Simulation of Cloud Computing Environment using CloudSim

Pankaj Sareen
Assistant Professor, SPN College Mukerian, India.

Dr. Tripat Deep Singh
Assistant Professor, GNIMT Ludhiana, India.

Abstract - Cloud computing is a recent advancement wherein IT infrastructure and applications are provided as 'services' to end users under a usage-based payment model. Analyzing and testing different scheduling and allocation algorithms for the development of these applications on a real cloud environment is really challenging because most of the cloud applications show changing incoming requests and moreover testing algorithms on a real cloud can cost us a lot. For testing the effectiveness of a particular policy that is to be implemented on a cloud we need a simulation environment that can provide us an environment that is close to the actual cloud, and can generate results that can help us in the analysis of the policies so that we can deploy them on actual Clouds. The CloudSim toolkit supports both system and behavior modeling of Cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies. It implements generic application provisioning techniques that can be extended with ease and limited effort. Currently, it supports modeling and simulation of Cloud computing environments and also it exposes custom interfaces for implementing policies and provisioning techniques for allocation of VMs under Cloud computing scenarios. Several researchers are using CloudSim in their investigation. This paper defines CloudSim and then explores it's all variants available in CloudSim such as CloudAnalyst, GreenCloud, Network CloudSim, EMUSIM and MDCSim. Comparison of all CloudSim Variant with respect to networking, platform and language is also made in this paper. This paper highlights the brief introduction and working of CloudSim. Further this work focuses about important parameters which are required to include in real life cloud based application. This paper also talks about working of CloudSim and how to implement cloud infrastructure in CloudSim with example.

Index Terms - Cloud Computing, CloudSim, Cloudlet, Data center, Simulation, CloudAnalyst, MDCSim, GreenCloud, Virtual Machine, Provisioning Policies, VM Placement, EMUSIM, CloudCoordinator, Predicates, CIS

1. INTRODUCTION

Cloud computing delivers infrastructure, platform, and software as services, which are made available as subscription-based services in a pay-as-you-go model to users and in recent years, it is the biggest issue in IT fields.

Industries such as Amazon, Google, Microsoft, HP and IBM have heavily invested on it. Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services [1]. In Cloud Computing Case, The Simulation Tools gives or offers significant benefits to the Customers and Providers. For Customers, It allows them to test their services in controllable environment with free of cost and to check the performance before publishing to the real clouds. Meanwhile for Providers, allow them to check the kinds of leasing according to various prices and load. In addition, this will lead to optimize the resources access cost with improving the profits. Without these tools, both of the Customers and Providers must rely on imprecise evaluations, or on try-and-error approaches, these approaches may lead to inefficient services performance and reduce revenue generation. In addition, Simulators helps researchers and industry-based developers to test the performance of a developed application service in a suitable and easy to setup environment. In the absence of such simulation platforms, Cloud customers and providers have to rely either on theoretical and imprecise evaluations, or on try-and-error approaches that lead to inefficient service performance and revenue generation. These tools open up the possibility of evaluating the hypothesis in a controlled environment where one can easily reproduce results.

1.1 Existing Simulation Toolkits

The various Cloud Simulation Toolkits are [2]:

1.1.1. CloudAnalyst

CloudAnalyst was derived from CloudSim and extends some of its capabilities and features proposed [3]. CloudAnalyst separates the simulation experimentation exercise from a programming exercise. It also enables a modeler to repeatedly perform simulations and to conduct a series of simulation experiments with slight parameters variations in a quick and easy manner. CloudAnalyst can be applied to examining behavior of large scaled Internet application in a cloud environment.

1.1.2. GreenCloud

GreenCloud is a CloudSim that have green cloud computing approach with confidently, painlessly, and successfully. In other words, GreenCloud is developed as an advanced packet level cloud network simulator with concentration on cloud communication [4]. GreenCloud extracts aggregates and makes fine grained information about the energy consumed by computing and communication elements of the data center equipment such as computing servers, network switches and communication links.

GreenCloud Aim is to develop high-end computing systems such as Clusters, Data Centers, and Clouds that allocate resources to applications hosting Internet services to meet users' quality of service requirements and to minimize consumption of electric power by improving power management, dynamically managing and configuring power-aware ability of system devices.

GreenCloud can reduce Data Center Power Consumption by workload consolidation via DC virtualization and by improving sustainability by reducing host count.

1.1.3. Network CloudSim

Network CloudSim is an extension of CloudSim as a simulation framework which supports generalized applications such as high performance computing applications, workflows and e-commerce [5]. Network CloudSim uses Network Topology class which implements network layer in CloudSim, reads a BRITE file and generates a topological network. In network CloudSim, the topology file contains nodes, number of entities in the simulation which allows users to increase scale of simulation without changing the topology file. Each CloudSim entity must be mapped to one BRITE node to allow proper work of the network simulation. Each BRITE node can be mapped to only one entity at a time. Network CloudSim allows for modeling of Cloud data centers utilizing bandwidth sharing and latencies to enable scalable and fast simulations. Network CloudSim structure supports designing of the real Cloud data centers and mapping different strategies. Information of network CloudSim is used to simulate latency in network traffic of CloudSim.

1.1.4. EMUSIM

EMUSIM is an integrated architecture to anticipate service's behavior on cloud platforms to a higher standard. EMUSIM combines emulation and simulation to extract information automatically from the application behavior via emulation and uses this information to generate the corresponding simulation model. Such a simulation model is then used to build a simulated scenario that is closer to the actual target production environment in application computing resources and request patterns. Information that is typically not disclosed by platform owners, such as location of virtual machines and number of virtual machines per host in a given time, is not required by EMUSIM. EMUSIM is built on top of two software systems: Automated Emulation Framework (AEF) for emulation and CloudSim for simulation [6].

1.1.5. MDCSim

MDCSim is a commercial discrete event simulator developed at the Pennsylvania State University. It helps the analyzer to model unique hardware characteristics of different components of a data center such as servers, communication links and switches which are collected from different dealers and allows estimation of power consumption. MDCSim is the most prominent tool to be used as it has low simulation overhead and moreover its network package maintains a data center topology in the form of directed graph [5].

The comparison of these Simulation toolkits is shown in table 1:

Various CloudSim	**Platform**	**Programming Language**	**Networking**	**Simulator Type**	**Availability**
CloudAnalyst	CloudSim	Java	Limited	Event Based	Open Source
GreenCloud	NS2	C++/OTCL	Full	Packet Level	Open Source
Network CloudSim	CloudSim	Java	Full	Packet Level	Open Source
EMUSIM	AEF	Java	Limited	Event Based	Open Source
MDC SIM	CSIM	C++/Java	Limited	Event Based	Commercial

Table 1: Comparison of Various CloudSim

2. CLOUDSIM OVERVIEW

CloudSim [7, 8] is toolkit for simulation of cloud computing written in Java languages. CloudSim is built in CLOUDS (Cloud Computing and Distributed System) Laboratory by the University of Melbourne, Australia. It is developed in java platform including the pre developed modules such as SimJava and GridSim. It is customizable tool [9]; it allows extension and definition of policies in all the components of the software stack, which makes it suitable as a research tool that can handle the complexities arising from simulated environments. It allows fast evaluation of scheduling and resource allocation mechanisms within cloud data centers which are sometimes are not easy to access. CloudSim performs system modeling for Cloud system components, such as datacenter and VM, and behavioral modeling as resource provisioning policy and can analyze performance of them. CloudSim, however, does not consider failure mode of Cloud resources such as VM, host, datacenter, etc. There are many advantages of using CloudSim for initial performance testing like:

- Time effectiveness: it takes very less effort and time to implement Cloud-based applications.
- Flexibility: developers can easily model and test the performance of their applications and its services in heterogeneous environments (Microsoft Azure, Amazon EC2).

2.1 Features of CloudSim

CloudSim can help to overcome the Cloud computing challenges by providing many features [10] like:

- support for modeling and simulation of large scale Cloud computing data centers
- support for modeling and simulation of virtualized server hosts, with customizable policies for provisioning host resources to virtual machines
- support for modeling and simulation of application containers
- support for modeling and simulation of energy-aware computational resources
- support for modeling and simulation of data center network topologies and message-passing applications
- support for modeling and simulation of federated clouds
- support for dynamic insertion of simulation elements, stop and resume of simulation
- support for user-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines

2.2 CloudSim Architecture

CloudSim simulator has a multi-layered structure which consists of three layers [11] as shown in Fig. 1

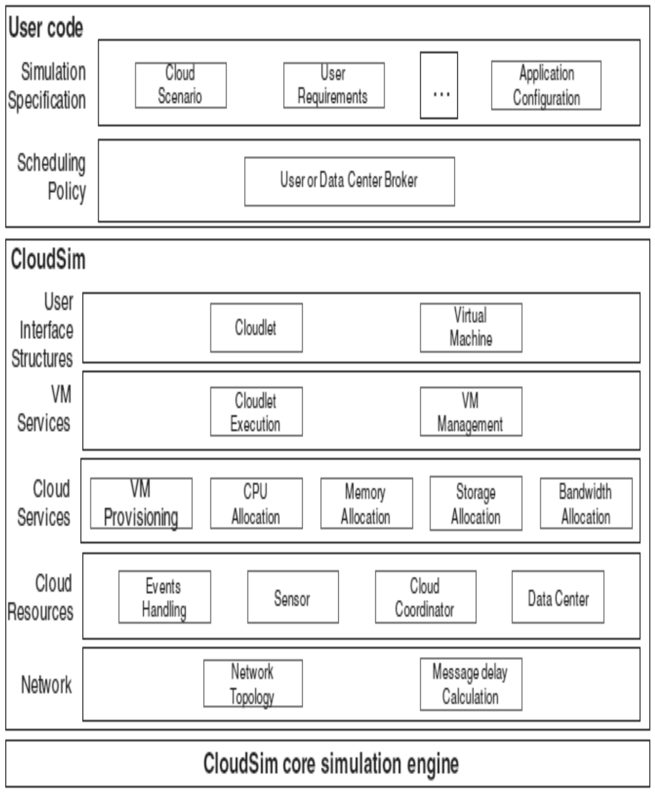


Fig.1 CloudSim Layered Architecture

2.2.1 Core Layer

This layer [12] supports several core functionalities, such as queuing and processing of events, creation of Cloud system entities (services, host, data center, broker, VMs), communication between components, and management of the simulation clock.

2.2.2 Simulation Layer

This layer is responsible for modeling and simulation of cloud-based data center. This layer includes dedicated interfaces for resource allocation such as CPU, RAM memory, storage and network bandwidth. The simulation layer manages the implementation of applications and monitor system status. It consists of five hierarchical layers:

- Network: responsible for network topology and determines the networks delays.
- Cloud resources: responsible for modeling infrastructure level services (Datacenter), monitoring the internal state of the resources in the datacenter, performing load balancing (Cloud Coordinator).
- Cloud services: providing hosts with virtual machines; allocation of resources such as CPU, memory, bandwidth. This layer enables the developers to implement their own techniques and different algorithms for resource allocation.
- VM services – includes components for managing virtual machines and cloudlets.
- User interface structures – provides an interface to configured virtual machines and tasks (cloudlets).

2.2.3 User Code Layer

This Layer allows developers to change the parameters of the main CloudSim objects: hosts (number of servers and their characteristics), virtual machines, users, resource planning policies. It Provides users with the ability to:

- Generate new configurations of applications.
- Perform tests on cloud-based environment based on custom configurations.
- Comparison and evaluation of techniques for allocating resources for clouds and federation of clouds.

As cloud computing is a rapidly evolving research area, there a severe lack of defined standards, tools and methods that can efficiently tackle the infrastructure and application level complexities. By extending the basic functionalities already exposed by CloudSim, researchers would be able to perform tests based on specific scenarios and configurations

3. DESIGN AND IMPLEMENTATION OF CLOUDSIM

3.1 CloudSim Classes

The Class design diagram for the simulator is depicted in Figure 2.

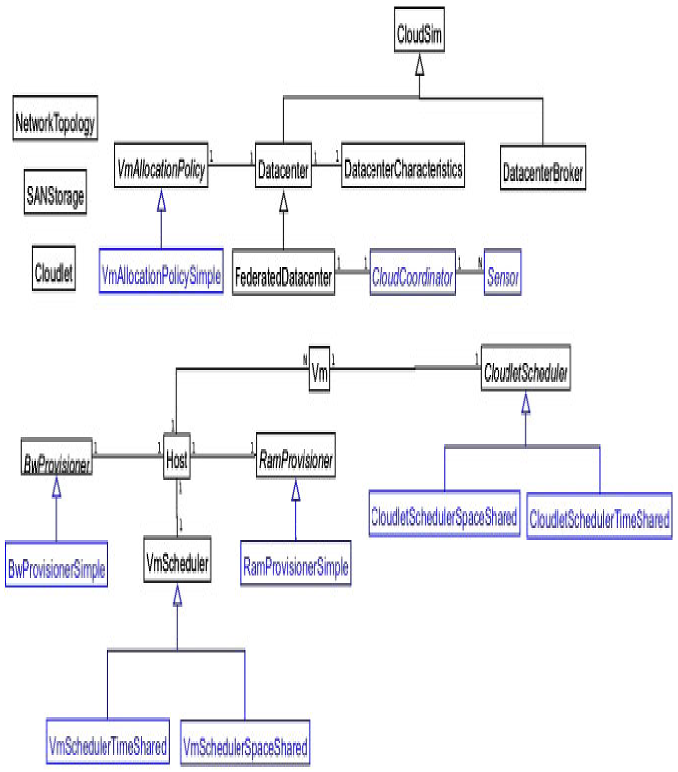


Fig. 2 CloudSim Class Diagram

Some of the classes are [13, 14]:

3.1.1 Bandwidth Provisioner

It is abstract classes that are used for provisioning of bandwidth to VM (Virtual Machine). Main task of this class is to undertake the allocation of network bandwidth to VM's set which is developed across the datacenter. If any researcher and developer want to extend this class, so they can extend this class with their own policies (priorities, QoS). BandwidthProvisioningSimple allow a VM to reserve as much as bandwidth as required.

3.1.2 Cloudlet

This class models the Cloud-based application services (content delivery, social networking, business workflow), which are commonly deployed in the data centers. CloudSim represents the complexity of an application in terms of its computational requirements. Every application component has a pre assigned instruction length and amount of data transfer that needs to be undertaken for successfully hosting the application.

3.1.3 CloudCoordinator

This abstract class provides federation capacity to a data center. This class is responsible for not only communicating with other peer CloudCoordinator services and Cloud Brokers (DataCenterBroker), but also for monitoring the internal state of a data center that plays integral role in load balancing / application scaling decision making.

3.1.4 DataCenter

This class models the core infrastructure level services (hardware, software) offered by resource providers in a Cloud computing environment. It encapsulates a set of compute hosts that can be either homogeneous or heterogeneous as regards to their resource configurations (memory, cores, capacity, and storage).Furthermore, every DataCenter component instantiates a generalized resource provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices.

3.1.5 DatacenterBroker.

This class models a broker, which is responsible for mediating between users and service providers depending on users' QoS requirements and deploys service tasks across Clouds. The broker acting on behalf of users identifies suitable Cloud service providers through the Cloud Information Service (CIS) and negotiates with them for an allocation of resources that meet QoS needs of users. The researchers and system developers must extend this class for conducting experiments with their custom developed application placement policies.

3.1.6 DatacenterCharacteristics:

This class contains configuration information of data center resources

3.1.7 Host

This class models a physical resource such as a compute or storage server. It encapsulates important information such as the amount of memory and storage, a list and type of processing cores (to represent a multi-core machine), an allocation of policy for sharing the processing power among VMs, and policies for provisioning memory and bandwidth to the VMs.

3.1.8 MemoryProvisioner

This is an abstract class that represents the provisioning policy for allocating memory to VMs. This component models policies for allocating physical memory spaces to the competing VMs. The execution and deployment of VM on a host is feasible only if the MemoryProvisioner component determines that the host has the amount of free memory, which is requested for the new VM deployment.

3.1.9 NetworkTopology

This class contains the information for inducing network behavior (latencies) in the simulation. It stores the topology information, which is generated using the BRITE topology generator.

3.1.10 SANStorage

This class models a storage area network that is commonly available to Cloud-based data centers for storing large chunks of data. SANStorage implements a simple interface that can be used to simulate storage and retrieval of any amount of data, at any time subject to the availability of network bandwidth. Accessing files in a SAN at run time incurs additional delays for task unit execution, due to time elapsed for transferring the required data files through the data center internal network.

3.1.11 Sensor

This interface must be implemented to instantiate a sensor component that can be used by a CloudCoordinator for monitoring specific performance parameters (energy-consumption, resource utilization). The methods defined by this interface are: (i) set the minimum and maximum thresholds for performance parameter and (ii) periodically update the measurement. This class can be used to model the real-world services offered by leading Cloud providers such as Amazon's CloudWatch and Microsoft Azure's Fabric Controller. One data center may instantiate one or more Sensors, each one responsible for monitoring a specific data center performance parameter.

3.1.12 VirtualMachine

This class models an instance of a VM, whose management during its life cycle is the responsibility of the Host component. A host can simultaneously instantiate multiple VMs and allocate cores based on predefined processor sharing policies (spaceshared, time-shared). Every VM component has access to a component that stores the characteristics related to a VM, such as memory, processor, storage, and the VM's internal scheduling policy, which is extended from the abstract component called VMScheduling.

3.1.13 VMProvisioner

This abstract class represents the provisioning policy that a VM Monitor utilizes for allocating VMs to Hosts. The chief functionality of the VMProvisioner is to select available host in a data center, which meets the memory, storage, and availability requirement for a VM deployment. The default SimpleVMProvisioner implementation provided with the CloudSim package allocates VMs to the first available Host that meets the aforementioned requirements. Hosts are considered for mapping in a sequential order. However, more complicated policies can be easily implemented within this component for achieving optimized allocations, for example, selection of hosts based on their ability to meet QoS requirements such as response time, budget.

3.1.14 VMMAllocationPolicy

This is an abstract class implemented by a Host component that models the policies (space-shared, time-shared) required for allocating processing power to VMs. The functionalities of this class can easily be overridden to accommodate application specific processor sharing policies.

3.2 CLOUDSIM EVENT MODEL

The CloudSim package doesn't have any dependencies to other simulation frameworks. It is therefore a self-contained simulation framework with all elements that are required. The class CloudSim is the main simulation class. The whole event simulation [15] is processed within the same thread. It is based on a clock that is not determined by the actual time of the day and simply starts with zero. The events are getting executed in a procedural way and not in a real time fashion as one might expect from a simulation framework. However, the execution of the simulation runs fast, but the log that is created contains the correct timestamps as it would have been executed in real time.

Figure 3 shows CloudSim Event Model. For simplification, this illustration depicts only some selected methods/attributes that have some relevance within the event model.

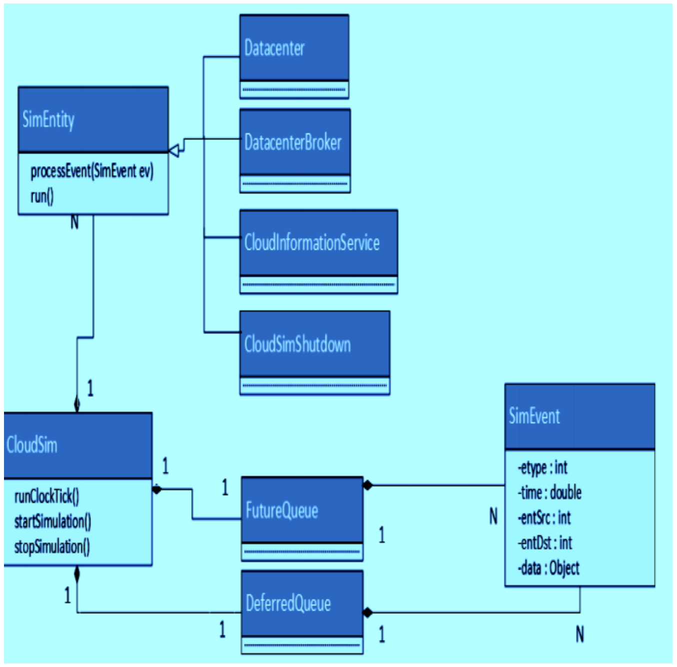


Fig. 3 CloudSim Event Model

Within the simulation there is the abstract SimEntity class. It is able to handle events and send events to other entities. Subclasses of the SimEntity are: Datacenter, DatacenterBroker, CloudInformationService and CloudSimShutdown.

The actual events are represented with the SimEvent class. An event contains the time, when it should be started, the source entity and destination entity (SimEntity class), the event type and some arbitrary data that can be transmitted with the event.

The CloudSim.runClockTick () method iterates through all entities (SimEntity classes) within the simulation and executes the run () method on these objects. They then process the events in the DeferredQueue that are sent to the entity

Whatever data centers we have, they must register with CIS First. After registration, user will send request. What so ever request is there, that is considered as Event. Whenever request comes, it is put into the queue which is known as FutureQueue. It is equivalent to process scheduler that will keep all tasks in the ready queue which is equivalent to FutureQueue.

Whenever you want to handle that event, you delete it from the FutureQueue and now you have to put that into DeferredQueue. FutureQueue contains events that are sent from one entity to another on some point in the future. As soon as the simulator's clock is at this point, the event is getting added to the DeferredQueue until they are all processed by the entity objects.

Another central class in the simulation process is the CloudInformationService. The Cloud Information Service (CIS) is an entity that provides cloud resource registration, indexing and discovery services. The Cloud has a list of hosts that tell their readiness to process Cloudlets by registering themselves with the CloudInformationService. Other entities such as the DatacenterBroker can contact this class for resource discovery service, which returns a list of registered resource IDs. In summary, it acts like a yellow page service. This class will be created by CloudSim upon initialization of the simulation [16].

3.2.1 Predicates

Predicates: Predicates are used for selecting events from the deferred queue. This is an abstract class and must be extended to create a new predicate. Some standard predicates are provided that are presented in Figure 4.

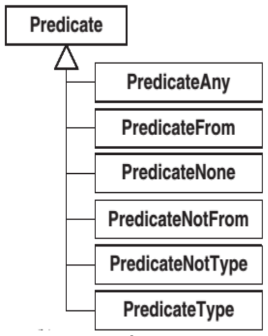


Fig. 4 CloudSim core simulation framework class diagram: predicates

- PredicateAny: This class represents a predicate that matches any event on the deferred event queue. There is a publicly accessible instance of this predicate in the CloudSim class, called CloudSim.SIMANY, and hence no new instances need to be created.
- PredicateFrom: This class represents a predicate that selects events fired by specific entities.
- PredicateNone: This represents a predicate that does not match any event on the deferred event queue. There is a publicly accessible static instance of this predicate in the CloudSim class, called CloudSim.SIMNONE; hence, the users are not needed to create any new instances of this class.
- PredicateNotFrom: This class represents a predicate that selects events that have not been sent by specific entities.
- PredicateNotType: This class represents a predicate to select events that do not match specific tags.
- PredicateType: This class represents a predicate to select events with specific tags.

3.3 Cloudlets Processing in Data Centre

Processing of task units in Data Centre is shown in figure 5 as a sequence diagram.

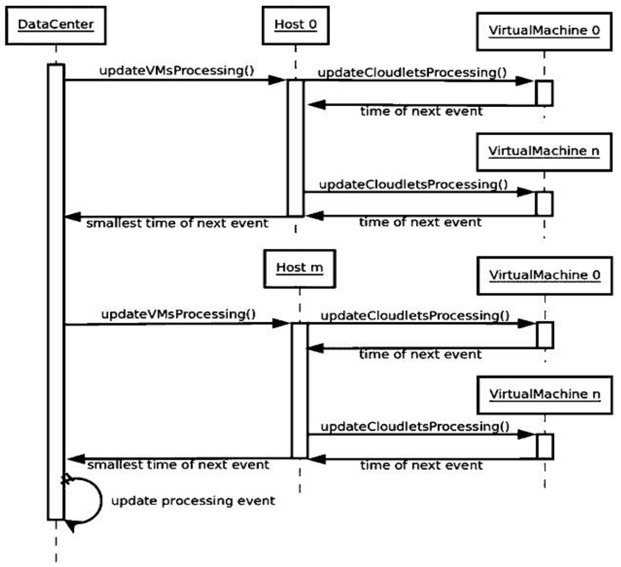


Fig. 5 Cloudlets Processing in Data Centre

Figure 5 shows what happens when an event is handled and what information is needed to pass. Here we have one data center; upon data center we have hosts and upon hosts we have virtual machines. Main purpose of virtual machine is to execute Cloudlet; whenever that event is over, we have to send information back to data center because with the help of data center, users are interacting through the data center for executing their jobs; whenever job completes, we have to send reply back to data center; and from data center, users are able to get back the reply.

Now, data center sends message updateVMProcessing () to its hosts, this event is just to know that what is going on that particular virtual machine; at which state now it is i.e. whether you have created virtual machine; whether virtual machine is executing some task or it had done it and we are able to destroy virtual machine. So, Data center will send the query updateVMProcessing () to all the hosts available. Now what host will do? It will send message to Virtual machine regarding what is happening to Cloudlet. It will send message updateCloudletProcessing () to virtual machine. Now virtual machine sends back the reply i.e. time of next event. Next event can either be execution of next instruction of Cloudlet or the messages that will show now the execution is complete, now you can either shutdown virtual machine or destroy it. All virtual machines will send back the same reply.

If host has 10 virtual machines, all machines will send time of their next event. So now host will get back 10 replies; host will just send single reply to data center; it will just choose smallest reply from these 10 replies; it will send smallest time of next event. Data center depending upon this type of event will send back reply to customer.

3.4 CIS Registry

In the beginning of the simulation, each Datacenter entity registers itself with the CIS (Cloud Information Service) Registry. CIS provides database level match-making services for mapping user requests to suitable Cloud providers. Brokers acting on behalf of users consult the CIS service about the list of Clouds who offer infrastructure services matching user's application requirements. In case the match occurs the broker deploys the application with the Cloud that was suggested by the CIS.

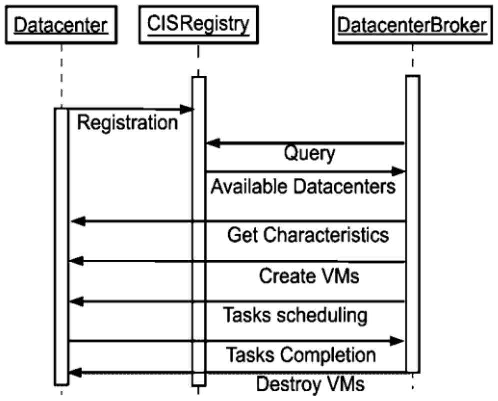


Fig. 6 CIS Registry

Whenever there are Cloud service providers, they have to register themselves with CIS. Data centers are registered with CIS Registry. If users are submitting their requests to brokers, broker will first find the list of available Datacenters. So, it will send query to CIS that what type of data centers it have. Users can also query regarding their characteristics from particular data center. Let us suppose user is from Ludhiana and we have one data center in Chandigarh, one in Delhi and one in Mumbai. Now what should be the user preference; obvious Chandigarh because it is near. So user will query regarding Chandigarh data center. Based upon these characteristics, if they are able to run user task then suppose user will say he just want 5 Virtual machines to be deployed; over that particular one he will schedule his task and so on; After task is completed, user can send message to destroy virtual machine.

3.5 CloudSim physical resources model

Physical resources are defined on different levels with different attributes. Table 2 lists some important attributes, when it comes to resource scheduling and allocation.

Resource	Attribute	Meaning
Host	peList	Number of processing cores.
	ram	The amount of memory associated with the host.
	bandwidth	The bandwidth that is reserved for the host.
	storage	The hard disk size a host has.
VM	numberOfPes	Number of processing elements (cores) required.
	ram	The amount of memory required.
	bandwidth	The bandwidth that is required.
	size	Storage size in MB.
	userID	The user identity of the owning user.
Cloudlet	cloudletLength	Number of Million Instructions (MI) that are required for the job.
	cloudletFileSize	Disk space needed, when starting the job (MB).
	cloudletOutputSize	Disk space needed, when the job is finished (MB).
	numberOfPes	Number of processing elements (cores) the job needs to execute.
	userID	The user identity of the owning user.
	vmID	The vmID where this cloudlet is supposed to run.
	utilizationModelCpu	The utilization model of the CPU.
	utilizationModelRam	The utilization model of the RAM.
	utilizationModelBw	The utilization model of the bandwidth.

Table.2 Resource Objects

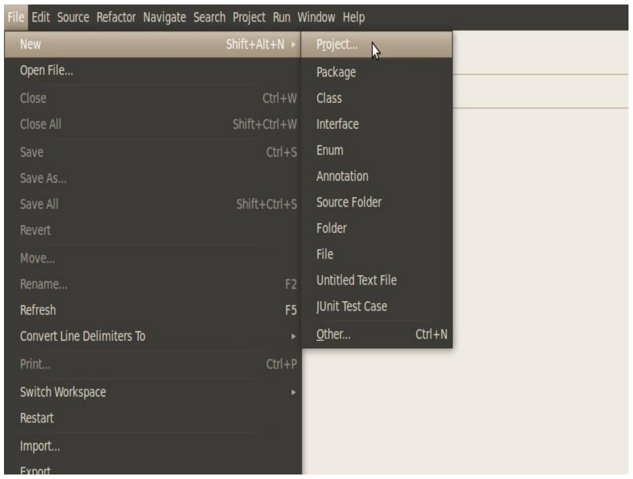
4. Working with CloudSim

4.1 Installing CloudSim

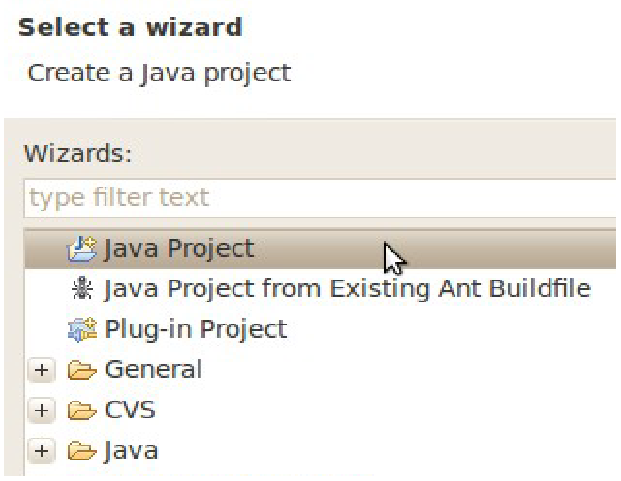
CloudSim is library (tool) for cloud computing simulation written in java language. So we should have a basic knowledge of java programming. CloudSim installation is not needed because it is a library so we just have to unpack the package and then add .jar file as a library into our project. It can work with any programming IDEs that support java like netbeans or eclipse [17].

Following are the steps for how to use CloudSim with netbeans [18]:

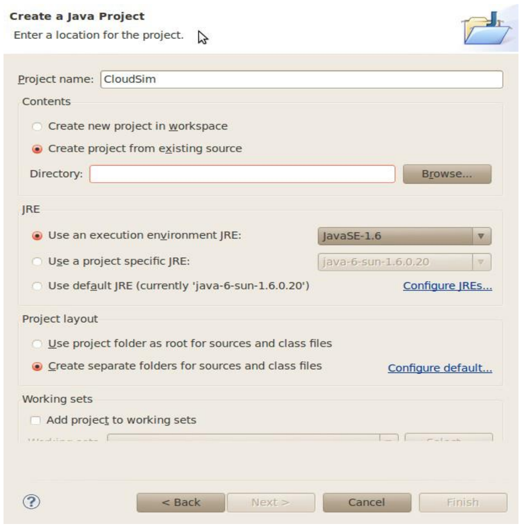
1. Download eclipse from http://www.eclipse.org/downloads/
2. Extract eclipse to particular directory. Here let's say C:\eclipse
3. Download CloudSim from http://code.google.com/p/cloudsim/downloads/list
4. Extract CloudSim to particular directory. Here let's say C:\cloudsim-3.0.2
5. Open Eclipse
6. In the Eclipse menu, select "New" → "Project..."



7. In the "Select a Wizard" window, select "Java Project" then click "Next"



8. In the "Create a Java Project" window, fill the field "Project name" with CloudSim. Then select "Create project from existing source". In the "Directory" field, select the directory extracted from the CloudSim package. If you have more than one JVM, in this window you have to select Sun Java 6. Then, select "Finish" to complete project creation.



Create a Java Project
Enter a location for the project.

Project name: CloudSim

Contents
- Create new project in workspace
- [✓ Create project from existing source]

Directory: Browse...

JRE
- [✓ Use an execution environment JRE:] JavaSE-1.6
- Use a project specific JRE: java-6-sun-1.6.0.20
- Use default JRE (currently 'java-6-sun-1.6.0.20') Configure JREs...

Project layout
- Use project folder as root for sources and class files
- [✓ Create separate folders for sources and class files] Configure default...

Working sets
- Add project to working sets

< Back Next > Cancel Finish

9. After these steps, CloudSim you can navigate through CloudSim packages, and develop your own simulations using CloudSim.

4.2 CloudSim Life Cycle

CloudSim Life Cycle is shown in figure 7.

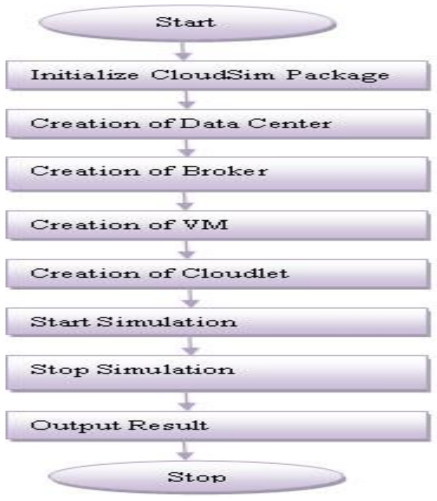


Fig. 7 CloudSim Life Cycle

Step 1: First step: Initialize the CloudSim package. It should be called before creating any entities.

Int num_user = 1; //number of cloud users
Calendar calendar = Calender.getInstance(); // Calendar whose fields have been initialized with the current date and time.
Boolean trace_flag = false; //mean trace events
CloudSim.init(num_user, calendar, trace_flag);

Step 2: Create Datacenters

Datacenter datacenter0 = createDatacenter ("Datacenter_0");

Step3: Create Broker

```
    DatacenterBroker broker = createBroker();
    int brokerId = broker.getId();
```

Step 4: Create virtual machine

```
    vmlist = new ArrayList<Vm>();
    // VM 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
    Vm vm = new Vm(vmid, brokerId, mips, pesNumber,
    ram, bw, size, vmm, new
    CloudletSchedulerTimeShared());                // create
    VM
    vmlist.add(vm);   // add the VM to the vmList
    broker.submitVmList(vmlist); // submit vm list to the
broker
```

Step 5: Create Cloudlet

```
    cloudletList = new ArrayList<Cloudlet>();
    // Cloudlet properties
    int id = 0;
    long length = 400000;
    long fileSize = 300;
    long outputSize = 300;
    UtilizationModel utilizationModel = new
UtilizationModelFull();
    Cloudlet cloudlet = new Cloudlet (id, length,
    pesNumber, fileSize, outputSize, utilizationModel,
    utilizationModel, utilizationModel);
    cloudlet.setUserId(brokerId);
    cloudlet.setVmId(vmid);
    cloudletList.add(cloudlet);             // add the cloudlet
to the list
    broker.submitCloudletList(cloudletList); // submit
cloudlet list to the broker
```

Step 6: Start the simulation

```
    CloudSim.startSimulation();
```

Step 7: Stops the simulation

```
    CloudSim.stopSimulation();
```

Step 8: Print results when simulation is over

```
    List<Cloudlet> newList =
broker.getCloudletReceivedList();
    printCloudletList(newList);
    Log.printLine("CloudSimExample1 finished!");
```

Create a Data Center

Step 1: Create a list to store hosts

```
    List<Host> hostList = new ArrayList<Host> ();
```

Step 2: Create a list to store PEs or CPUs/Cores

```
    List<Pe> peList = new ArrayList<Pe>();  // In this
example, it will have only one core.
    int mips = 1000;
```

Step 3: Create PEs and add these into a list.

```
    peList.add(new Pe(0, new
PeProvisionerSimple(mips))); // need to store Pe id and MIPS
Rating
```

Step 4: Create Host with its id and list of PEs and add them to the list of machines

```
    int hostId = 0;
    int ram = 2048; // host memory (MB)
    long storage = 1000000; // host storage
    int bw = 10000;
    hostList.add(new Host(hostId,new
    RamProvisionerSimple(ram),new
    BwProvisionerSimple(bw),storage,peList,new
    VmSchedulerTimeShared(peList))); // This is our
    machine
```

Step 5: Create a DatacenterCharacteristics object that stores the properties of a data center: architecture, OS, list of Machines, allocation policy: time- or space-shared, time zone and its price (G$/Pe time unit).

```
    String arch = "x86"; // system architecture
    String os = "Linux"; // operating system
    String vmm = "Xen";
    double time_zone = 10.0; // time zone this resource
located
    double cost = 3.0; // the cost of using processing in
this resource
    double costPerMem = 0.05; // the cost of using
memory in this resource
    double costPerStorage = 0.001; // the cost of using
storage in this resource
    double costPerBw = 0.0; // the cost of using bw in
this resource
    LinkedList<Storage> storageList = new
LinkedList<Storage>();
    DatacenterCharacteristics characteristics = new
    DatacenterCharacteristics(arch, os, vmm, hostList,
    time_zone, cost, costPerMem, costPerStorage,
    costPerBw);
```

Step 6: Final step is to create a Datacenter object

```
    Datacenter datacenter = null;
        try {
            datacenter = new Datacenter(name,
            characteristics, new
            VmAllocationPolicySimple(hostLis
            t), storageList, 0);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

5. VARIOUS POLICIES IN CLOUDSIM

5.1 VMSchedulingPolicy

CloudSim models scheduling of CPU resources at two levels: Host and VM.

At Host level, the host shares fractions of each processor element (PE) to each VM running on it. Because resources are shared among VMs, this scheduler is called VmScheduler. The scheduler a host uses is a parameter of the Host constructor. In the VM level, each virtual machine divides the resources received from the host among Cloudlets running on it. Because in this level resources are shared among Cloudlets, this scheduler is called CloudletScheduler. The scheduler a VM uses is a parameter of its constructor.

In both levels, there are two default policies available:

- SpaceShared: In this policy required PEs by Cloudlets/VMs are exclusively allocated. It means that if there are more running elements (VMs or Cloudlets) than available PEs, the last elements to arrive wait on a queue until enough resources are free.
- TimeShared: In this policy fraction of available PEs are shared among running elements, and all the elements run simultaneously.

Policies for VM Scheduling and Cloudlet Scheduling can be used in any combination. For example, researchers can use VmSchedulerTimeShared and CloudletSchedulerSpaceShared, or researchers can use VmSchedulerTimeShared and CloudletSchedulerTimeShared. It is possible even having a host running VMs with different Cloudlet scheduling policies, or a data center with hosts with different VM Scheduling policies. To define your own policy, you have to extend either VmScheduler or CloudletScheduler, create the methods for deciding sharing of PEs and pass the new class during construction of the relevant object. For example, extend VmScheduler and pass the object to the host; or extend CloudletScheduler and pass the object to the VM.

5.2 VMProvisioningPolicy

The provisioning problem consists of defining, among the available hosts in the data center, which one should receive a new machine requested by a user. Provisioning of hosts to VMs in data centers follows a simple strategy where the host with less running VMs receives the next VM. This behavior is defined in the VmAllocationPolicySimple class. To change this behavior, extend VmAllocationPolicy to define the new provisioning behavior, and pass this object in the initialization of Datacenter.

- VmAllocationPolicy is an abstract class that represents the provisioning policy of hosts to virtual machines in a datacenter.
- PowerVmAllocationPolicyAbstract is an abstract class which defined a power-aware VM allocation policy.
- PowerVmAllocationPolicyMigrationAbstract is an abstract class which defined a power-aware VM allocation policy that dynamically optimizes the VM allocation using migration.

The other classes, which defined some different policy of power-aware VM allocation policy, are all extended from PowerVmAllocationPolicyMigrationAbstract or PowerVmAllocationPolicyAbstract. Researchers can also implement dynamic VM reallocation algorithms by implementing the optimizeAllocation method of the PowerVmAllocationPolicyAbstract class, which is called at every time frame and passed with the full set of current VMs in the data center.

5.3 VMSelectionPolicy

The VM selection problem consists of defining, among the VMs in a certain host, which one should migrate to a new machine requested by a user because of the power consideration. PowerVmSelectionPolicy is an abstract class that represents the VM selection policy. CloudSim defines four VM selection policies for the host to choose which VM should be migrated:

- The Minimum Migration Time (MMT) policy select a VM that requires the minimum time to complete a migration relatively to the other VMs allocated to the host.
- The Random Choice (RC) policy selects a VM to be migrated according to a uniformly distributed discrete random variable.
- The Maximum Correlation (MC) policy selects those VMs to be migrated that have the highest correlation of the CPU utilization with other VMs.
- The Minimum Utilization (MU) policy selects a VM to be migrated that requires the minimum CPU utilization relatively to the other VMs in the host.

6. CONCLUSION

Cloud computing has been one of the fastest growing parts in IT industry. Simulation based approaches become popular in industry and academia to evaluate cloud computing systems, application behaviors and their security. Several simulators have been specifically developed for performance analysis of cloud computing environments including CloudSim, GreenCloud, NetworkCloudSim, CloudAnalyst, EMUSIM and MDCSim. The CloudSim simulator is probably the most sophisticated among the simulators overviewed. Researchers from different Universities use and develop CloudSim by creating their own mechanisms for resource allocation and delivery of services. They use it for evaluation of algorithms for resource allocation; analysis of energy efficiency of data centers; optimization of cloud environments. This paper introduced the CloudSim simulator including its architecture, and how to use it to model the cloud environment. CloudSim supports flexible, scalable, efficient and repeatable evaluation of provisioning policies for different applications. This paper also compares different simulator on the basis of some parameter like platform, programming language, physical support and graphical support. In the future, we will use CloudSim platform to evaluate algorithm which aims to improve the average resource utilization of the cloud datacenter and achieve energy efficiency.

REFERENCES

[1]. M. Armbrust, M. Armbrust, A. Fox, A. Fox, R. Griffith, R. Griffith, A. Joseph, A. Joseph, RH, and RH, “Above the clouds: A Berkeley view of cloud computing,” *Univ.California, Berkeley, Tech. Rep. UCB*, pp. 07–013, 2009.

[2]. Xiaoying Bai, Muyang Li, Bin Chen, W.T. Tsai & J. Gao (2011), “Cloud Testing Tools”, *Proceedings of the 6th IEEE International Symposium on Service Oriented System Engineering*, Pp. 1–12.

[3]. B. Wickremasinghe (2009), “CloudAnalyst: A CloudSim based Tool for Modeling and Analysis of Large Scale Cloud Computing Environments”, *MEDC Project Report*.

[4]. Dzmitry Kliazovich, Pascal Bouvry & Samee Ullah Khan (2010), “GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers”, *Springer Science+Business Media*, LLC, Luxembourg.

[5]. Dr. Pawan Kumar & Gaganjot Kaur , “Study of Comparison of Various Cloud Computing Simulators”, IITT College of Engineering & Technology, *2nd National Conference in Intelligent Computing & Communication*, GCET Greater Noida, India

[6]. Dr. Rahul Malhotra & Prince Jain (2013), “An EMUSIM Techniques and its Components in a Cloud Computing Environment”, International Journal of Computer Trends and Technology (IJCTT), Vol. 4, No. 8, Pp. 2435–2440.

[7]. Tarun Goyal,Ajit Singh, Akansha Agrawal, CloudSim: Simulator for cloud computing infrastructure and modeling‖, International Conference on modeling , optimization and computing Published by Elsevier Ltd (ICMOC 2012).

[8]. Kalpana Ettikyala, Y Rama Devi, A Study on Cloud Simulation Tools‖, International Journal of Computer Applications , Volume 115 – No. 14, April 2015, ISSN: 0975 – 8887.

[9]. 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” in Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, pp.1-9,2009.

[10]. CloudSim Features , http://www.cloudbus.org/cloudsim/ Accessed 17 July, 2016

[11]. Calheiros, R., R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software: Practice and Experience (SPE), 41, January 2011, Number 1, 23-50, ISSN: 0038-0644, New York, USA, Wiley Press.

[12]. Howell F, Mcnab R. SimJava: A discrete event simulation library for java. Proceedings of the First International Conference on Web-based Modeling and Simulation, San Diego, U.S.A., 1998

[13]. Rushikesh Shingade , Amit Patil , Shivam Suryawanshi , M. Venkatesan , ―Efficient Resource Management in Cloud Computing‖, e-ISSN : 0975-4024 Rushikesh Shingade et al. / International Journal of Engineering and Technology (IJET), Vol 7 No 6 Dec 2015-Jan 2016 p-ISSN 0975-4024.

[14]. Rodrigo N. Calheiros, Rajiv Ranjan, Cesar A. F. De Rose, and Rajkumar Buyya, CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services‖ SOFTWARE – PRACTICE AND EXPERIENCE, Published online 24 August 2010 in Wiley Online Library (wileyonlinelibrary.com).

[15]. Patrick A. Taddei, Design and Development of a CloudSim Module to Model and Evaluate Multi-resource Dependencies, Bachelor Thesis, Communication Systems Group (CSG), Department of Informatics (IFI), University of Zurich.

[16]. Cloudsim 3.0 api. http://www.cloudbus.org/cloudsim/doc/api/ Accessed 22 July, 2016.

[17]. Harsha Amipara, A Survey on CloudSim Toolkit for Implementing Cloud Infrastructure, International Journal of Science Technology & Engineering , Volume 1, Issue 12, June 2015

[18]. Install CloudSim with eclipse, http://www.acadox.com/action_handler/download/resource/3525/6005.pdf Accessed 29 July,2016.