Efficient Virtual Machine Placement for On-Demand Access to Infrastructure Resources in Cloud Computing

Sameer Kumar Mandal
Dept. of Computer Science & Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, India

Pabitra Mohan Khilar
Dept. of Computer Science & Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, India

ABSTRACT
Cloud computing is the utility computing that provides virtualized resources, applications, and services using distributed network and Internet. Cloud computing service offers the ability to scale up and scale down your computing requirements and most importantly to reduce the cost of deployment. Many organizations are migrating to cloud computing services to lower the risk and for better business continuity.In case of on-demand access user requests infrastructure services for immediate access and for a very short interval of time, they have to pay certain charge depending upon that duration.In cloud computing, infrastructure requests are served by the allocation of virtual machines to those requests; these virtual machines should be placed on the underlying hardware infrastructure called datacenter. In this paper we have proposed a model for the efficient allocation of virtual machines on the cloud infrastructure to reduce the allocation time and to optimize the resource utilization. The proposed model is simulated and its performance is compared with two other existing models.

General Terms
Cloud Computing, Virtual Machine andResource allocation

Keywords
Cloud Computing, Virtualization, Resource Allocation, Resource Utilization, Scheduling, Infrastructure as a Service (IaaS), Open Nebula, Eucalyptus

1. INTRODUCTION
Cloud computing is the utility computing that provides unlimited virtualized resource to build a customized infrastructureor platform to run applications or full phase services as a pay-as-you-use basis. Cloud computing has changed the paradigm of system deployment. The details of complex system implementation are abstracted from the end user and with the help of virtualization techniques resources are virtualized that gives an illusion of infinitely scalable and universally available resources [3][4]. In cloud computingservices are offered with the help of Internet to the end users as Something-as-a-Service. Three universally accepted service models are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [5]. Software as a Service provides a complete environment with pre-installed applications along with user interface. Client can access these applications using any device capable of operating a web browser. Platform as a Service provides a framework for the developers to create and deploy their own application on a hosted infrastructure. The customer does not have to bother about the underlyinghardware; they only have to configure the operating environment in the interface provided to them and they can use any programing language supported by the cloud service provider to build their application in that environment.In Infrastructure as a Service computing model client can borrow the fundamental hardware resources for building their own framework. They can customize their entire framework with the help of virtual machines, virtual storage, virtual network *etc*. The client does not have to manage the actual physical resources, as they are provided with virtual resources which can be managed programmatically.

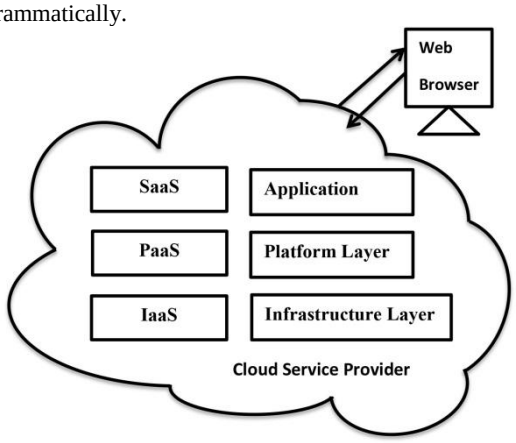

Fig 1: Cloud Computing Service Models

In Infrastructure as a Service model the infrastructure requests are served by allocating virtual machines to those requests [4][10]. So on service providers side one of the primary concern is to allocate the virtual machine images into the actual physical resource *i.e.* the underlying hardware infrastructure. This allocation should be efficient so that resource utilization will be optimized and more requests can be served in lesser time. The virtual machine placement strategy in cloud computing can be classified into three categories [6]; reservation, on-demand access and spot markets. In case of reservation type a user has to pay a certain fee for a particular period for each instance of a virtual machine. In case of on-demand access user requests virtual machine for immediate access for a very short interval of time, and pay the charge depending upon that duration. While in case of spot markets the users need to specify the amount they are willing to pay for the requested virtual machines, as in spot markets there is frequent fluctuation in provider's price. The users are allocated with virtual machines only when the provider's price is same or less than the user's fee. This allocation part is done by the hypervisor [2] of the service provider. The hypervisor contains the list of all the physical resources available in the datacenters and information about their states. So the hypervisor has to design properly and a

robust resource allocation algorithm should run in the hypervisor to do this work efficiently.Since clients or users are provided with the computing resources only according to their requirement so the rest of it can be provided to other users. Many open source cloud computing solutions like Xen Cloud Platform, Nimbus, OpenNebula, Eucalyptus, TPlatform, Apache Virtual Computing lab, Enomaly etc. [12] have addressed this issue. But the framework of OpenNebula and Eucalyptus is quite impressive.

In this paper we have designed an efficient algorithm that follows a best fit strategy for allocation of virtual machine requests to the physical host. Because of the best fit strategy the resource utilization is optimized to some extent. We have also followed a tree based approach to represent the resource list, instead of maintaining a resource queue for quick allocation of virtual machine to the host machines.

The rest of this paper is organized as follows. In section 2 we discuss prior works which address issues related to this topic. Section 3 describes the proposed framework followed by proposed algorithm in sub section 3.1. Section 4 presents the performance evaluation of the proposed algorithm. Finally Section 5 concludes the paper by summarization of the work.

2. RELATED WORK

OpenNebula is an open source toolkit used for deploying cloud service over the underlying hardware devices, the deployment can be either private, public or hybrid [7][14]. It manages all the storage and computing requirements of the deployed infrastructure. It uses the kernel based virtual machine- KVM and can fit into any existing datacenter by integrating it with storage and networking solution. The virtual machine scheduler used in OpenNebula is known as match making scheduler, which uses a predefined rank [9][12] for prioritizing the available resources. Based on the priority, this scheduler places the virtual machine on the host having the highest priority [11]. This algorithm takes requirements and predefined rank as its input and produces the resource number in which to place the virtual machine as output. When a virtual machine request arrives at the scheduler, first it sweeps away the resources which are not befitting into the requirement. The rank scheduling algorithm is delineated below.

1: ***Start***

2: ***for*** *each Host ∈Host_List* ***do***

3: ***if*** *(Host satisfies the Requirements)* ***then***

4: *add (Member_list, Host);*

5: ***end***

6: ***end***

7: Priority_list←Sort_by_rank (Member_list, Rank);

8: Allocated_Host←Priority_list(1);

9: ***end***

Eucalyptus or ‘elastic utility computing architecture for linking your program to useful systems’ is an open source tool; used to build framework for a private cloud [15][16]. Its interface is based on Amazon EC2 cloud platform. It was contrived to endorse third party extensions. Eucalyptus buttress virtual machines over Xen hypervisor. For placement of a virtual machine on the underlying host Eucalyptus uses Round robin resource selection and Greedy First fit algorithm [9][12]. In first fit Greedy strategy whichever node that can run the virtual machine found first is selected as the host for virtual machine placement. However the Round Robin resource selection strategy keeps on selecting the host nodes until it finds a suitable host node that can run the virtual machine. Both of these algorithms take virtual machine requests as input and produce the resource number in which to place the resource as output. The greedy virtual machine placement algorithm is delineated below.

1: ***Start***

2: Initialize res_id← -1; slp_res_id←-1; completed ← 0;

3: ***for*** *all resource ∈Res_List&not completed*

4: ***do***

5: ***if****(resource∈ (Suspended_state || Waking_state) &res_id = -1)* ***then***

6: *res_id←curr_res_id;*

7: *completed ← completed+1;*

8: ***end***

9: ***if*** *(resource ∈Sleeping_state&res_id =-1)* ***then***

10: *capacity ←find (remaing_res_capacity)*

11: ***if*** *(capacity > 0)* ***then***

12: *slp_res_id←res_id;*

13: ***end***

14: ***end***

15: ***end***

16: ***if*** *(res_id== -1&slp_res_id == -1)* ***then***

17: *out_res_id← -1;*

18: *return out_res_id;*

19: ***else if*** *(res_id = -1)* ***then***

20: *fetch resource from Res_List having res_id;*

21: *out_res_id← fetched res_id;22:* ***end***

23: ***else if*** *(slp_res_id == -1)* ***then***

24: *fetch resource from Res_list having slp_res_id*

25: *out_res_id← fetched slp_res_id;26:* ***end***

27: ***else***

28: ***if*** *(resource ∈Sleep_state)* ***then***

29: *Start that resource;*

30: *out_res_id← started res_id;*

31: ***end****32:* ***end***

33:return out_res_id;34: ***end***

Gupta A. *et al*. [17] proposed a HPC aware scheduler for Infrastructure as a Service cloud for improving performance of HPC applications in cloud. They have addressed the problem of adequate allocation of assorted range of application by using monotonous pool of resources. Service level agreement (SLA) is also taken into consideration. They have designed a two-step process methodology; first process discriminates applications based on the utilization of shared

resources in a multi core node, while in second process applications are grouped based on their complementary profile. A multi-dimensional online bin packing heuristic is applied here to achieve optimal resource allocation by considering high performance application. They have implemented the proposed algorithm using Nova scheduler, which is the scheduler for OpenStack (an open source cloud computing framework) and have shown their algorithm is performing 45% better compared to dedicated execution while limiting jitter to 8%.

Zang Y. *et al.* [13] proposed a scheme for integration of on-demand infrastructure resource by considering resource consumption and allocation. They introduced two adoptive control loops for managing infrastructure requests for PaaS application: resource allocation loop and resource consumption optimization loop. To improve the resource utilization they used optimization loop which includes a management function and to provide appropriate amount of resource they used allocation loop. They have combined these two loops for providing resource on-demand by running it repeatedly. A framework called SmartRod is implemented to verify the correctness of the scheme.

Kim H. *et al.* [10] designed a system based on Grid middleware to improve performance and resource utilization of cloud system. As the resource requests are dynamic and not monotonous hence static configuration is not capable to serve these requests. But by using virtualization reliability can't be guaranteed, hence they proposed three methods to solve this limitation of virtualization. Xen and Globus toolkit is used to simulate the framework. They performtwo kinds of experiments: middleware overhead analysis and Grid middleware utilization.

3. PROPOSED MODEL

Figure 2 elucidates the model for allocation of virtual machines to physical host machines present in the datacenters. Client/user can place their list of requirements for any service in the interface provided with the help of a web browser, that may require computational and storage infrastructure. That request is communicated to the cloud service provider through internet. The service provider will serve the requests by placing individual requests to suitable virtual machines that can perform the requisite operations. The virtual machine specification is now forwarded to the hypervisor of the service provider. Hypervisor uses the virtualization tools to create virtual machines of the specified specification. The virtual machine manager keeps track of the created virtual machines. But the problem is to map the virtual machine requests to the host machines. To resolve this problem virtual machine manager sends the virtual machine specifications to the virtual machine scheduler (VM Scheduler). But in this model instead of sending that request directly, a binary search tree of the requested virtual machines is created and that is sent to the VM Scheduler. This is done to improve the resource utilization, which is explained in the next sub section in the algorithm. Now VM scheduler will take the maximum requirement node from that virtual machine tree and will search for a host that will best fit the requirement. For implementing the best-fit strategy all the physical resources are also logically organized in a binary search tree, sincesearching a host using this data structure will take in average *O (log N)* time where *N* is the number of physical hosts. This will reduce the time for allocation. After getting the proper host the scheduler will return the host number to the virtual machine manager for placement of virtual machine on that host.

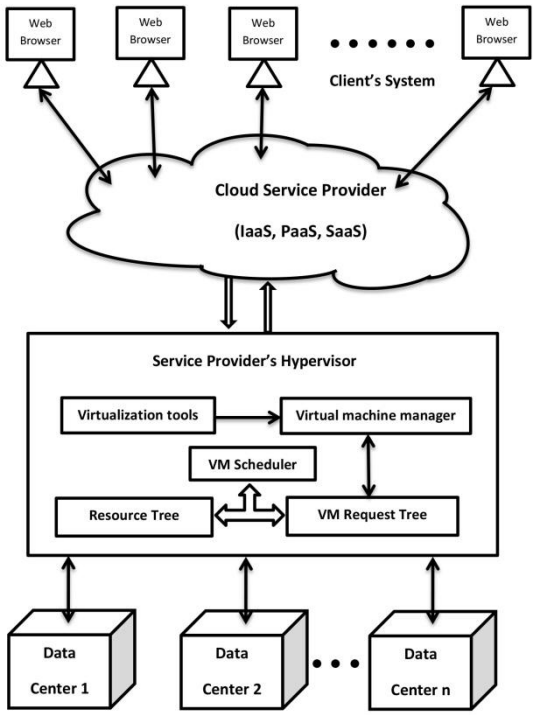

Fig 2: Proposed framework for VM placement

Now the virtual machine manager has all information about the virtual machine and its location; it will now send a service activation message to the client/user. After that client/user can access the service for the duration specified.

3.1 The VM Scheduler Algorithm

Let $Host_{List}$= $\{H_1, H_2, H_3, \ldots H_n\}$ is the list of physical hosts available which have to process the virtual machine requests. The virtual machine requests that have been made in a particular time interval are collected at the virtual machine manager and are stored in aVM_{Queue}, VM_{Queue}= $\{VM_1, VM_2, VM_3, \ldots VM_n\}$. The time interval is used to avoid the problem of starvation and is very small (let's say two seconds) to avoid the delay. Here we have to do a mapping $VM_{Queue} \rightarrow Host_{List}$,the mapping should be done in such a way that it will optimize the use of resources and should be faster. Figure 3 elucidate the flow chart of the proposed algorithm where best fit is decided based on the following formula.

$$\frac{RVMS}{AHS} = \begin{cases} 1 & Perfect\ Fit \\ (0,1) & Possible\ Candidate \\ (1,+\infty) & Reject \end{cases}$$

Here *RVMS* is required virtual machine specification, and *AHS* is the available host specification. When a virtual machine request arrives at the scheduler, it calcutes this value at each host arranged in the binary search tree starting from the root node traversing towards the leaf node, until it finds a best fit. If it finds this value to 1 for a perticular host then that is the perfect fit and that host nood is assigned as the best fit node for that virtual machine. If the value calculated is greater than 1 then that host is not a candidate host for that virtual machine and the search will continues to its right child host

node.

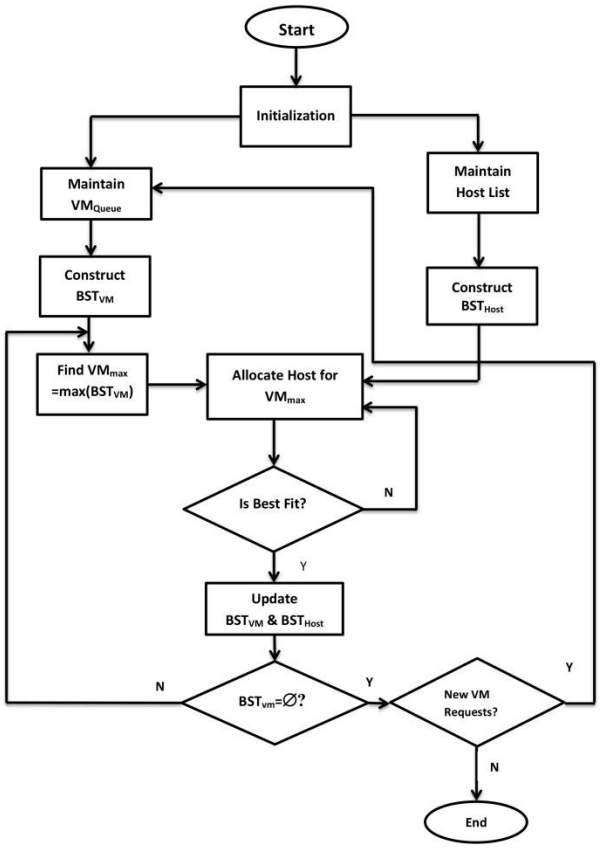

Fig 3: Flow chart for VM Scheduling algorithm

If the value is less than 1 *i.e.* a fraction between 0 to 1 then the search bigins in the direction of left host node and the value at each left and right child node is calculated. If the value is found less than the value that has been calculate at its parent host node then it will stop searching further and declare it's parent node as the best fit node. For example let there are five host nodesH_1 to H_5 present at the datacenter having memory specification (in MB) 1024, 360, 525, 575, 680 respectively. These host machines are organized in the host tree as shown in the figurebelow.

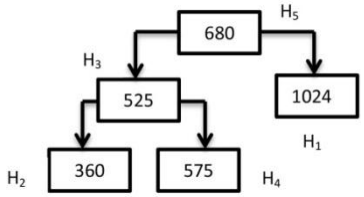

Fig 4: Host representation

Suppose a virtual machine request having memory specifacation 480 MB arrives at the scheduler. Now scheduler will calculate the $RVMS/AHS$ value at the root node *i.e* H_5. The value is found to be 0.71. Now the searchwill continue in left direction *i.e.* H_3. The value at H_3 is 0.91. Next host node is H_2 and the value iscalculated to 1.33 which is greater than 1, hence rejected. So the best fit host is its parent host node having the calculated value 0.91. Hence host H_3 is the best host to run this virtual machine request. The purpose of arranging the virtual machine requests in binary search tree is: suppose two virtual machine *(VM)* requests arrive at a time having memory requirment 510 MB and 400 MB. If the allocation of 510 MB VM is processed first then it will get allocated at the H_3 host, and 400 MB request is allocated at H_4 host. The remaining memory at H_3 is 15 MBand at $H_4$175 MB. But if the 400 MB request is processed first then first it will get allocated to host H_3, the remaining memory is 125 MB and 510 MB request is allocated to host H_4, the remaining memory at $H_4$is 65 MB. So in the first case we are left with a big fragment of memory *i.e.* 175 MB, which can be used to allocate some new requests. In the second case we got two fragments of memory but that may not get utilized (let's say when a VM request for 150 MB memory arrives). Hence this technique improves the resource utilization. Also to insert a new request in that time interval will take in average *O(log n)*time. The VM scheduling algorithm takes VM_{Tree} *i.e* BST_{VM} as input and returns allocated host machine as the output. Thedetailed algorithm is delineated below.

1: ***while*** $BST_{VM} \neq \emptyset$ ***do***

2: $VM_{max} \leftarrow Max(BST_{VM})$;

3: $Host_{Alloc} \leftarrow Allocate(VM_{max}, BST_{Host})$;

4: $Update(BST_{VM}, VM_{max})$;

5: $Update(BST_{Host}, Host_{Alloc})$;

6: ***end***

Algorithm Allocate(VM_{max}, BST_{Host})

1: ***Begin***

2: $curr\_threshold \leftarrow 0$;

3: $threshold \leftarrow \frac{VM_{max}}{BST_{Host}^{Root}}$;

4: $Max\_value \leftarrow BST_{Host}^{Max}$;

5: ***if*** *(threshold == 1)*

6: return BST_{Host}^{Root};

7: ***end***

8: ***if*** *(threshold <1 & curr_threshold < threshold)*

9: $Allocate(VM_{max}, BST_{Host}^{Left})$;

10: $curr\_threshold \leftarrow threshold$;

11: ***end***

12: ***else if*** *(threshold > 1 & curr_threshold < threshold)*

13: $Allocate(VM_{max}, BST_{Host}^{Right})$;

14: $curr\_threshold \leftarrow threshold$;

15: ***end***

16: ***else if*** *(VM_{max}> Max_value)*

17: return 0;

18: ***end***

19: ***else***

20: wait untill the time interval finish; *21:****end***

22: return BST_{Host}^{Root} *;23:* ***end***

4. PERFORMANCE EVALUATION

The proposed model is simulated using CloudSim toolkit [18]; an environment for simulating the cloud computing applications. The proposed VM Scheduling algorithm is compared with OpenNebula Ranking algorithm and Eucalyptus Greedy algorithm, by simulating them using the same toolkit. For simulating this environment we have simulated one datacenter inside which seven host machines of different specifications are running. We also have simulated seven virtual machine requests of different specifications which will be placed on the host machines. The specification of the host machines and virtual machine requests are depicted in table 1 and table 2 respectively.

Table 1. Host machines specification

Host ID	Memory	CPU core	HDD	BW
1	2048	4	50	10000
2	1024	4	50	10000
3	2048	4	50	10000
4	512	2	20	8000
5	1024	4	40	10000
6	1024	4	50	10000
7	512	2	30	8000

Table 2. Virtual machine requests specification

VM ID	Memory	CPU core	HDD	BW
1	256	1	10	500
2	2048	1	10	500
3	1024	1	10	500
4	512	1	10	500
5	1024	1	10	500
6	256	1	10	500
7	512	1	10	500

In this experiment we are only varying the memory requirement (In MB) of virtual machine requests *i.e.* we are considering memory intensive applications. All the three algorithms are executed independently with the same host and virtual machine instances shown in the above two tables. Fig 5 shows the mapping between host machine and virtual machine request. Fig 6 depicts the available memory at each host after all the virtual machines have been scheduled to proper hosts, which shows the resource utilization of threealgorithms. Finally Fig 7 shows the plotted graph between the scheduling algorithms and time consumed by each of them to perform the complete operation.

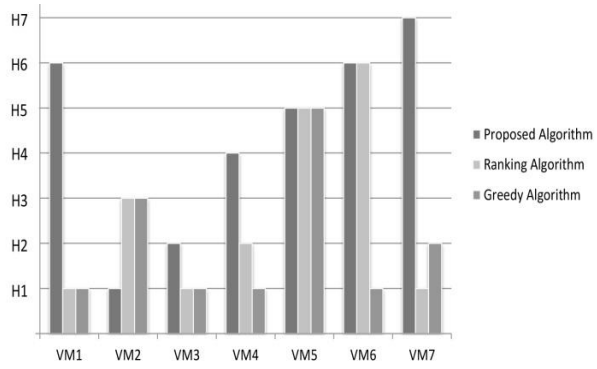

Fig 5: Host machine Vs. Virtual machine allocation

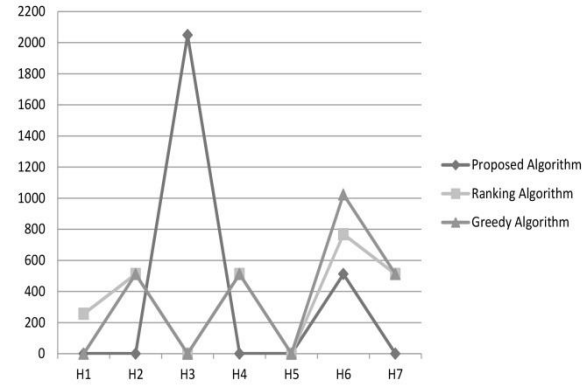

Fig 6: Available memory at each host after allocation

Above graph shows that the highest available memory at a single host is at host H3 using the proposed algorithm, so host H3 can serve a high memory request if any arrives at that time. The proposed algorithm has two big peaks in the graph at H3 and H6, whereas the other algorithms have multiple small peaks which show that the resource utilization of proposed algorithm is better than that of the other two.

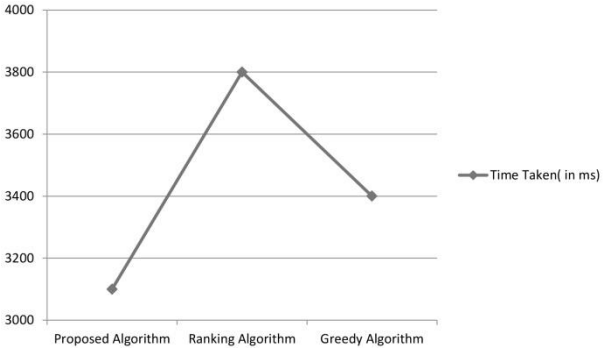

Fig 7: Time taken to allocate the VM requests

5. CONCLUSION

In cloud computing infrastructure requests are served by allocating virtual machines on the underlying physical infrastructure. As in on-demand access user requests resources for a short time interval and for immediate use, hence these requests have to be served quickly by allocating them virtual machines. These virtual machines have to beplaced on proper host machines on the actual infrastructure, so that more number of resource requests can be served in lesser time. Hence a proficient resource allocation algorithm should run at

the hypervisor that can do this job in lesser time and optimize the use of resources. In this paper our experimental results shows that the proposed framework can improve the resource utilization to serve more number of resource requests at a time without compromising the allocation time.

6. REFERENCES

[1] Chunye, G., Jie, L., Qiang, Z., Chen, H. and Zhenghu, G. 2010. The Characteristics of Cloud Computing. 39th International Conference on Parallel Processing Workshop. IEEE Computer Society, 1530-2016.

[2] Boss, G., Malladi, P., Legregni, L. and Hall, H. 2007 how a business can use cloud computing to reduce cost. IBM white paper.

[3] Zhang, S., Zhang, S., Chen, X. and Huo, X. 2010. Cloud Computing Research and Development Trend. 2nd International Conference on Future Networks.

[4] Sosinsky, B. 2012. Cloud Computing Bible. Wiley Publishing Inc.

[5] Mell, P. and Grance, T. 2011. The NIST Definition of Cloud Computing. NIST Special Publication, 800-145.

[6] Mills, K., Filliben, J. and Dabrowski, C. 2011. Comparing VM-Placement Algorithms for On-Demand Clouds. Third IEEE International Conference on Cloud Computing Technology and Science. IEEE Computer Society, 978-0-7695-4622-3.

[7] Endo, P. T. and Goncalves, G., E. 2010. A Survey on Open-Source Cloud Computing Solutions. VIII Workshop on Clouds, Grid Applications.

[8] Sadashiv, N. and Kumar, S. 2011. Cluster, Grid and Cloud Computing: A Detailed Comparison. The 6th International Conference on Computer Science & Education (August 3-5).,SuperStar Virgo, Singapore.

[9] Sotomayor, B., Montero, R. S., Llorente, I. M. and Foster, I. 2009. Internet Computing IEEE (Sept.-Oct.). Volume 13, Issue: 5, page(s) 14-22.

[10] Kim, H., Kim, W., Lee, K., Newby, G. B. and Kim, Y. 2009. Experimental Study to Improve Resource Utilization and Performance of Cloud Systems based on Grid Middleware. KSII The first International Conference on Internet (December 2009).

[11] Zhong, H., Tao, K. and Zhang, X. 2010. An Approach to Optimize Resource Scheduling Algorithm for Open-Source Cloud Systems. The Fifth Annual ChinaGrid Conference. IEEE Computer Society, 978-0-7695-4106-8.

[12] Patel, P. and Singh, A. K. 2012. A Survey on Resource Allocation Algorithms in Cloud Computing Environment. Golden Research Thoughts, Volume 2, Issue. 4 (Oct 2012), ISSN: 2231-5063.

[13] Zhang, Y., Huang, G., Liu, X. and Mei, H. 2010. Integrating Resource Consumption and Allocation for Infrastructure Resource on-Demand. IEEE 3rd International Conference on Cloud Computing. IEEE Computer Society, 978-0-7695-4130-3/10.

[14] OpenNebula. http://opennebula.org/about:about.

[15] The Eucalyptus. http://www.eucalyptus.com/eucalyptus-cloud.

[16] Peng, J., Zhang, X., Lei, Z., Zhang, B., Zhang, W. and Li, Q. 2009. Comparison of Several Cloud Computing Platform. Second International Symposium on Information Science and Engineering. IEEE Computer Society, 978-0-7695-3991-1/09.

[17] Gupta, A., Milojicic, D. and Balle, S. M. 2012. HPC-Aware VM Placement in Infrastructure Clouds. Paralal Programming Laboratory. Department of Computer Science, University of Illinois.

[18] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F. and Buyya, R. 2011. CloudSim: A ToolKit for Modeling and Simulation of Cloud Computing Environment and Evaluation of Resource Provisioning Algorithm.http://www.cloudbus.org/cloudsim/.