Implement an improved Datacenter Broker Policy (DCB) with Mapreduce Model

Enakshmi Nandi/Research Scholar
Dept. of Computer Science and Engineering
University of Kalyani
Kalyani, India

Debabrata Sarddar/Assistant Professor
Dept. of Computer Science and Engineering
University of Kalyani
Kalyani, India

Abstract— **Cloud computing offers different services as public or private or combined to users either free or on payment. Performance of Cloud is an important issue ,so that for maintaining quality of service (QoS) like efficient load balancing ,smart job allocation ,response time optimization, reduction of bandwidth, best Virtual Machine (VM) selection etc for reducing overall execution time of a cloudlet is a challenging factor in Cloud computing. Algorithms, policies, and methodologies are required to achieve high user satisfaction and real utilization in cloud computing by ensuring the efficient and fair allocation of every computing resource. When a new job arrives in cloud environments, the service broker is responsible for selecting the datacenter that will execute that job. Also selection of datacenters serves a valuable role for increasing the performance of a cloud environment. Datacenter Broker Policy (DCB) is required for binding cloudlet with VM and helps to reduce the overall execution time of cloudlet and make a system active and balanced. In this paper we proposed Mapreduce Model for effective allocation of cloudlet to VM which helps better make span of a system and maintain a proper load balancing.**

Keywords- Cloud Computing, Quality of Service (QoS), Datacenter Broker Policy, Execution time,Cloudsim

Introduction

Cloud computing is a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, and dynamically scalable managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet [1, 2]. For improving cloud services, large-scale software systems, such as social networking sites and e-commerce applications, are gaining popularity. These software systems mainly utilize cloud services to minimize costs and improve service quality to end users. Several factors affect the cost and quality of the cloud. Some of these factors are the distribution of the user bases, the availability of the web based infrastructure within those geographic areas [3], the dynamic nature of the usage patterns of the user base, and how well cloud services can be adjusted or dynamically reconfigured. The cloud service providers want to give the better service and performance than the software programs which were installed locally on end-user machines. Allocating cloudlet to VM in a heterogeneous cloud environment is a challenging matter. So formation of proficient environment, an efficient allocation policy is required. There are many cloudlet allocation policies [4] in cloud computing which are available to allocate the cloudlets to the various resources or VMs in an optimal way. In present study we proposed an improved Datacenter broker Algorithm which will allocate a cloudlet with VM and providing the correct VM with correct load for improving the Quality of Service (QoS) of the overall system. Here we consider Mapreduce model for proper allocation of cloudlet to VM. In remaining part of this paper we discuss our result and give a comparison with other existing policy and conclude our discussion with future research direction.

I. RELATED WORK

A. Cloudsim Toolkit

CloudSim Toolkit Several Grid simulators [5,6] such as GridSim, SimGrid, and GangSim are capable of modeling and simulating the grid application in a distributed environment but fails to support the infrastructure and application-level requirements creating from cloud computing paradigm [7]. The grid simulators are unable to separate the multi-layer service (Software as a Service, Platform as a Service, and Infrastructure as a Service) discrimination clearly as needed by cloud computing paradigm. In particular, there is negligible support on existing grid simulation toolkits for modeling of virtualization-enabled resource and application management environment [8].In cloud environment modeling and simulation toolkits must provide support for economic entities for enabling real-time trading of services between cloud users and cloud service providers. Cloudsim [8] is a cloud infrastructure modeling and simulation toolkit that supports real-time trading of services between users and providers. An open source CloudSim framework [9] as shown in Fig. 1 is developed on GridSim toolkit [10] offers support for economic-driven resource management and cloudlet scheduling, bandwidth management, cost management etc. The Cloudsim toolkit is customizable. So, it is suitable for the researchers to implement their own policies in the domain of cloud by extending relevant classes of Cloudsim toolkit. In present study object is to expanding CloudSim by utilizing the

broker policy. The data center Broker policy is a decision making procedure through which an individual can make the best link between cloudlets and VMs [11]. Some modules of CloudSim toolkit mentioned in Fig. 1 are relevant to our research as follows.

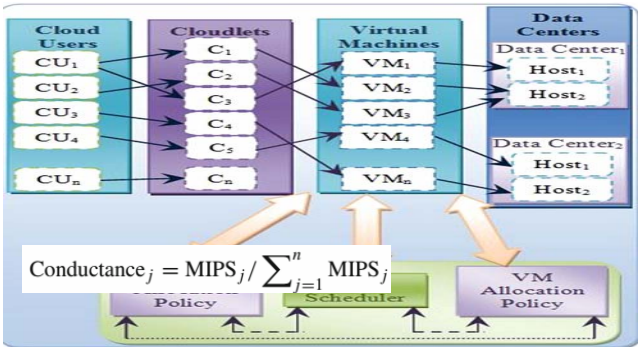

Fig1: CloudSim Work style

• *Cloud User (CU)* - The CU is an end user of cloud environment. They attained the service provided by cloud datacenter.
• *Data center (DC)* - Data center is composed of a set of hosts or physical nodes. It acts like an IaaS provider.

• *Cloudlet* -It is a container of tasks. A cloudlet is a request sent from CU for processing [12] to the DC.

• *Virtual machine (VM)* A virtual machine is a multi-user shared resource that provides heterogeneous service to all computer or network resources.

• *Host*- Host is a physical node, located inside every data center which may run on various virtual machine(s). The host contains Processing Element(s) or Core(s). In Cloudsim 3.0.3 [13], there is a class called data center Broker which remains several policies associated with several research issues like resource management, cloudlet scheduling, cost calculation and optimization etc. The researchers can apply their own logic and policies by extending few base classes. Some of the relevant policies are described below.

• *VM Scheduler* -This schedules the VM(s) to the Processing element or Core of a Host. It is required for allocating processing power to VMs [12, 14]. Some policies (space-shared, time-shared) are in-built in Cloudsim 3.0.3.
• *VM Allocation Policy*- VM Allocation Policy [12,14] is used to select available host in a data center which encounter the memory, storage, and availability requirement for a VM deployment. Some policies (such as, vm allocation simple) are in-built in Cloudsim 3.0.3. Researchers may apply their own policies by extending the base class.

• *Cloudlet Allocation Policy*- Cloudlet allocation policy [15] is used to select available VM(s) in a system. It admits cloudlet scheduling mechanism which schedules the cloudlet(s) to the VM(s).

B. Cloudlet Allocation Policy

The cloudlet allocation policy helps allocating a cloudlet with a virtual machine (VM) and decreases the completion time of the cloudlets as well as reduces the makespan of the VMs and the
host in the DC. The two basic existing cloudlet allocation policies are describe below:

• *Round Robin Allocation (RRA) Policy* [10] Round Robin Allocation Policy [14] binding the cloudlet to first available VM. For example, consider there are four cloudlets (CL1, CL2, CL3, CL4*)* and two VMs *(*VM1, VM2*)* present in the system. Table 1 shows the allocation pattern. According to this policy, cloudlet

Cloudlet	Virtual Machine (VM)
CL_1	VM_1
CL_2	VM_2
CL_3	VM_1
CL_4	VM_2

Table 1 Cloudlet binding with VM

CL1 allocated to VM1, CL2 allocated to VM2 and CL3 and CL4 allocated to the VM1 and VM2, accordingly.

• *Conductance algorithm (CA)* [17] -here each VM consider as a pipe. It calculates the Conductance (processing power) as per Eq. (1) of each VM as the ratio of its capacity to the sum of the capacity of all the VMs present in a System.

…… (1)

After the calculation of conductance, multiply the conductance of that particular VM with the length of the cloudlet list. To find out the strip length, Eq. (2) is used. It defines the number of cloudlets the VM can process.
Striplength $_j$ = Conductance $_j$ × *(*Length of the cloudlet list*)* … …… (2)

The existing policies do not think the procedure for determine the minimum makespan of the VM(s) as well as the host(s) in DC. To get over this problem, we propose a new cloudlet allocation policy which improves the makespan of the VMs as well as the host which will be discussed in Sect. 3.

II. MAPREDUCE MODEL

Mapreduce Model is widely recognized as the most important programming model for Cloud computing. It is a technique for dividing work across a distributed system. In MapReduce programming model, users have to define only two functions - a map and a reduce function. MapReduce is based on a very simple idea for parallel processing of data-intensive applications supporting arbitrarily divisible load sharing. First, split the data into blocks, assign each block to an instance/process and run these instances in parallel. Once all the instances have completed the computations assigned to them, start the second phase; merge the partial results produced by individual instances. MapReduce is a programming model inspired by the map and the reduce primitives of the Lisp programming language. It was consider for processing and generating large data sets on computing clusters [100]. As a result of the computation, a set of input < key, value > pairs is transformed into a set of output < key, value > pairs. Mapreduce philosophy is given below.

1) An application starts a master instance and M worker instances for the Map phase and later R worker instances for the Reduce phase.
2) The master partitions the input data in M segments.
3) Each map instance reads its input data segment and processers the data.
4) The results of the processing are stored on the local disks of the servers where the map instances run.
5) When all map instances have finished processing their data the R reduces instances read the results of the first phase and merges the partial results.
6) The final results are written by the reduce instances to a shared storage server.

The master instance monitors the reduce instances and when all of them report task completion the application is terminated as shown in the figure 2.

Call *M* and *R* the number of *Map* and *Reduce* tasks, respectively, and *N* the number of systems used by the *MapReduce*. When a user program invokes the *MapReduce function*, the following sequence of actions occured :

• The run-time library splits the input files into *M splits* of 16 to 64 MB each, identifies a number *N* of systems to run, and starts multiple copies of the program, one of the system being a *master* and the others *workers.* The master assigns to each idle system either a *map* or a *reduce* task. The master makes $O(M + R)$ scheduling decisions and keeps $O(M \times R)$ worker state vectors in memory. These considerations limit the size of *M* and *R*; at the same time, efficiency considerations require that $M, R >> N$.

• A worker being assigned a *Map* task reads the corresponding input split, parses < *key*, *value* > pairs and passes each pair to a user-defined *Map* function. The intermediate < *key*, *value* > pairs developed by the *Map* function are buffered in memory before being written to a local disk, partitioned into *R* regions by the partitioning function

• The locations of these buffered pairs on the local disk are passed back to the master, who is responsible for forwarding these locations to the reduce workers. A reduce worker uses remote procedure calls to read the buffered data from the local disks of the map workers; after reading all the intermediate data, it sorts it by the intermediate keys. For each unique intermediate key, the key and the corresponding set of intermediate values are passed to a user-defined *Reduce* function. The output of the *Reduce* function is appended to a final output file.

• When all Map and Reduce tasks have been completed, the master wakes up the user program. The system is fault tolerant; for each Map and Reduce task, the master stores the state (idle, in-progress, or completed) and the identity of the worker machine. The master pings each worker periodically and marks the worker as failed if it does not respond; a task in progress on a failed worker is reset to idle and becomes eligible for rescheduling. The master writes periodic checkpoints of its control data structures and, if the task fails, it can be restarted from the last checkpoint. To minimize network bandwidth the input data is stored on the local disks of each system.

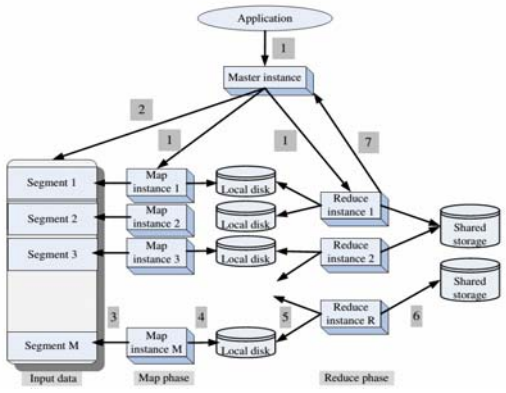

Fig 2: Mapreduce Philosophy

Identify applicable sponsor/s here. *(sponsors)*

III. PROPOSED METHOD

Step1>> Datacenter Broker shorted VM according to their capacity in ascending order and when Cloudlets are submitted to DCB module then DCB arranged them according to their length.

Step 2>> Now consider Mapreduce Model, so that higher MPs VM should be allotted more percentage of load than lower MIPS VMs. In this process load can be distributed properly.

Step 3>> An application starts a master instance and M worker instances for the Map phase and later R worker instances for the Reduce phase.

Step4>> The master partitions the input MIPS in M segments.

Step5>> Each map instance reads its input data segment and processers the MIPS instruction according to size of VM.

Step6>> The results of the processing Weighted value are stored on the local disks of the servers where the map instances run.

Step5>> When all map instances have finished processing their data the R reduces instances read the results of the first phase and merges the partial results.

Step 6>> The final results are written by the reduce instances to a shared storage server by multiplying the weighted value with length of cloudlet list.

Step7>> The master instance monitors the reduce instances and when all of them report task of allocation of batches of cloudlets to VMs have been completed and the application is terminated and all cloudlets are allocated with respective VMs proper. Here Proper load balancing has maintained.

IV. RESULT AND ANALYSIS

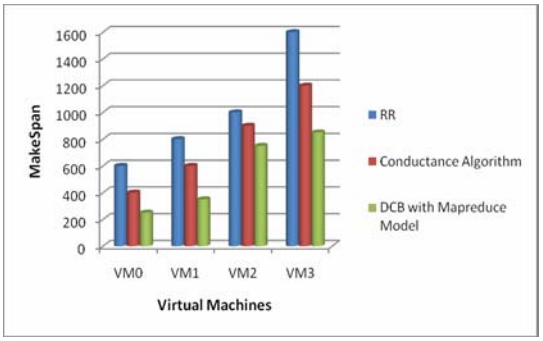

FIG3**:** COMPARISONS BETWEEN MAKESPAN RESULTS OF TWO EXISTING DCB POLICIES WITH OUR PROPOSED POLICY

Figure 3 shows that overall makespan of the system is 800.1 with comprising with others existing DCB policy and figure 4 shows the comparison between aforementioned allocation policies in terms of execution time. We can say from this figure that DCB policy with Mapreduce Model gives better performance with less execution time than others.

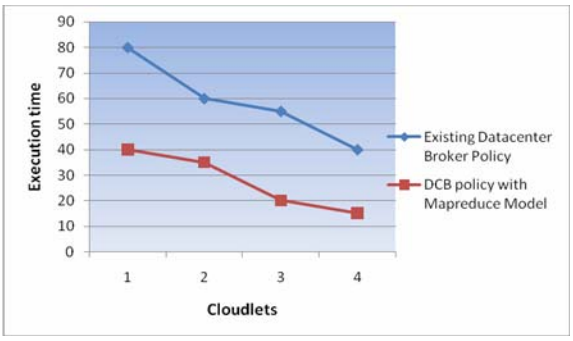

FIG 4: *COMPARISON OF EXECUTION TIME BETWEEN TWO DIFFERENT CLOUDLET ALLOCATION POLICIES*

V. CONCLUSION

Our proposed method emphasis on proper cloudlets allocation with different VMs inside a Datacenter module and give better make span of VMs and execution time of cloudlets also decreased with maintain proper load balancing by map and reduce phase of Mapreduce model. Here no VMs overloaded with high MIPS. Thus resource utilization and Quality of Service must be improved through our policy. We can conclude that our object to develop of DCB module using proper DCB policy to distribute loads intelligently for entire available VMs inside a Datacenter by keeping other VMs busy, so that the overall makespan will improve is massively successful. In future we investigate a better algorithm for live VM migration to other host inside a Datacenter with another intelligent algorithm in the Cloud environment.

ACKNOWLEDGMENT

We would like to thank our HOD Dr. kalyani Mali in department of Computer Science and Engineering in University of Kalyani for purpose of laboratory and university infrastructure.

REFERENCES

[1] Foster, I., et al. Cloud computing and grid computing 360-degree compared. in Grid Computing Environments Workshop, 2008. GCE'08. 2008. Ieee.

[2] Ahmed, M., et al., An advanced survey oncloud computing and state-of-the-art research issues. Int J Comput Sci Issues (IJCSI),2012. 9.

[3] Mohammed Radi, Faculty of Applied Science Alaqsa University, Gaza Palestine, Computer Science Department**,** "Efficient Service Broker Policy For Large-Scale Cloud Environments".

[4] Bhatia, W.; Buyy, 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).

[5] George Amalarethinam, D.I.; Muthulakshmi, P.: An overview ofthe scheduling policies and algorithms in Grid Computing. Int. J.Res. Rev. Comput. Sci. **2**(2), 280–294 (2011).

[6] 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.

[7] Ghalem, B.; Fatima Zohra, T.; Wieme, Z.: Approaches to improve the

resources management in the simulator CloudSim. In: ICICA 2010, LNCS 6377, pp. 189–196, (2010). doi:10.1007/ 978-3-642-16167-4_25
[8] Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.;Buyya, R.: CloudSim: a toolkit for modelling andsimulation of cloud computing environments and evaluation of resource provisioning algorithms. Published online 24 August 2010 in WileyOnline Library (wileyonlinelibrary.com). doi:10.1002/spe.995
[9] 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)
[10] Bhatia, W.; Buyy, R.; Ranjan, R.: CloudAnalyst: a CloudSim based 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).
[11] Varun Kumar Ojha, Mainak Adhikari, Sourav Banerjee, Vaclav SnaselDesign and Implementation of an Improved Datacenter Broker Policy to Improve the QoS of a Cloud, CHAPTER · JANUARY 2014 ,
DOI: 10.1007/978-3-319-08156-4_28, ResearchGate.
[12] Rawat, P.S.; Saroha, G.P.; Barthwal, V.: Quality of service evaluation of SaaS modeller (Cloudlet) running on virtual cloudcomputing environment using CloudSim. Int. J. Comput. Appl. **53**(13), 35–38(2012).
[13] Quansheng, G.; Jiwu, S.; Xiping, M.: Design and implementation of dynamic balance load based on LVS system. Comput. Res. Develop. **41**(16), 923–929 (2004)
[14] Parsa, S.; Entezari-Maleki, R.: RASA: a new grid task scheduling algorithm. Int. J. Digit. Content Technol. Appl. **3**, 91–99 (2009).
[15] Buyya, R.; Ranjan, R.; Calheiros, R.N.: Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: challenges and opportunities. In: Proceedings of the 7th High Performance Computing and Simulation Conference (HPCS 2009, ISBN: 978-1-4244-4907-1, IEEE Press, New York, USA), Leipzig, Germany, June 21–24, 2009.
[16] Chatterjee, T.; Ojha, V.K.; Adhikari, M.; Banerjee, S.; Biswas, U.; Snasel, V.: Design and Implementation of a new Datacenter Broker policy to improve the QoS of a Cloud. In: © Springer International Publishing Switzerland 2014, Proceedings of ICBIA 2014, Advances in Intelligent Systems and Computing, vol. 303, pp 281- 290 (2014). doi: 10.1007/978-3-319-08156-4_28.
[17] Dan C. Marinescu, Computer Science Division ,Department of Electrical Engineering & Computer Science University of Central Florida, Orlando, FL 32816, USA,Cloud Computing: Theory and Practice,December 6,2012.

AUTHORS PROFILE

Debabrata Sarddar is an Assistant Professor at the Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia, West Bengal, India. He completed his PhD from Jadavpur University. He did his M. Tech in Computer Science & Engineering from DAVV, Indore in 2006, and his B.E in Computer Science & Engineering from NIT, Durgapur in 2001. He has published more than 75 research papers in different journals and conferences. His research interests include Wireless and Mobile Systems and WSN, and Cloud computing.

Enakshmi Nandi received her M.Tech in VLSI and Microelectronics from Techno India, Salt Lake, West Bengal and B.Tech in Electronics and Communication Engineering from JIS College Of Engineering, West Bengal under West Bengal University of Technology, West Bengal, India. At present, she is Research scholar in Computer Science and Engineering from University of Kalyani. Her research interests include Cloud Computing, Mobile Communication system,
Device and Nanotechnology.