection: When we find the exact node which is overloaded, then it is very important to identify the best VM. When the VM is been selected then we should migrate from the node. To solve this issue, VM selection can be implemented here to get an optimal solution. The algorithm (Wood *et al*., 2007) which clearly states that choose the VM which is having a limited amount of storage. This reduces time in live migration time period (Cleveland, 1979). After this use a maximum CPU utilization for the selected VM. This method is known as minimum migration time with maximum CPU utilization (Fig. 3, Algorithm B).

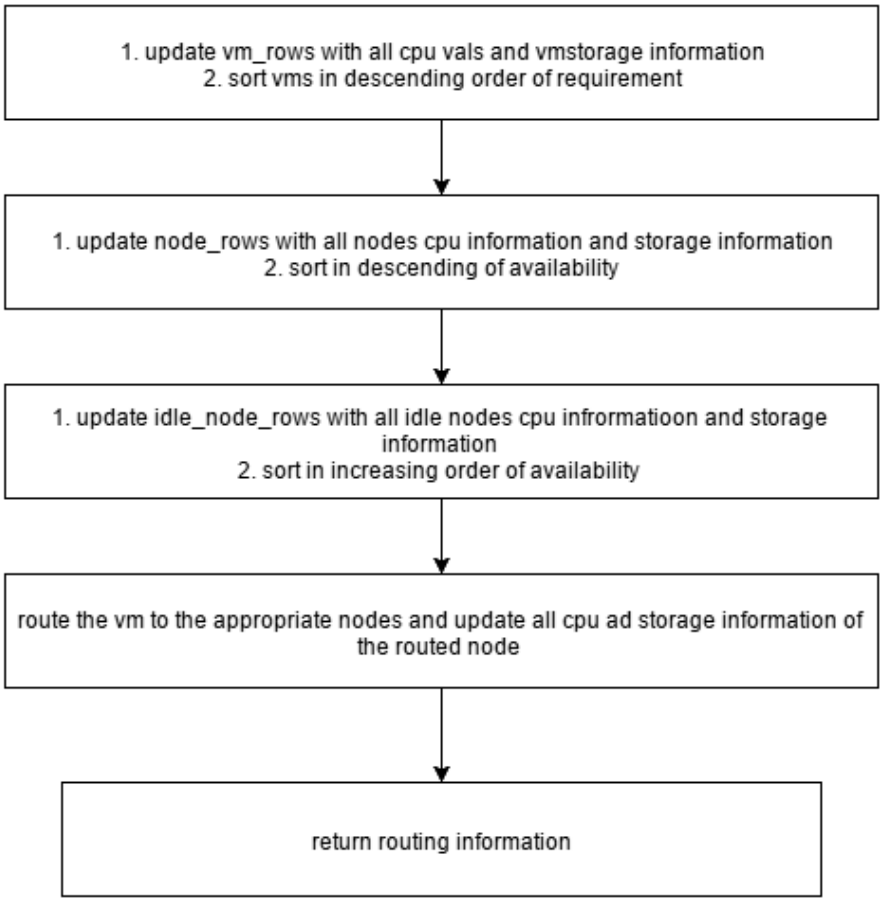

Fig. 4: VM placement

Algorithm B; CPU utilization:

Input: x, vmsCpuRoute, vmsStorageRoute
Output: A VM to migrate
minimumStorage←minimum(values of vmsStorageRoute)
maximunCpu←0assignVm←none
forvm, cpu in vmsCpuRoute
do
If (vmsStorageRoute [vm]>minimumStorage) then
Continue
V←last_x values of cpu
Mean←sum (V)/len (V)
If (maximunCpu<mean) then
maximunCpu←mean
assignVm←vm
Return assignVm

VM placement: Some of the methods under the bin packing problem are first fit method, first fit decreasing method, best fit method, worst fit method, etc., where the bin size can be fixed or variable. These bin sizes represent the CPU capacity in the physical nodes and the items mapped to the bin will the virtual machines. There may be additional parameters like the CPU storage or the amount of RAM required as said (Beloglazov *et al*., 2013). Making a new step with open stack neat they had tried to change the best fit algorithm which shows to use no >11/9. OTP +1 bins that makes an optimal conclusion with the number of bins. This BFD algorithm clearly sates about the how to control over an additional constrains which includes, current ideal node and the amount of storage needed by the VM. If there is another node which is in an execution mode then ideal node cannot be placed here (Fig. 4, Algorithm C).

Algorithm C; node execution:

Input: x, nodecpu, nodeStorage, ideal nodecpu, idealnodestorage, vmscpu, vmsstorage
Output: A map of VM UUIDs to node names
vmRows←empty list
forvm, cpu in vmsCpu
do
vals←last n values of cpu
Append a tuple of the mean of vals, vmsStorage[vm], and vm to vmRows
vms←sortDecreasing (vmRows)
nodeRows? empty list
for node, cpu in nodesCpu
do
Append a tuple of cpu, nodesStorage [node], node to nodeRows
Nodes←sortIncreasing (nodeRows)

```
idleNodeRows←empty list
for node, cpu in idleNodesCpu
do
    Append a tuple of cpu, idleNodesStorage[node], node to
idleNodeRows
idleNodes←sortIncreasing (idleNodeRows)
 Routing←empty map
 For vmCpu, vmStorage, vmUuid in vms
do
    Route to←false
    While not route to do
        Allocated←false
    For every node in nodes
do
        If (nodesCpu [node]≥vmCpu and nodesStorage [node]≥vmStorage)
then
            Routing [vmUuid]←node
nodesCpu [node]←nodesCpu [node]-vmCpu
nodesStorage [node]←nodes Storage [node]-vmStorage
            Route to←true
            Allocated←true
            Break
If (not allocated) then
If (idleNodesIsNotEmpty) then
initializedNode←pop the first from idleNodes
            Append initializedNode to nodes
            Nodes←sortIncreasing (nodes)
nodesCpu [initializedNode [2]]←initializedNode [0]
 Nodes Storage [initializedNode [2]]←initializedNode [1]
    Else
            Break
If len(vms) == len(routing) then
 Return routing
Return empty map
```

This algorithm, initially checks for the availability of the host. If it is free then the virtual machines will be mapped to it. If not, it will try for the next host. If any overload is found, the overload request will be triggered and the overloaded virtual machines will be popped out and redirected to other hosts. For the case of under-loaded scenario, depending upon the input traffic, either new virtual machines will be mapped to it or the virtual machines existing in the under-loaded host will be popped out to make it idle and then it will be switched off.

RESULTS AND DISCUSSION

Deploying this algorithm, the scenario is first checked to know whether the hosts or the physical nodes are under-loaded or overloaded. Knowing this, the algorithm acts according to the need of the situation, either by picking up over-loaded VM(s) and re-mapping them to other hosts or in case of under-load, either pop out the VM(s) and make the host idle or map some more VM(s) and balance the load.

Keeping this logic, we are trying to compare the working of this algorithm with various factors like power consumption and total price with respect to best fit, first fit and worst fit to know which of all gives the optimal result. The results as estimated by CloudSim (Calheiros *et al*., 2011) are shown as.

Figure 5 shows VM requests VS Power Consumption with respect to best fit, first fit and worst fit algorithms. The consumption of power in optimized best fit algorithm is very low when compared to the other standard best fit, first fit and the worst fit algorithm. Hence while talking about power consumption, our optimized best fit algorithm holds best.

Figure 6 shows the number of VM requests that optimised best fit, best fit, first fit and worst fit receive and the total price at that point. When VM requests arrive, optimized best fit has the lowest total price at almost every random point when compared to the other standard algorithms proving that the optimized best fit algorithm would be the best in all possible situations.

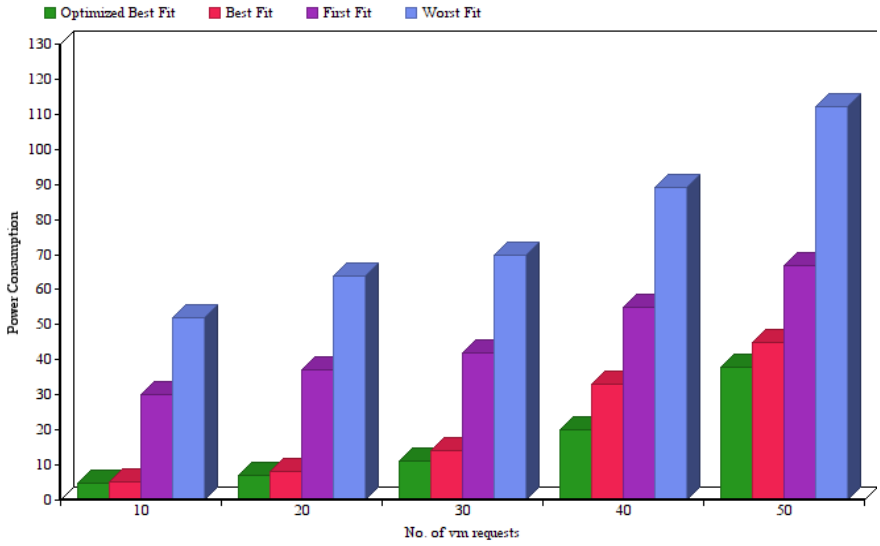

Fig. 5: No. of VM requests vs. power consumption

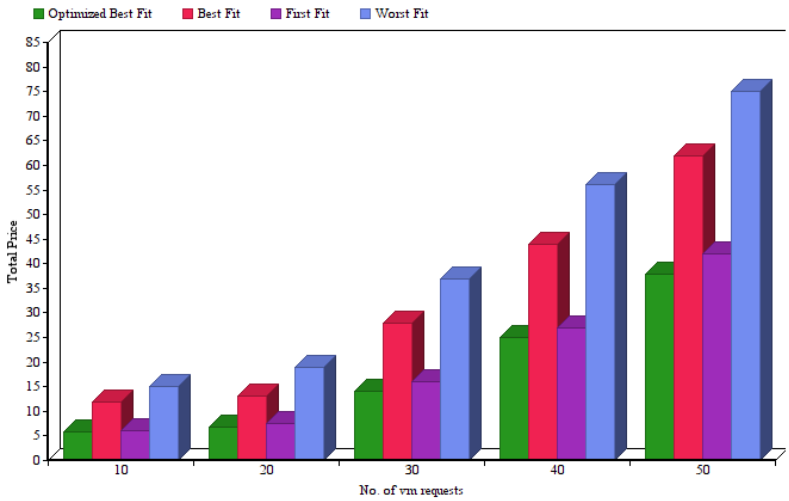

Fig. 6: No. of VM requests vs. total price

CONCLUSION

For the proper placement of virtual machine, the scenario has to be well understood. In this study, we propose the idea of first checking the condition (host over load/host under load) where the virtual machine falls into and the host is detected. After detecting the host, the correct VM is selected and it is placed. The above process is well explained using 3 algorithms. Keeping in mind the various constraints available, 2 graphs have been drawn where the power consumption and the total price is plotted against the No. of VM requests. The generated algorithm proves to stand better than the standard best fit algorithm. Thus, the optimization of virtual machine placement is done.

REFERENCES

Barroso, L.A., 2005. The price of performance. Queue, 3: 48-53.

Beloglazov, A., 2013. Energy-efficient management of virtual machines in data centers for cloud computing. PhD Thesis, Department of Computing and Information Systems, The University of Melbourne, Berlin, Germany.

Buyya, R., C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, 2009. Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst., 25: 599-616.

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A.F. de Rose and R. Buyya, 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Pract. Experience, 41: 23-50.

Cleveland, W.S. and C. Loader, 1996. Smoothing by Local Regression: Principles and Methods. In: Statistical Theory and Computational Aspects of Smoothing. Wolfgang, P.D.H. and P.D.M.G. Schimek (Eds.). Physica-Verlag HD, Heidelberg, Germany, ISBN: 978-3-7908-0930-5, pp: 10-49.

Cleveland, W.S., 1979. Robust locally weighted regression and smoothing scatterplots. Robust locally weighted regression and smoothing scatterplots. 74: 829-836.

Feller, E., L. Rilling and C. Morin, 2012. Snooze: A scalable and autonomic virtual machine management framework for private clouds. Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), May 13-16, 2012, IEEE Computer Society, Ottawa, Canada, ISBN: 978-0-7695-4691-9, pp: 482-489.

Flinn, J. and M. Satyanarayanan, 2004. Managing battery lifetime with energy-aware adaptation. ACM. Trans. Comput. Syst. TOCS., 22: 137-179.

Jung, G., M.A. Hiltunen, K.R. Joshi, R.D. Schlichting and C. Pu, 2010. Mistral: Dynamically managing power, performance and adaptation cost in cloud infrastructures. Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems (ICDCS), June 62-73, 2010, IEEE, Genoa, Italy, ISBN: 978-1-4244-7261-1, pp: 62-73.

Prakash, P., G. Kousalya, S.K. Vasudevan and K.K. Rangaraju, 2014. Distributive power migration and management algorithm for cloud environment. J. Comput. Sci., 10: 484-491.

Wood, T., P. Shenoy, A. Venkataramani and M. Yousif, 2007. Black-box and gray-box strategies for virtual machine migration. Proceedings of the 4th USENIX Conference on Networked Systems Design and Implementation, April 11-13, 2007, Cambridge, Massachusetts, USA., pp: 17-17.