Design and Implementation of an Improved Datacenter Broker Policy to Improve the QoS of a Cloud

Tamojit Chatterjee[1], Varun Kumar Ojha[2], Mainak Adhikari[3], Sourav Banerjee[1], Utpal Biswas[4], and Václav Snášel[2]

[1] Dept. of Computer Science & Engineering, Kalyani Govt. Engineering College, India
{tamojit.chatterjee9,souravacademia}@gmail.com
[2] IT4Innovations, VSB Technical University of Ostrava, Ostrava, Czech Republic
{varun.kumar.ojha,vaclav.snasel}@vsb.cz
[3] Dept. of Computer Science & Engg., IMPS College of Engineering and Technology, Malda, India
onlinemainak@yahoo.com
[4] Dept. of Computer Science & Engg., University of Kalyani, India
utpal01in@yahoo.com

Abstract. Cloud Computing offers various remotely accessible services to users either free or on payment. A major issue with Cloud Service Providers (CSP) is to maintain Quality of Service (QoS). The QoS encompasses different parameters, like, smart job allocation strategy, efficient load balancing, response time optimization, reduction in wastage of bandwidth, accountability of the overall system, best Virtual Machine (VM) (which reduce the overall execution time of the requested Cloudlets) selection etc. The Datacenter Broker (DCB) policy helps binding a Cloudlet with a VM. An efficient DCB policy reduces the overall execution time of a Cloudlet. Allocating cloudlets properly to the appropriate VMs in a Datacenter makes a system active, alive and balanced. In present study, we proposed a conductance algorithm for effective allocation of Cloudlets to the VMs in a Datacenter by taking into consideration of power and capacity of VMs, and length of Cloudlets. Experimental results obtained using CloudSim toolkit under heavy loads, establishes performance supremacy of our proposed algorithm over existing DCB algorithm.

Keywords: Cloud Computing, Quality of Service, Cloud Service Provider, Virtual Machine, Datacenter, Datacenter Broker.

1 Introduction

Cloud computing [1, 2, 3, 14] is an emerging computer paradigm on the underlying foundations of service oriented architecture [4, 5], virtualization [7, 13] and utility computing. It has brought the concept of physical location independence to its true meaning. Since, what it does, is provided users with high end infrastructure even when the user is at a location where such infrastructure is impossible to setup. Cloud computing is thus a business package where companies provide computation power, huge storage space and various other software services to user applications through a common interface without the knowledge of the location of resources. In this domain,

P. Krömer et al. (eds.), *Proceedings of the Fifth Intern. Conf. on Innov. in Bio-Inspired Comput. and Appl. IBICA 2014*, Advances in Intelligent Systems and Computing 303,
DOI: 10.1007/978-3-319-08156-4_28,

the background activities like Virtual Machine (VM) allocation, load sharing, load balancing, process migration, distributed shared memory access is completely abstracted from the user's purview. Here, the end users or the customers can access the cloud based applications [5, 6] as well as infrastructure through logging in to a Cloud interface. The Cloud application providers strive to give the same or better service and performance than if the software programs were installed locally on end-user machines. The broker policy for binding or allocating Cloudlet to VM in a heterogeneous Cloud like environment is an important issue. To make the Cloud services proficient in that environment, one of its challenges is to provide an efficient broker policy. The Cloudlet binding policy plays a vital role to improve the overall performance and minimize the execution time. So many broker policies are there in distributed computing to allocate the Cloudlets to the different resources or VMs optimally. A proper allocation policy may lead to a good assignment of Cloudlets to the suitable resources or VMs that may eventually lead to improve the Quality of Service of the overall system.

In present study, we proposed an improved Datacenter Broker algorithm that will allocate a Cloudlet to a VM, on providing the correct VM with correct load, so that they are not idle in a Datacenter and as a result the finish time of the Cloudlets are low and hence the system utilization has been improved. It also takes length of the Cloudlets into consideration to be processed, so that lengthy Cloudlets are allocated to the most powerful VMs. We are using the CloudSim 3.0.3 simulation platform [9, 18] which supports First Come First Serve (FCFS) policy [7] and Round Robin (RR) scheduling strategies for internal scheduling of jobs as well as for VM creation. FCFS and RR suffer from Long average waiting time for longer jobs necessitating for the deployment of a better scheduling strategy at the cluster level. Our proposed Datacenter Broker algorithm provides better result than the aforementioned policies. It takes into consideration the length of the Cloudlets to be processed so the lengthy (CPU intensive) cloudlets go the most powerful VMs.

The remaining part of this paper is organized as follows. Section 2 described about the CloudSim Toolkits. In section 3, we shall discuss our proposed DCB algorithm for Cloudlet allocation in a VM by Datacenter Broker with an example to illustrate the prominence of the proposed algorithm. In section 4, comparison and simulated result is shown. Finally, section 5 shall conclude our discussion with future research direction.

2 Cloudsim Toolkits

Several Grid simulators [10, 12], such as GridSim, SimGrid, and GangSim are capable of modelling and simulating the Grid application in a distributed environment, but fails to support the infrastructure and application-level requirements arising from Cloud computing paradigm [16]. A Cloud infrastructure modelling and simulation toolkits must support for real-time trading of services between customers and providers. Open source CloudSim framework [18] shown in figure 1 is developed on GridSim toolkit [17] offers support for economic-driven resource management and application scheduling simulation. It provides users a series of extended entities and methods. In addition, it helps users to analyze their own scheduling and allocation

strategy at different levels including modification of module deployment techniques and conduct related performance testing by expanding few interfaces. Present study aims at expanding CloudSim by utilizing the broker policy. The Datacenter Broker policy is a decision making procedure through which makes the best match between cloudlets and VMs. Modules of CloudSim toolkit which relevance to our research are as follows.

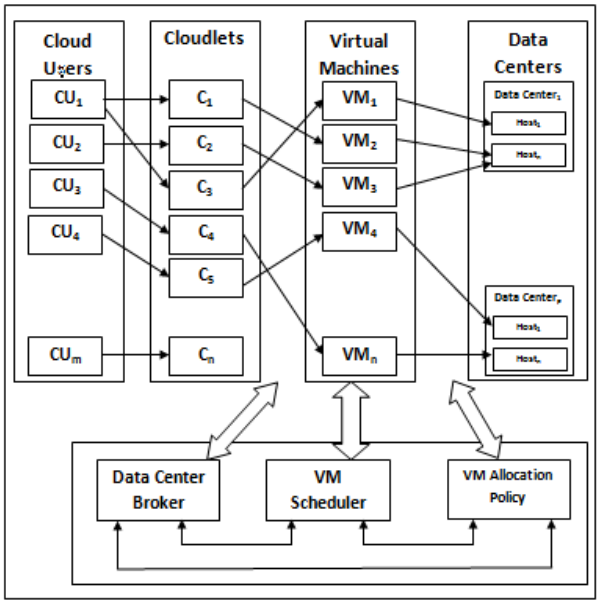

Fig. 1. Cloudsim Work Style

- **Cloud Information Service (CIS) –** CIS [7] provides database level match-making services; it maps user requests to suitable cloud providers. CIS and Datacenter Broker of CloudSim realized resource discovery and information interaction, it is the core of simulated scheduling [17, 18, 19].
- **Datacenter Broker (DCB) –** This class models a broker, which is responsible for mediating between users and service providers depending on users' QoS requirements [7]. And the broker deploys service tasks across clouds. User-developed scheduling algorithms are implemented in Datacenter Broker method.
- **VM Scheduler –** VM scheduler is an abstract class implemented by a Host component, represents the policies (space-shared, time-shared) required for allocating processing power to VMs [8].
- **VM Allocation Policy –** VM Allocation Policy is used to select available host in a Datacenter, which meets the memory, storage, and availability requirement for a VM deployment. The Datacenter Broker allocates or binds the cloudlets to the first available VM. In table I, there are four jobs and two VMs. According to this policy, Cloudlet CL_1 allocated to VM_1, CL_2 allocated to VM_2 and CL_3 and CL_4 allocated to the VM_1 and VM_2 respectively.

Table 1. Cloudlet Binding with VM

CLOUDLET	VIRTUAL MACHINE (VM)
CL_1	VM_1
CL_2	VM_2
CL_3	VM_1
CL_4	VM_2

The existing Datacenter Broker Policy suffers from various disadvantages, like it is non-intelligent in the decisions it takes that don't consider the capacity of the VMs and the length of the Cloudlets. Hence it suffers from the following drawbacks: As a result, large Cloudlets are often assigned to the VMs with low MIPS and hence take a longer time to execute as well as increasing the waiting time and the response time of the Cloudlets. More over sometimes it may also happen that the most powerful VMs get the least lengthy Cloudlets and hence its resource utilization gets wasted and at the same time decreasing the overall performance of the system.

3 Proposed Conductance Algorithm

We shall make the following modification to DCB in order to improve its performance. At first, VMs are sorted in ascending order inside a Datacenter using their processing capability characteristic i.e. Millions of Instructions per Second (MIPS).

$$\textit{Expected Processing Time} = (\textit{MI of a Cloudlet} / \textit{MIPS of VM}) \quad (1)$$

At second step, best VMs in terms of processing capability will be selected for a Cloudlet in order to improve the overall performance of the system.

3.1 Cloudlet Allocation Strategies

The Cloudlets may be generated in two ways, one is automatically in a random order with difference in all its characteristics otherwise the Cloudlets may be generated and classify by user. The generated Cloudlets will be submitted to the Datacenter Broker module which contains the information regarding the VMs. In our proposed Cloudlet allocation algorithm, Conductance Algorithm tries to allocate the Cloudlets to VMs correctly by drawing inspiration from the way water works in a pipe system. Before start the allocation operation, the Datacenter Broker sorts the available VMs in a Datacenter in ascending order according to their capacity. When a batch of Cloudlets are submitted to the Datacenter Broker, then DCB first arrange the batch of Cloudlets in ascending order according to their length and apply conductance algorithm on sorted Cloudlets where the higher MIPS VMs should be allotted more percentage of the load than the lower MIPS VMs.

3.2 Working Principle

In proposed Conductance Algorithm, consider each VM as a pipe which is shown in figure 2. Now the basic idea is that the thicker the pipe the more water it can fit into it i.e. the higher MIPS of a VMseems the higher conductance (processing power).
Now, we calculate the conductance (processing power) as per (2) of each VM as the ratio of its capacity to the sum of the capacity of all the VMs present in a Datacenter.

$$Conductance_i = {Mips_i}/{\sum_{i=1}^{N} Mips_i} \quad (2)$$

After the calculation of the Conductance, multiply the Conductance of that particular VM with the length of the cloudlet list. To determine the strip length as per (3) of the cloudlet list the VM can process (Conduct). That is to determine the part of the Cloudlet list the VM can acquire i.e. the no. of Cloudlets the VM can process. Pseudo code of proposed Conductance Algorithm is given in algorithm 1.

$$stripLength_i = conductance_i * (length\ of\ the\ cloudlet\ list) \quad (3)$$

Algorithm 1: Pseudo Code of Conductance Algorithm

```
Sort: Cloudlet list in decreasing order of their length
Sort: vmList in decreasing order of their MIPS
Declare totalMIPS
Let totalMIPS = 0
    For each vm in vmList begin
          totalMips += vm.getMips()
    end
Declare conductance, length
Let length = cloudletList.size ()
Declare from
Let from = cloudletList.size() - 1;
    For each vm in vmList begin
       conductance = vm.getMips()/totalMips;
       Let range = roundOff(conductance*length);
       while(range-- > 0 AND from > 0)
            Bind the cloudlet having id as form to vm
    end
Done
```

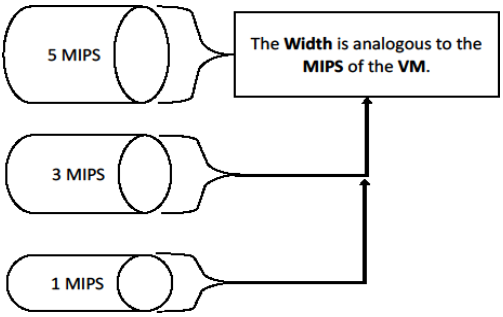

Fig. 2. Pictorial representations of VMs as a Pipe

3.3 Demonstration of Conductance Algorithm

For simplicity and better explanation, we have six Cloudlets, a single Datacenter and three VMs. The Cloudlets are assigned to the VM with the help of proposed Conductance Algorithm. The sorted VMs with their capacity are shown in figure 2. A batch of six Cloudlets with their length in Million Instructions (MI) is shown in figure 3.

Fig. 3. Batch of Cloudlets with length [All Lengths are in MI (Million Instructions)]

Cloudlets are sorted in ascending order and Conductance of the three VMs are computed using (2) in the Datacenter is shown in figure 4.

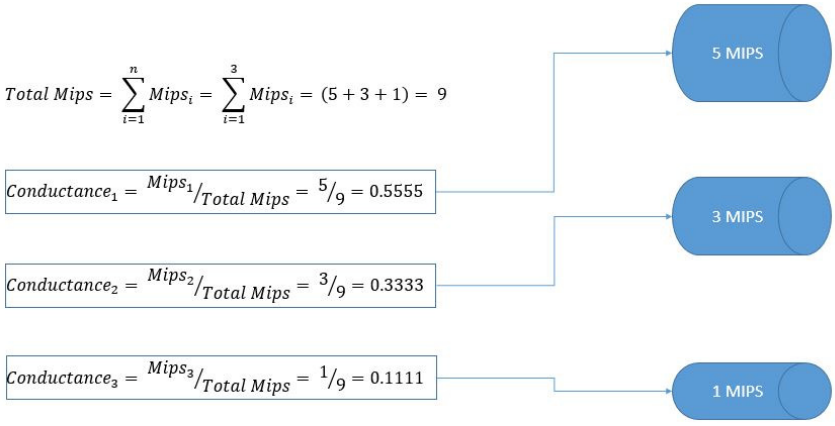

Fig. 4. Conductance calculation of the VMs

Finally, strip lengths of the VMs are computed using (3) and sorted Cloudlets are allocated to the proper VM in the Datacenter according to the strip value as shown in figure 5.where length of Cloudlet list is 6.

From figure 5, it may be observe that VMs whose MIPS is maximum, is allocated to more Cloudlets compare to the VMs who's MIPS is small. As shown in figure 5, VM whose MIPS is 5, is allocated to 3 highest length Cloudlets. Similarly, VM, whose capacity is 3, is allocated to next two highest capacities Cloudlet. Finally the VN, whose capacity is 1, is allocated to smallest length Cloudlet from the list. Therefore, our proposed Conductance algorithm substantially improves the *execution time* of the Cloudlets as well as improves the *makespan* [15] of overall system.

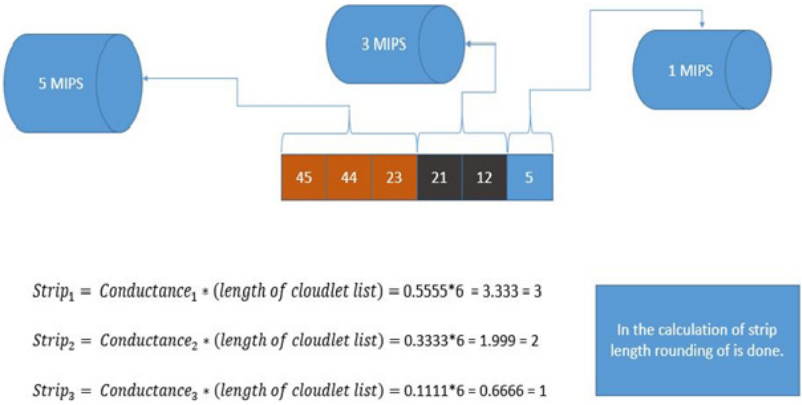

Fig. 5. Strip calculation of the VMs

3.4 Limitations of Conductance Algorithm

- It does not perform better than the existing Datacenter Broker Policy if the MIPS of the VMs of a Datacenter are all equal and if numbers of Cloudlets much less than numbers of VMs.
- The low MIPS VMs sometimes get free too quickly thus wasting its resources.
- The high MIPS VMs sometimes get overloaded when the Length of the longest Cloudlets assigned to them are very large.
- The accuracy of the number of Cloudlets assigned to VMs depends on the rounding off algorithm and if naive rounding algorithms are used then it will result in incorrect assignments.

4 Comparison and Simulated Result

The simulated comparison result of 20 Cloudlets and 4 VMs are shown in figures 6 and 7. The *makespan* of individual VMs of two different Cloudlet allocation policies are shown in figure 6 where x-axis denotes the VM IDs and the y-axis denotes the *makespan* of the VMs. From figure 6, it is clear that the *makespan* of VM_0 and VM_2 using Conductance algorithm are significantly better (smaller is better) than existing DCB, but the *makespan* of VM_1 and VM_3 are insignificantly little poor.

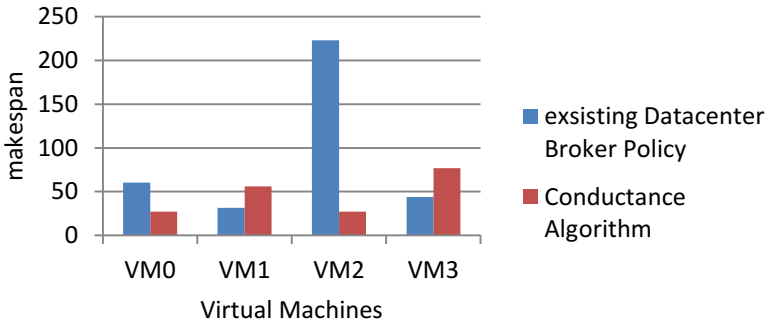

Fig. 6. Simulated comparison result of two different Cloudlet allocations Policies

Overall *makespan* of the system using existing Datacenter allocation Policy is 223. Whereas, overall *makespan* of the system using Conductance Algorithm is 76.8. Figure 7 illustrates comparison between aforementioned allocation policies in terms of execution time where, x axis indicates Cloudlet IDs while y axis indicate execution time. From figure 7, it may easily be concluded that the execution time of the cloudlets are significantly improved using the proposed Conductance algorithm. Therefore, we may conclude that, Conductance algorithm performs substantially better than existing DCB allocation policy.

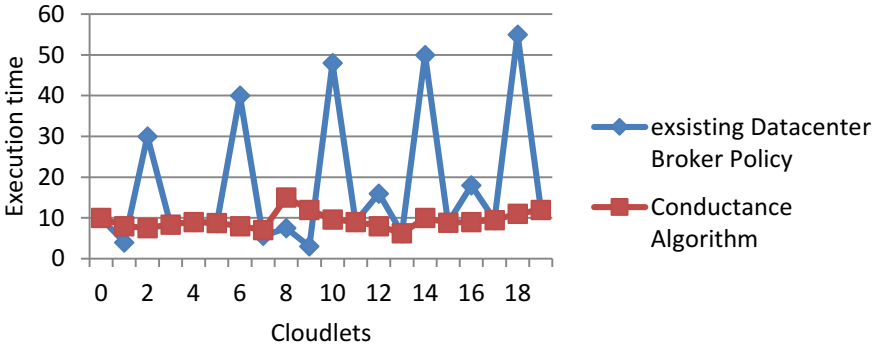

Fig. 7. Simulated comparison result of two different Cloudlet allocations Policies

5 Conclusion and Future Work

Our study encompasses discussed on Cloudlet allocation to the different VMs inside a Datacenter with the help the Conductance algorithm which provides better *makespan* of the VMs in the Datacenter and the execution time of the Cloudlets also reduced. Hence the QoS and the resource utilization of overall system must be improved. In our future study, we shall focus on development of DCB module using intelligent algorithms to identify loads intelligently for the entire available VMs inside a Datacenter and keep all the VMs busy as much as possible so that *makespan* of whole

system would improve. The capacity of the VMs will be indexed in a hash table so that information regarding the execution load of all VMs are dynamically updated. We shall also investigate live VM migration to the other host inside a Datacenter with the help of the 'Vmotion' Distributed Service [13] in the Cloud environment.

Acknowledgment. This work was supported by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

References

1. Xiong, K., Perros, H.: Service Performance and Analysis in Cloud Computing, pp. 693–700, $25.00 © 2009 IEEE (2009) 978-0-7695- 3708-5/09
2. Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I.: Virtual Infrastructure Management in Private and Hybrid Clouds, 1089-7801/09/$26.00 © 2009 IEEE (2009)
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A Berkeley View of Cloud computing. Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA (February 10, 2009)
4. Aymerich, F.M., Fenu, G., Surcis, S.: An Approach to a Cloud Computing Network, pp. 113–118. ©2008 IEEE (2008) 978-1-4244-2624- 9/08/$25.00
5. Lei, X., Zhe, X., Shaowu, M., Xiongyan, T.: Cloud Computing and Services Platform Construction of Telecom Operator. In: 2nd IEEE International Conference on Digital Object Identifier, Broadband Network & Multimedia Technology, IC-BNMT 2009, pp. 864–867 (2009)
6. Adhikari, M., Banerjee, S., Biswas, U.: Smart Task Assignment Model for Cloud Service Provider. Special Issue of International Journal of Computer Applications (0975 – 8887) on Advanced Computing and Communication Technologies for HPC Applications - ACCTHPCA (June 2012)
7. Buyya, R., Ranjan, R., Calheiro, R.N.: Modeling and Simulation of scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities
8. Parsa, S., Entezari-Maleki, R.: RASA: A New Grid Task Scheduling Algorithm. International Journal of Digital Content Technology and its Applications 3, 91–99 (2009)
9. Brucker, P.: Scheduling Algorithms, 5th edn. Springer Press (2007)
10. George Amalarethinam, D.I., Muthulakshmi, P.: An Overview of the scheduling policies and algorithms in Grid Computing. International Journal of Research and Reviews in Computer Science 2(2), 280–294 (2011)
11. El-kenawy, E.-S.T., El-Desoky, A.I., Al-rahamawy, M.F.: Extended Max-Min Scheduling Using Petri Net and Load Balancing. International Journal of Soft Computing and Engineering (IJSCE) 2(4), 2231–2307 (2012) ISSN: 2231-2307
12. Mohammad Khanli, L., Analoui, M.: Resource Scheduling in Desktop Grid by Grid-JQA. In: The 3rd International Conference on Grid and Pervasive Computing. IEEE (2008)
13. White Paper- VMware Infrastructure Architecture Overview, VMware
14. Yang, J., Khokhar, A., Sheikht, S., Ghafoor, A.: Estimating Execution Time For Parallel Tasks in Heterogeneous Processing (HP) Environment. 1994 IEEE (1994) 0-8186-5592-5194 $3.00 Q

15. Amalarethinam, D.I.G., Selvi, F.K.M.: A Minimum Makespan Grid Workflow Scheduling Algorithm. © 2012 IEEE (2012) 978-1-4577-1583-9/ 12/ $26.00
16. Belalem, G., Tayeb, F.Z., Zaoui, W.: Approaches to Improve the Resources Management in the Simulator CloudSim. In: Zhu, R., Zhang, Y., Liu, B., Liu, C. (eds.) ICICA 2010. LNCS, vol. 6377, pp. 189–196. Springer, Heidelberg (2010)
17. Bhatia, W., Buyya, R., Ranjan, R.: CloudAnalyst: A CloudSimbased Visual Modeller for Analysing Cloud Computing Environments and Applications. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446–452 (2010)
18. Calheiros, R.N., Ranjan, R., De Rose, C.A.F., Buyya, R.: CloudSim: A Novel Framework for modelling and Simulation of Cloud Computing Infrastructures and Services (2009)
19. Calheiros, R.N., Ranjan, R., De Rose, C.A.F., Buyya, R.: CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia (2009)