CHAPTER 3

RESEARCH METHODOLOGY

3.1 SIMULATION TOOL USED: CLOUDSIM

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. Based on Rodrigo et al (2009) evaluation results reported by CloudSim, further fine tune the service performance. The main advantage of using CloudSim for initial performance testing include:

- **Time effectiveness**: it requires very less effort and time to implement Cloud-based application provisioning test environment and

- **Flexibility and applicability**: developers can model and test the performance of their application services in heterogeneous Cloud environments (Amazon EC2, Microsoft Azure) with little programming and deployment effort.

It offers the following novel features: (i) support for modeling and simulation of large scale Cloud computing environments, including data centers, on a single physical computing node; (ii) a self-contained platform for modeling Clouds, service brokers, provisioning, and allocations policies; (iii) support for simulation of network connections among the simulated system elements; and (iv) facility for simulation of federated Cloud environment that inter-networks resources from both private and public domains, a feature critical for research studies related to Cloud-Bursts and automatic application scaling.
Some of the unique features of CloudSim are:

- availability of a virtualization engine that aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node

- Flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services. These compelling features of CloudSim would speed up the development of new application provisioning algorithms for Cloud computing.

### 3.1.1 CloudSim Architecture

![Layered CloudSim Architecture](image)

*Figure 3.1 Layered CloudSim Architecture*
Figure 3.1 shows the multi-layered design of the CloudSim software framework and its architectural components. Initial releases of CloudSim used SimJava as discrete event simulation engine that supports several core functionalities, such as queuing and processing of events, creation of Cloud system entities (services, host, data center, broker, and virtual machines), communication between components, and management of the simulation clock. However in the current release, SimJava layer has been removed in order to allow some advanced operations that are not supported by it. The CloudSim simulation layer provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for virtual machines (VMs), memory, storage, and bandwidth.

The fundamental issues such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state are handled by this layer. A Cloud provider, who wants to study the efficiency of different policies in allocating its hosts to VMs (VM provisioning), would need to implement their strategies at this layer. Such implementation can be done by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer related to provisioning of hosts to VMs.

A Cloud host can be concurrently allocated to a set of VMs that execute applications based on SaaS provider’s defined QoS levels. This layer also exposes functionalities that a Cloud application developer can extend to perform complex workload profiling and application performance study. The top-most layer in the CloudSim stack is the User Code that exposes basic entities for hosts (number of machines, their specification and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. By extending the basic entities given at this layer, a Cloud application developer can perform following activities: (i) generate a mix of workload request distributions,
application configurations; (ii) model Cloud availability scenarios and perform robust tests based on the custom configurations; and (iii) implement custom application provisioning techniques for clouds and their federation.

### 3.1.1.1 Modeling the Cloud

Infrastructure-level services (IaaS) related to the clouds can be simulated by extending the Datacenter entity of CloudSim represented in Figure 3.2. 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. A Datacenter can manage several hosts that in turn manage VMs during their life cycles.

![Architecture of CloudSim Data Center](image)

**Figure 3.2 Architecture of CloudSim Data Center**
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 virtual machines. 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. By default, VmAllocationPolicy implements a straightforward policy that allocates VMs to the Host in 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.

3.1.1.2 Modeling the VM Allocation

Cloud host that has a single processing core, there is a requirement of concurrently instantiating two VMs on that host. Even though in practice VMs are contextually (physical and secondary memory space) isolated, still they need to share the processing cores and system bus. Hence, the amount of hardware resources available to each VM is constrained by the total processing power and system bandwidth available within the host. This critical factor must be considered during the VM provisioning process, to avoid creation of a VM that demands more processing power than is available within the host.
In order to allow simulation of different provisioning policies under varying levels of performance isolation, CloudSim supports VM provisioning at two levels: first, at the host level and second, at the VM level. At the host level, it is possible to specify how much of the overall processing power of each core will be assigned to each VM. At the VM level, the VM assigns a fixed amount of the available processing power to the individual application services (task units) that are hosted within its execution engine. At each level, CloudSim implements the time-shared and space-shared provisioning policies. To clearly illustrate the difference between these policies and their effect on the application service performance.

3.1.1.3 Modeling the Cloud Marketplace

Market is a crucial component of the Cloud computing ecosystem; it is necessary for regulating Cloud resource trading and on-line negotiations in public Cloud computing model, where services are offered in a pay-as-you-go model. Hence, research studies that can accurately evaluate the cost-to-benefit ratio of emerging Cloud computing platforms are required. Furthermore, SaaS providers need transparent mechanisms to discover various Cloud providers’ offerings (IaaS, PaaS, SaaS, and their associated costs). Thus, modeling of costs and economic policies are important aspects to be considered when designing a Cloud simulator.

Cloud marketplace is modeled based on a multi-layered (two layers) design. First layer contains economic of features related to IaaS model such as cost per unit of memory, cost per unit of storage, and cost per unit of used bandwidth. Cloud customers (SaaS providers) have to pay for the costs of memory and storage when they create and instantiate VMs whereas the costs for network usage are only incurred in event of data transfer. Second layer models the cost metrics related to SaaS model. Costs at this layer are directly applicable to the task units (application service requests) that are served by the
application services. This behavior may be changed or extended by CloudSim users.

3.1.1.4 Modeling the Network Behavior

Modeling comprehensive network topologies to connect simulated Cloud computing entities (hosts, storage, end-users) is an important consideration because latency messages directly affects the overall service satisfaction experience. An end user or a SaaS provider consumer who is not satisfied with the delivered QoS is likely to switch their Cloud provider hence it is very important requirement that Cloud system simulation frameworks provide facilities for modeling realistic networking topologies and models. Inter-networking of Cloud entities (data centers, hosts, SaaS providers, and end-users) in CloudSim is based on a conceptual networking abstraction. In this model, there are no actual entities available for simulating network entities, such as routers or switches.

Figure 3.3 Network Communication Flow

Instead, network latency that a message can experience on its path from one CloudSim entity (host) to another (Cloud Broker) is simulated based on the information stored in the latency matrix. The transition diagram
representing such an interaction is depicted in Figure 3.3. This method of simulating network latencies gives us a realistic yet simple way of modeling practical networking architecture for simulation environment. Further, this approach is much easier and cleaner to implement, manage, and simulate than modeling complex networking components such as routers, switches etc.

3.1.1.5 Modeling a Federation of Clouds

Fundamental aspects that must be handled when simulating a federation of clouds include: communication and monitoring. The first aspect (communication) is handled by the data center through the standard event-based messaging process. The second aspect (data center monitoring) is carried out by the CloudCoordinator. Every data center in CloudSim needs to instantiate this entry in order to make itself a part of Cloud federation. The CloudCoordinator triggers the inter-cloud load adjustment process based on the state of the data center. The specific set of events that affect the adjustment are implemented via a specific sensor entity. Each sensor entity implements a particular parameter (such as under provisioning, over provisioning, and SLA violation) related to the data center. For enabling on-line monitoring of a data center host, a sensor that keeps track of the host status (utilization, heating) is attached with the CloudCoordinator.

3.1.2 CloudSim Core Simulation Framework

Fundamental classes of CloudSim, which are also the building blocks of the simulator. The overall Class design diagram for CloudSim is shown in Figure 3.4.

- CloudSim: This is the main class, which is responsible for managing event queues and controlling step by step (sequential) execution of simulation events. Every event that is generated by the CloudSim entity at run-time is
stored in the queue called future events. These events are sorted by their time parameter and inserted into the queue. Next, the events that are scheduled on each step of the simulation are removed from the future events queue and transferred to the deferred event queue. Following this, an event processing method is invoked for each entity, which chooses events from the deferred event queue and performs appropriate actions.

- **DeferredQueue**: This class implements the deferred event queue used by CloudSim.

- **FutureQueue**: This class implements the future event queue accessed by CloudSim.

![CloudSim Class Diagram](image-url)

**Figure 3.4: CloudSim Class Diagram**
- **CloudInformationService**: A Cloud Information Service (CIS) is an entity that provides resource registration, indexing, and discovering capabilities. CIS supports two basic primitives: (i) publish(), which allows entities to register themselves with CIS and (ii) search(), which allows entities such as CloudCoordinator and Brokers in discovering status and endpoint contact address of other entities. This entity also notifies the (other?) entities about the end of simulation.

- **SimEntity**: This is an abstract class, which represents a simulation entity that is able to send messages to other entities and process received messages as well as fire and handle events. All entities must extend this class and override its three core methods: startEntity(), processEvent() and shutdownEntity(), which define actions for entity initialization, processing of events and entity destruction respectively. SimEntity class provides the ability to schedule new events and send messages to other entities, where network delay is calculated according to the BRITE model. Once created, entities automatically register with CIS.

- **CloudSimTags**: This class contains various static event/command tags that indicate the type of action that needs to be undertaken by CloudSim entities when they receive or send events.

- **SimEvent**: This entity represents a simulation event that is passed between two or more entities. SimEvent stores following information about an event: type, init time, time at which the event should occur, finish time, time at which the event should be delivered to its destination entity, IDs of the source and destination entities, tag of the event and data that has to be passed to the destination entity.

- **CloudSimShutdown**: This is an entity that waits for the termination of all end-user and broker entities, and then signals the end of simulation to CIS.
• **Predicate**: 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 3.5 (b).

![CloudSim core simulation framework class diagram](image)

**Figure 3.5** CloudSim core simulation framework class diagram:
(a) main classes (b) 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.SIM_ANY, 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.SIM_NONE, 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 don't match specific tags.

- **PredicateType**: This class represents a predicate to select events with specific tags.

3.2 SIMULATION SETUP

3.2.1 HARDWARE SPECIFICATION

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel core i3</td>
</tr>
<tr>
<td>Processor Speed</td>
<td>3.20GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>2 GB</td>
</tr>
<tr>
<td>Hard Disk Capacity</td>
<td>320GB</td>
</tr>
<tr>
<td>Key Board</td>
<td>Standard Keyboard</td>
</tr>
</tbody>
</table>

3.2.2 SOFTWARE SPECIFICATION

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Windows 7 Ultimate</td>
</tr>
<tr>
<td>Language</td>
<td>JDK 1.8</td>
</tr>
<tr>
<td>IDE</td>
<td>Eclipse</td>
</tr>
<tr>
<td>Cloud Environment</td>
<td>CloudSim ToolKit</td>
</tr>
</tbody>
</table>

3.3 SUMMARY

This research work is carried over with CloudSim for virtual machines for simulation with distinct features. Each cloudlet has defined with representation of processing unit and resources in cloud environment. This work was aimed to improve the performance of cloud computing with reservation clustering technique. It arranges virtual machines in cluster form before allocating them to the resources. This arrangement provides efficient CPU utilization and load sharing among the resources, so performance can be enhanced in some aspects. The simulated results are analyzed and compared with existing methods. It is obtained that proposed algorithms with reservation cluster is having better performance results than existing algorithm.