Chapter 7

Experimentation Environment

7.1 Simulation Platform (CloudSim)

The CloudSim toolkit is a generalized and extensible simulation framework that facilitates modeling, simulation, and experimentation of emerging technologies of Cloud computing infrastructures and application services. It also covers numerous methods that enable the researcher/programmers to focus on specific system design issues that they want to implement or investigate without getting concerned about the low level specification details of infrastructures and services of Cloud. 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 consisting of both single and inter-networked clouds (federation of clouds) (Calheiros, Ranjan, Beloglazov, De Rose & Buyya (2011)). Moreover, it exposes custom interfaces for implementing policies and provisioning techniques for allocation of VMs under inter-networked Cloud computing scenarios. Several researchers from organizations are using CloudSim in their investigation on Cloud resource provisioning and energy-efficient management of data center resources. Main Features The main Features of Cloudsim are (Calheiros, Ranjan, Beloglazov, De Rose & Buyya (2011)):

- Support for modeling and simulation of large scale Cloud computing datacenters
- 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 energy-aware computational resources
- Flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services
- Support for modeling and simulation of data center network topologies and message-passing applications
- Support for modeling and simulation of federated cloud.
- 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

7.1.1 Modeling the Cloud

Architecture: The Architecture of CloudSim is as shown in Figure 7.1

![CloudSim layered architecture](image)

Figure 7.1: CloudSim layered architecture

The core hardware Cloud infrastructure services are modeled in the simulator by different components. Basic core components classes of CloudSim are: Datacenter, DatacenterBroker, SANStorage, VirtualMachine, BWProvisioner,
Cloudlet, BWProvisioner, MemoryProvisioner, VMProvisioner, VMMAllocationPolicy. Description of this core classes are as follows:

- **Datacenter**: This class offers infrastructure level services in Cloud. It prototypes core (hardware, software) offered by service providers in a Cloud computing environment. It controls a set of physical hosts that can be categorized either as 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. Each of these components are working as follows:

- **DatacenterBroker**: This class models a broker, which is responsible for mediating between users and service providers depending on users’ QoS requirements and arranges 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 meets QoS needs of users. The researchers and system developers must extend this class for conducting experiments with their custom developed application placement policies.

- **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.

- **VirtualMachine**: This class models an instance of a VM, whose management during its life cycle is the responsibility of the Host component. As discussed earlier, a host can simultaneously instantiate multiple VMs and allocate cores based on predefined processor sharing policies (space-shared, 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.

- 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 (inherited from GridSim’s Gridlet component) and amount of data transfer (both pre and post fetches) that needs to be undertaken for successfully hosting the application.

- BWProvisioner: This is an abstract class that models the provisioning policy of bandwidth to VMs that are deployed on a Host component. The function of this component is to undertake the allocation of network bandwidths to set of competing VMs deployed across the data center. Cloud system developers and researchers can extend this class with their own policies (priority, QoS) to reflect the needs of their applications.

- 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. 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 datacenter, 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,
– 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 functionality of this class is shown in Figure 7.1. By VM processing, we mean set of operations related to VM life cycle that contains provisioning of a host to a VM, VM creation, VM destruction, and VM migration.

A Datacenter is composed by a set of hosts, which is responsible for managing VMs during their life cycles. Host is a component that represents a physical computing node in a Cloud. It is assigned a pre-configured processing (expressed in million of instructions per second in MIPS, per CPU core), memory, storage, and a scheduling 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. Allocation of application-specific VMs to Hosts in a Cloud-based data center is the responsibility of the Virtual Machine Provisioner component. This component exposes a number of custom methods for researchers, which aids in implementation of new VM provisioning policies based on optimization goals (user centric, system centric). The default policy implemented by the VM Provisioner is a straightforward policy that allocates a VM to the Host in First-Come-First-Serve (FCFS) basis. The system parameters such as the required number of processing cores, memory and storage as requested by the Cloud user form the basis for such mappings. Other complicated policies can be written by the researchers based on the infrastructure and application demands. For each Host component, the allocation of processing cores to VMs is done based on a host allocation. The policy takes into account how many processing cores will be delegated to each VM, and how much of the processing core’s capacity will effectively be attributed for a given VM. So, it is possible to assign specific CPU cores to specific VMs (a space-shared policy) or to dynamically distribute the capacity of a core among VMs (a time-shared policy), and to assign cores to VMs on demand, or to specify other policies. Each Host component instantiates a VM scheduler component that implements the space-shared or time-shared policies for allocating cores to VMs. Cloud system developers and researchers can extend the VM scheduler component for experimenting with more custom allocation policies. Next, the finer level details related to the time-shared and space-shared policies are described.
Modeling the VM allocation

One of the significant facets that make a Cloud computing different from a grid computing is the use of virtualization technologies and tools. Hence, compared to Grids, in Clouds the virtualization provides hosting environment for different services of Cloud-based applications. Virtualization creates VM over physical machine which are isolated from each other from core. This important factor considered during allocation is to avoid the creation of VM demands more resources in terms of processing power than the capacity of host. To allow simulation of different policies under different levels of abstraction, CloudSim provides VM scheduling at two levels: First, at the host level and second, at the VM level. At the first level, it is possible to specify how much of the overall processing power of each core in host will be assigned to each VM. At the next level, the VMs assign specific amount of the available processing power to the individual task units that are hosted within its execution engine. At each level, CloudSim implements the time-shared and space-shared resource allocation policies. Space-shared scheduling policy schedule one virtual machine at a given instance of a time on host and next VM will be scheduled after its completion. Same way task allocated on VM using this policy. Whereas, in Time-Shared scheduling policy it schedule all VM on host at the same time. It shared the time among all VM and schedule simultaneously on the host. This policy is also used to schedule the task on the VM.

7.1.2 Dataset(PlanetLab)

A typical workload in a cloud infrastructure will consist of a dynamic mix of heterogeneous and homogeneous applications like long computationally intensive jobs, heavy and response-time sensitive requests, and data and IO-intensive analytical tasks. An additional simulation entity within CloudSim is available called as the Utilization Model that exposes methods and variables for defining the resource and VM-level requirements of a SaaS application at the instance of deployment. In the CloudSim framework, Utilization Model is an abstract class that must be extended for implementing a workload pattern required to model the application’s resource demand. CloudSim users are required to override the method, getUtilization(), whose input type is discrete time parameter and return type is percentage of computational resource required by the Cloudlet. PlanetLab is not supporting only short-term experiments, but it also support long-running services that support a client base.
That is, rather than view PlanetLab strictly as a testbed, it can be considered as research testbed and a deployment platform. As a testbed, researchers can access to

(a) A large set of geographically distributed machines.
(b) A realistic network substrate that experiences congestion, failures, and diverse link behaviors.
(c) The potential for a realistic client workload.

Its value as a deployment platform is to provide researchers with a direct technology transfer path for popular new services, and users with access to those new services. We believe that supporting both roles is critical to the success of the system.