Group A Assignment No. 1(a)

Title : Install and use CloudSim

Aim : Install a CloudSim simulator tool and write a program to show the simulation output.

Objective : To study different components comprising a cloud and how to programmatically configure datacenter, VM and cloudlet as per the user requirements.

Theory : CloudSim is a new, generalized, and extensible simulation framework that allows modeling, simulation, and experimentation of emerging Cloud computing infrastructures and application services. By using CloudSim, researchers and industry-based developers can test the performance of a newly developed application service in a controlled and easy to set-up environment.

CloudSim library is used for the following operations:-

Support for modeling and simulation of virtualized server, hosts with CloudSim able policies for provisioning host resources to VM's

- Support for simulation of energy –aware computational resources
- Support for modeling and simulation of data center network topologies and message passing applications.
- Support for federated clouds.
- Support for dynamic insertion of simulation elements, and support for stopping and resuming simulation.
- Support for user defined policies to allot hosts to VMs, as well as allotting host resources to VMs.

Architecture of CloudSim :

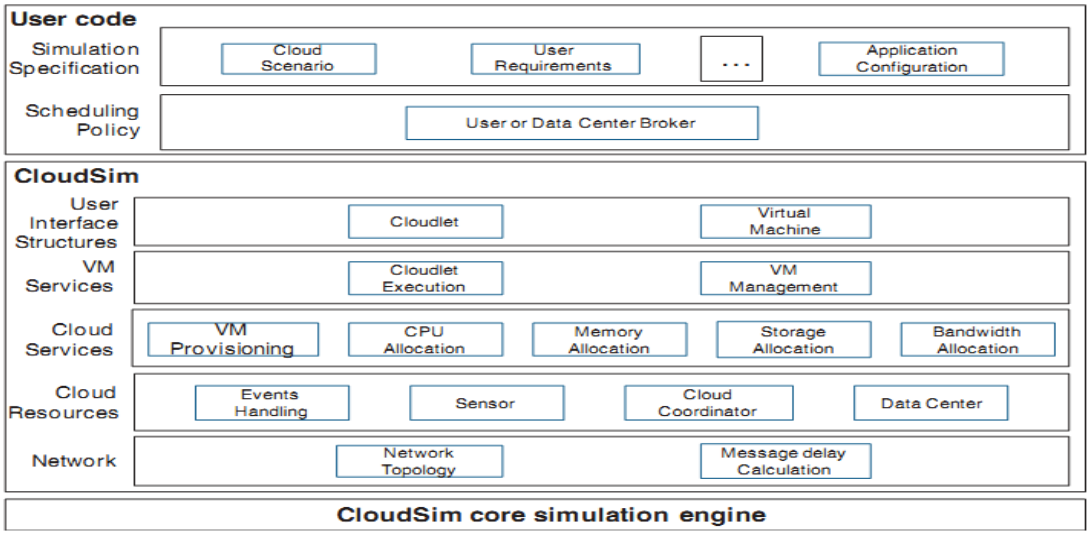


Figure 3. Layered CloudSim architecture.

The infrastructure-level services (IaaS) related to the clouds can be simulated by extending the data center entity of CloudSim. The data center entity manages a number of host entities. The hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the Cloud service provider. Here, the VM policy stands for the operations control policies related to VM life cycle such as: provisioning of a host to a VM, VM creation, VM destruction, and VM migration. Similarly, one or more application services can be provisioned within a single VM instance, referred to as application provisioning in the context of Cloud computing. In the context of CloudSim, an entity is an instance of a component. A CloudSim component can be a class (abstract or complete) or set of classes that represent one CloudSim model (data center, host).

A data center can manage several hosts that in turn manage VMs during their life cycles. Host is a CloudSim component that represents a physical computing server in a Cloud: it is assigned a pre-configured processing capability (expressed in millions of instructions per second—MIPS), memory, storage, and a provisioning policy for allocating processing cores to VMs. The Host component implements interfaces that support modeling and simulation of both single-core and multi-core nodes.

VM allocation (provisioning) is the process of creating VM instances on hosts that match the critical characteristics (storage, memory), configurations (software environment), and requirements (availability zone) of the SaaS provider. CloudSim supports the development of custom application service models that can be deployed within a VM instance and its users are required to extend the core Cloudlet object for implementing their application services. Furthermore, CloudSim does not enforce any limitation on the service models or provisioning techniques that developers want to implement and perform tests with. Once an

application service is defined and modeled, it is assigned to one or more pre-instantiated VMs through a service-specific allocation policy. Allocation of application-specific VMs to hosts in a Cloud-based data center is the responsibility of a VM Allocation controller component (called VmAllocationPolicy). This component exposes a number of custom methods for researchers and developers who aid in the implementation of new policies based on optimization goals (user centric, system centric, or both). By default, VmAllocationPolicy implements a straightforward policy that allocates VMs to the Host on a First-Come-First-Serve (FCFS) basis. Hardware requirements, such as the number of processing cores, memory, and storage, form the basis for such provisioning. Other policies, including the ones likely to be expressed by Cloud providers, can also be easily simulated and modeled in CloudSim. However, policies used by public Cloud providers (Amazon EC2, Microsoft Azure) are not publicly available, and thus a pre-implemented version of these algorithms is not provided with CloudSim.

For each Host component, the allocation of processing cores to VMs is done based on a host allocation policy. This policy takes into account several hardware characteristics, such as number of CPU cores, CPU share, and amount of memory (physical and secondary), that are allocated to a given VM instance. Hence, CloudSim supports simulation scenarios that assign specific CPU cores to specific VMs (a space-shared policy), dynamically distribute the capacity of a core among VMs (time-shared policy), or assign cores to VMs on demand.

Each host component also instantiates a VM scheduler component , which can either implement the space-shared or the time-shared policy for allocating cores to VMs. Cloud system/application developers and researchers can further extend the VM scheduler component for experimenting with custom allocation policies. In the next section, the finer-level details related to the time- shared and space-shared policies are described. Fundamental software and hardware configuration parameters related to VMs are defined in the VM class. Currently, it supports modeling of several VM configurations offered by Cloud providers such as the Amazon EC2.

Drawback of CloudSim :

Major Limitation is GUI, but instead of this it is used in universities and industry for implementation of cloud based algorithms.

Packages in CloudSim library :

org.cloudbus.cloudsim

org.cloudbus.cloudsim.core

org.cloudbus.cloudsim.core.predicates

org.cloudbus.cloudsim.distributions

org.cloudbus.cloudsim.lists

org.cloudbus.cloudsim.network

org.cloudbus.cloudsim.network.datacenter

org.cloudbus.cloudsim.power

org.cloudbus.cloudsim.power.lists

org.cloudbus.cloudsim.power.models

org.cloudbus.cloudsim.provisioners

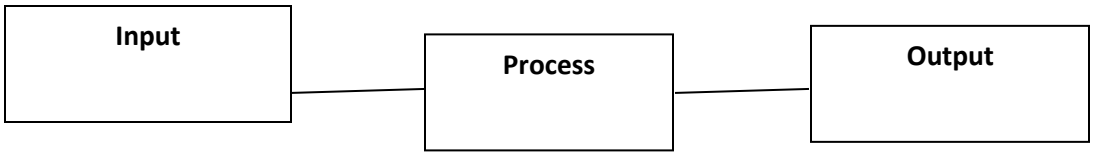
org.cloudbus.cloudsim.util

Out of these packages we require first one to execute all basic programs of CloudSim.

Steps :

1. Download cloudsim and extract the zip folder to cloudsim.
2. Create a java project in eclipse.
3. Right click on java project, go to properties -> java build path -> Libraries-> add external jar files. Browse through cloudsim/lib/cloudsim3.0.jar.
4. Execute CloudSimExample1.java in the folder : "cloudsim3.0/examples/org/cloudbus/cloudsim/ examples/"

Mathematical Modelling:



Let's S is the system such that,

S = { I, O, Fn, Sc, Fc }

Where,

I = { I1, I2,…..In } : Set of i/o

O = { O1, O2,……On} : Set of o/p

Fn = { Fn1, Fn2, …..Fnn} : Set of functions

Sc = { Sc1, Sc2, …..Scn} : Set of success cases

Fc = { Fc1, Fc2, …..Fcn} : Set of failure cases

I : Set of Inputs

I1 : Data Centers

I2 : Broker

I3 : Virtual Machine

I4 : Cloudlet

O : Set of Output

O1: Time

O2: Start Time

O3: Finish Time

O4: Simulation Results

Fn : Set of Functions

Fn1: Create Data Center

Fn2: Create Broker

Fn3: Print Cloudlet list

Sc : Success Cases

Sc1: Data Center Created

Sc2: Broker Created

Sc3: Virtual Machine Created

Sc4: Cloudlet Created

Sc5: Output Printed Created

Fc : Failure Cases

Fc1: Output not printed

Fc2: Data Center creation fails

Fc3: VM and cloudlet creation fail

Fc4: VM list and Cloudlet list not submitted to broker

Fc5: Broker fails

Program Explanation :

CloudSimExample1.java is a simple example showing how to create a datacenter with one host and run one cloudlet on it.

Following are the list of packages used in CloudSimExample1.java

org.cloudbus.cloudsim.Cloudlet - Cloudlet is an extension to the cloudlet.

org.cloudbus.cloudsim.CloudletSchedulerTimeShared - CloudletSchedulerTimeShared implements a policy of scheduling performed by a virtual machine.

org.cloudbus.cloudsim.Datacenter - Datacenter class is a CloudResource whose hostList are virtualized.

org.cloudbus.cloudsim.DatacenterBroker - DatacentreBroker represents a broker acting on behalf of a user.

org.cloudbus.cloudsim.DatacenterCharacteristics - DatacenterCharacteristics represents static properties of a resource such as resource architecture, Operating System (OS), management policy (time- or space-shared), cost and time zone at which the resource is located along resource configuration.

org.cloudbus.cloudsim.Host - Host executes actions related to management of virtual machines (e.g., creation and destruction).

org.cloudbus.cloudsim.Pe - CloudSim Pe (Processing Element) class represents CPU unit, defined in terms of Millions Instructions Per Second (MIPS) rating. ASSUMPTION: All PEs under the same Machine have the same MIPS rating.

org.cloudbus.cloudsim.Storage - An interface which defines the desired functionality of a storage system in a Data Cloud.

org.cloudbus.cloudsim.UtilizationModel - The UtilizationModel interface needs to be implemented in order to provide a fine-grained control over resource usage by a Cloudlet.

org.cloudbus.cloudsim.Vm - Vm represents a VM: it runs inside a Host, sharing hostList with other VMs.

org.cloudbus.cloudsim.VmAllocationPolicy - VmAllocationPolicy is an abstract class that represents the provisioning policy of hosts to virtual machines in a Datacentre.

Algorithmic Steps

1. Initialize the CloudSim package. It should be called before creating any entities.

Initialize the CloudSim library

2. Create Datacenters. Datacenters are the resource providers in CloudSim. We need at list one of them to run a CloudSim simulation
3. Create Broker
4. Create one virtual machine

   Virtual Machine description

   ```
   int vmid = 0;
   int mips = 1000;
   long size = 10000; // image size (MB)
   int ram = 512; // vm memory (MB)
   long bw = 1000;
   int pesNumber = 1; // number of cpus
   String vmm = "Xen"; // VMM name
   ```

   We create the VM then addt his VM to vmlist and thensubmit that vmlist to broker

5. Create one Cloudlet(Task)
   Cloudlet properties

   ```
   int id = 0;
   long length = 400000;
   long fileSize = 300;
   long outputSize = 300;
   ```

   We create cloudlet,add it to the cloudlet list and then submit that list to broker.

6. Starts the simulation
7. Print results when simulation is over.

Input :

1. Configuration parameters for Virtual Machine such as such as ramsize, hypervisor name, number of CPUs
2. Characteristics of a data center such as architecture, OS, list of Machines, allocation policy: time- or space-shared, time zone
3. Cloudlet properties such as length, fileSize and outputSize.

Output : output of simulation

Platform : Ubuntu 12.04 , Eclipse Java Jdk1.7

Conclusion : The cloudsim is installed and reports are generated for given scenario.

FAQ's

1. How CloudSim is different than actual cloud?

2. What are benefits of CloudSim?
3. What are other simulators to simulate cloud environment?
4. What is cloudlet?