2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, T.N., India. 6 - 8 May 2015. pp.207-211.

Simulation of Cloud Infrastructure using CloudSim Simulator: A Practical Approach for Researchers

Pravesh Humane[1] and **Prof. J.N. Varshapriya**[2]

Computer Engineering Department, Veermata Jijabai Technological Institute (VJTI), Mumbai, India
Email: [1]pravesh.humane@gmail.com, [2]varshapriyajn@vjti.org.in

***Abstract*— Cloud Computing has evolved greatly in the recent years due to its characteristics of being secure, highly reliable, fault-tolerant, sustainable, and highly scalable; or we can say that it is capable of dynamically delivering IT resources over the Internet. Hence a lot of researchers and experts are working on improvising the cloud technology. To assess the performance of cloud environment, cloud simulators play an important role as it is a challenging task to perform experiments on the real cloud (which would incur huge cost in terms of currency if experiments are repeated). This paper gives a step by step practical approach towards simulating cloud components or cloud environment using the (open source) CloudSim simulator. It shows in-depth analysis and the factors responsible for simulating cloud concepts for your experiments or research or industry specific application.**

***Keywords*— Cloud Computing, Cloud simulator, CloudSim, Data Center, High Performance Computing**

I. INTRODUCTION

Cloud computing, often referred to as simply "the cloud", is the on-demand delivery of computing resources— everything from applications to data centers— over the Internet on pay-as-you-use basis. It is a new approach that reduces the complexity in IT by leveraging the efficient pooling of demand-based, self-controlled virtual infrastructure, consumed "as a service". It is the Internet-based computing, whereby shared hardware, software, and data is provided to the clients on demand, like the electricity grid.

Whether you are running applications that share photos to millions of mobile users or you're supporting the critical operations of your business, the "cloud" provides efficient access to flexible and low cost IT technologies. If you 'move to the cloud', you need not make large upfront investments in hardware or spend a lot of time managing that hardware. Alternatively, you can provision exactly the right type and size of computing infrastructure you need to power your newest bright idea or operate your IT unit. You may access as many resources as you need, right away, and pay only for what you use. Cloud Computing provides an easy way to access hardware, like the servers, databases, storage and more application services over the Internet. Cloud providers such as AWS own and maintain the network-connected hardware required for these application services, while you equip and use what you need via a web application [1].

The above reasons have led most of the organizations to adopt the cloud computing model thereby reducing operating costs and capital expenditure. Some of the common Cloud-based application services include web hosting, social networking, content delivery and real time data processing, which has different architecture, composition, and deployment requirements. Assessing the performance of provisioning and allocation policies in a real cloud environment involves overhead of huge costs. If a real HPC environment is used for benchmarking the performance under variable conditions, it is often forced by the austerity of the resources. Thus, it is difficult to perform such experiments in repeatable and scalable environments. To overcome this issue, cloud simulators are used. These tools advances the prospect of evaluating the assumptions in a contained environment where results could be revived. This approach offer compelling benefits to the IT industry by allowing the examination of their services in a controlled environment and experiment with different workload to develop an adaptive application provisioning techniques.

II. BACKGROUND

There have been many studies using simulation techniques to investigate behavior of large scale distributed systems and tools to support research on cloud. Some of these simulators are GridSim, MicroGrid, OptorSim, SimGrid and CloudSim. The first three focus on Grid computing systems. CloudSim is the only simulation framework for studying Cloud computing. Grid simulators have been used to

evaluate costs of executing distributed applications in Cloud infrastructures. SimGrid is a general framework to simulate distributed applications in Grid platforms. Among the presently available simulators, only GridSim supports economic-driven resource management and application scheduling [2].

III. THE CLOUDSIM SIMULATOR

CloudSim is invented as CloudBus Project at the University of Melbourne, Australia and supports system design and modeling of cloud components such as data centers, virtual machines (VMs) and provisioning strategies. It implements generic application provisioning methods that can be customized with ease and confined efforts.

CloudSim allows the researchers to focus on specific system design issues needlessly being concerned about the low level details associated with cloud-based infrastructures and services. It is a java-based toolkit; and it includes certain actors or entities which are nothing but java classes which communicate with each other during the simulation process [3].

The following are the common entities or actors which perform actions:

1. *Virtual Machine (VM)*— Logical machine on which applications will be run.
2. *VM Generator*— This generates the VMs in the cloudsim simulator and submits all the VMs to the broker.
3. *Datacenter (DC)*— This provisions the resources, and may have one or more than one host.
4. *Host*— Physical machine on which logical machines are placed.
5. *Cloud Information Service (CIS)*— An entity which manages all the registering of resources.
6. *Broker*— Broker is an agent, who will submit the VMs in specific host and then bind cloudlets, applications, processes on specific VMs and after execution of all the cloudlets, the free VMs will be automatically destroyed.
7. *Datacenter Broker*— An agent who is responsible for communication between Datacenter and submission of Cloudlets and VMs.
8. *Cloudlets*—- Cloudlets are the processes or applications that run on VMs.
9. *Cloudlet Generator*— Cloudlet generator will generate the cloudlets and submit the cloudlet list to the broker.
10. *BWProvisioner*— This is an abstract class that clones the provisioning policy of bandwidth to VMs that are deployed on a Host component.
11. *MemoryProvisioner*— This is an abstract class that represents the provisioning policy for allocating memory to VMs.
12. *VMAllocationPolicy*— This is an abstract class implemented by a Host component that models the policies (which may be space-shared or time-shared) required for allocating processing power to VMs.

You will find these entities as the java classes in the Cloudsim API (docs) [4]. All these classes have corresponding methods which make your simulation process easier.

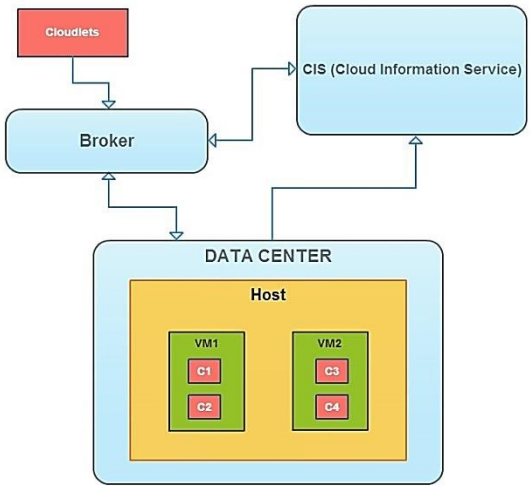

Fig. 1. CloudSim Model

IV. SIMULATION MODEL

This is the CloudSim framework or model in Fig. 1. It has the entities- Cloud Information Service (CIS), Data Center, Broker, Virtual Machines (VM1 & VM2), Host and Cloudlets (C1, C2, C3, C4). The CIS is like the registry containing all the resources in the cloud. All the resources in cloud are required to be registered to the CIS before they could be used. The Data Center contains one or more Hosts (only one in this case). The Hosts has the following configuration parameters- Processing Elements (PEs), RAM, Bandwidth (Bw) etc. The Virtual Machines will always reside inside the Host as this is a virtualized environment. The Cloudlets run on the VMs. Broker is an entity of 'DatacenterBroker' class. The Broker is responsible to submit tasks to the Datacenter. The broker interacts with the CIS and requests information of registered resources. Then CIS returns the characteristics of the

Datacenter to the broker. The broker has Cloudlets (applications) which are yet to be allocated. Once the broker gets all the information from CIS, it submits Cloudlet List to the Datacenter. On submission, the cloudlets are allocated to the VMs running on the Hosts of the Datacenter. The allocation decision is taken by the default allocation policies of the cloudsim (like CloudletScheduler, VmAllocationPolicy VmScheduler etc.). The Hosts (which reside in Datacenter) schedules the VMs and the VMs (which reside in Host) schedules the Cloudlets.

V. STEPS INVOLVED IN SIMULATION

This will make you understand the basics of simulation process.

1. Cloudsim is configured in any of the java IDE (Eclipse, Netbeans etc.). A java class is created where you will write the code for your cloudsim program.
2. First, the number of users that are present in your simulation are set. Given by the "broker count".
3. The CloudSim library is initialized by calling the function-

   CloudSim.init(num_user, calendar, trace_flag). This calls a method which initialize common variables- initCommonVariable(cal, traceFlag, numUser)
4. Object of CIS (Cloud Information Service) is created using- new CloudInformationService(<name>).
5. Next, data center is created using 'createDatacenter' method. This will also create Hosts; and their characteristics like the PEs, RAM, Bandwidth (Bw) utilization etc. are configured.
6. To create HostList and PeList:
7. *List<Host> hostList = new ArrayList<Host>();
8. *List<Pe> peList = new ArrayList<Pe>();
9. *hostId=0, ram=2048, bw=10000, storage=1000000
10. To add the HostList and PeList to the list of machines:
11. *peList.add (new Pe(0,new PeProvisionerSimple (mips)))
12. *hostList.add(new Host(hostId, new RamProvisionerSimple(ram), new BwProvisionerSimple(bw), storage, peList, new VmSchedulerTimeShared(peList)), these methods are called.
13. Finally the Datacenter object is created:
14. *datacenter = new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList), storageList, 0)
15. Make sure all the parameters are passed correctly. The architecture (x86 or x64), OS (Linux, Windows etc.), Storage (eg. SAN) and VMM (Xen, KVM, Vmware etc.) are also configured here.
16. Thus the name, characteristics and allocation policies have been defined for the data center.
17. An instance of 'DatacenterBroker' is created by calling-
18. broker = new DatacenterBroker("Broker"). This deals with the communication between datacenter and submission of Cloudlets and VMs. This is an important step; your simulation will not run if this is skipped.
19. Now Virtual Machine instances are created using-
20. *Vm <vmName> = new Vm(<characteristics>, <scheduling policy>). Their characteristics like vmId, mips, size, PEnos, RAM, VMM, Bw utilization, etc. are configured.
21. The list of virtual machines "VmList" is added and submitted to the broker by calling-
22. *broker.submitVmList(***vmlist***)
23. The elemental infrastructure for cloud is now completed at this step.
24. Now, the cloudlets (applications or processes) are created with parameters like Id, length, PesNo, fileSize, mips (million instructions per second), Bw, outputSize, utilizationModel etc. by calling-
25. *Cloudlet <name> = new Cloudlet(<characteristics>)
26. The list of cloudlets "CloudletList" is submitted to the broker by- *broker.submitCloudletList(***cloudletList***)
27. The simulation process is now started by calling-
28. *CloudSim.***startSimulation**()
29. Now all the functions are invoked and simulation events are triggered.
30. And then the simulation process is stopped by calling-
31. *CloudSim.***stopSimulation**()
32. Here all the entities are shut down.
33. Finally the status (output) of the simulation is printed by calling-
34. ***printCloudletList**(newList)

This is how the simulation works.

VI. CLOUDSIM EVENTS

The CloudSim Events unfolds the working of simulation process which runs in the background of the code. Fig. 2 shows the flow of events. It again has three entities- CIS, Datacenter and Broker which communicate with each other during simulation in an abstract way. First, the Datacenter sends CIS a request to register its resources so that the CIS is aware about the total availability of resources. The Broker has Cloudlets waiting to be executed; so it requests the CIS asking for the list of resources so that it can execute the cloudlets. The datacenter is notified and it sends the datacenter characteristics to the broker. Now the broker sends a request to the datacenter to create VM. Once the VMs are created, datacenter sends an Acknowledgment (ACK) to the broker. On receiving the ACK, broker submits the Cloudlet List to the datacenter for execution. Datacenter allocates these cloudlets on the recently created VMs and once they are done with the execution, it sends a message to the broker that cloudlets have executed. This is how the sequence of events take place. All of these is taken care by the 'Cloudsim.java' class. Few examples of such events are- ENULL, SEND, HOLD_DONE, CREATE etc.

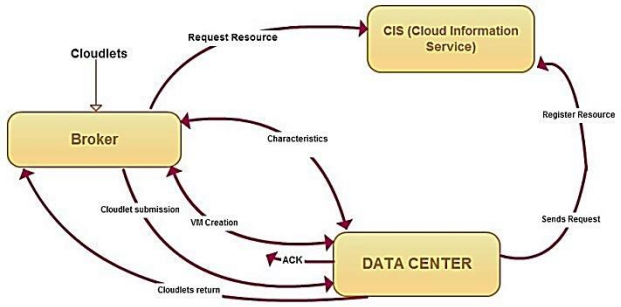

Fig. 2. CloudSim Events

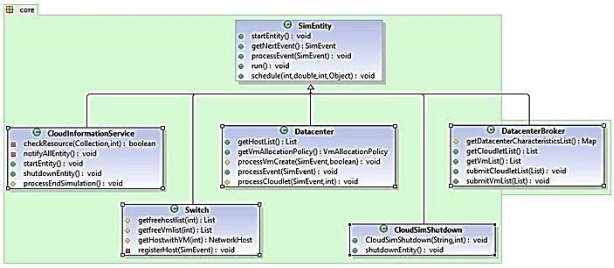

Fig. 3. Class Design Diagram for CloudSim

This (Fig. 3) is the class diagram for cloudsim considering only the major classes. The 'SimEntity' is the parent class, and the CIS, Datacenter, DatacenterBroker, Switch and CloudSim Shutdown are all inherited from SimEntity. These classes have corresponding methods which represents the actions performed by them.

VII. SIMULATION FLOW

When the StartSimulation() method of 'CloudSim.java' class is called, a series of events are triggered. The simulation flow of cloudsim goes through two phases. In the Phase I, the events are generated and then they are added to 'Future Queue'. In Phase II, the events which are added to future queue are moved to the 'Deferred Queue'. We need these queues because in a dynamic environment, there might arise a need to create new components during runtime; so the list of components are maintained in future queue. Once all the set of components are added to future queue (waiting queue), they are moved to the deferred queue one by one. The deferred queue is like the ready queue, so all the events in deferred queue gets processed sequentially. So the use of these queues allows cloudsim to possess dynamic nature.

VIII. EXPERIMENTS AND EVALUATION

In this section, we present experiments and evaluation that were undertaken to measure the efficiency of simulating cloud infrastructure using cloudsim. The experiments were conducted on an Intel Core 2 Duo machine having configuration 2.2GHz with 1MB of L2 cache and 4 GB of RAM running Windows 8.1 Pro 64bit and JDK 1.7. We created a data center, a broker and a user for performing tests. The number of Hosts were increased gradually from 100. We analyzed the time required for simulation and amount of memory required for resources instantiation as the number of hosts in the data center is increased.

From the Figures 4 and 5, we can see that the time grows exponentially with the number of hosts/machines and the memory consumption grows linearly with the number of hosts.

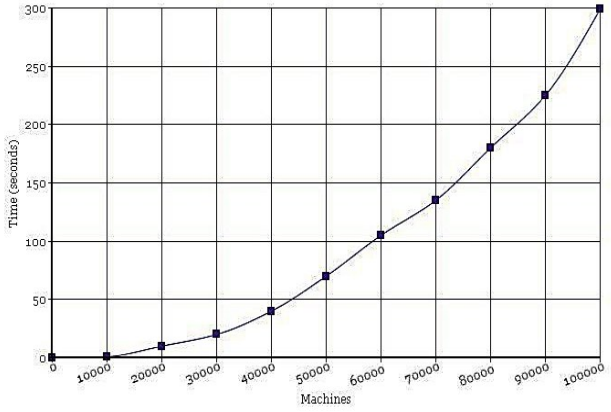

Fig. 4. Time for Simulation of Machines

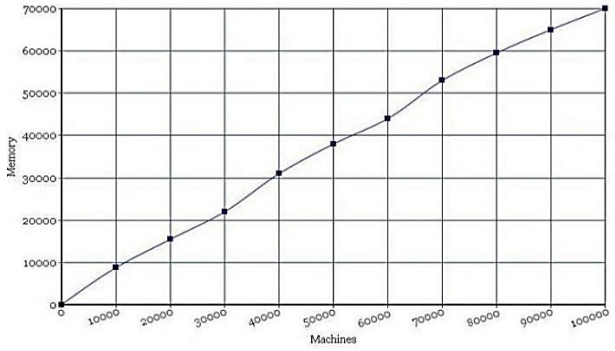

Fig. 5. Memory Required for Resources Instantiation

IX. CONCLUSION AND FUTURE WORK

The practical approach discussed in this paper has helped us in carrying out a successful simulation of cloud infrastructure and therefore, we can conclude that this approach can be used not only for basic data center simulation but also for research or customized applications or experiments.

The CloudSim tool being an open source project, provides generic (eg. Abstract) classes and features that can be broadly used, and the users may develop case-specific behavior according to their own needs. The cloudsim has various abstract provisioning classes such as- Vm Allocation Policy, Bandwidth Provisioner, Vm Scheduler, Cloudlet Scheduler, Power Vm Allocation Policy, Memory Provisioner etc. These classes can be overridden with ease according to specific application or research by defining your own implementation of the abstract methods. These policies are 'extended' and then the new improved strategy can thus be implemented; later integrating it into the cloudsim source package. This will thus improvise the cloud simulation technology and serve a vast range of users working in the domain of data centers. So the future scope includes modifying the abstract default policies; which would conclusively result in the addition of new features to this open source cloudsim project.

REFERENCES

[1] http://aws.amazon.com/
[2] Dr. Rahul Malhotra, Prince Jain "Study and Comparison of Various Cloud Simulators Available in the Cloud Computing", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 9, September 2013
[3] http://www.cloudbus.org/cloudsim/
[4] http://www.cloudbus.org/cloudsim/doc/api/
[5] Rajkumar Buyya, Rajiv Ranjan, Rodrigo N. Calheiros "Modeling and Simulation of Scalable Cloud ComputingEnvironments and the CloudSim Toolkit: Challenges and Opportunities", High Performance Computing & Simulation, 2009. HPCS '09. International Conference on
[6] http://code.google.com/p/cloudsim/
[7] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, Rajkumar Buyya "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms" Wiley Online Library, 24 August 2010
[8] Tarun Goyal, Ajit Singh, Aakanksha Agrawal "Cloudsim: simulator for cloud computing infrastructure and modeling", International Conference on modeling, optimization and computing (ICMOC-2012)
[9] R. Buyya. The cloudsim toolkit version 3.0.3, http://code.google.com/p/cloudsim