Study of Resource Allocation Policies and Security Threats for Data Centers

Jerry Latham, Alexia Williams, Xiang Zhao (xiang.zhao@aamu.edu)

Alabama A&M University, College of Engineering, Technology and Physical Sciences, Department of Electrical Engineering and Computer Science

What is Cloud Computing?

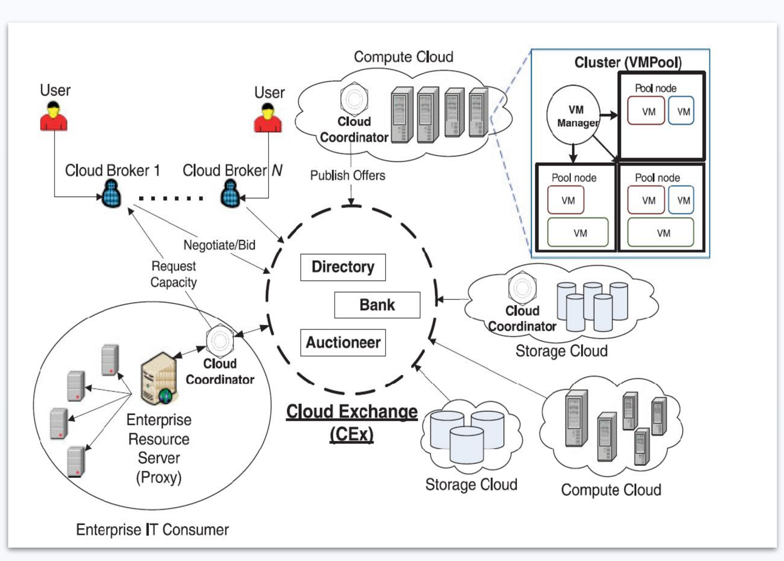
Cloud Computing is the delivery of computing services such as servers, storage, databases, networking, and software to individual users or enterprises. Cloud computing aims to cut costs and help users focus on their core objectives instead of overcoming IT obstacles due to hardware, software or time constraints.

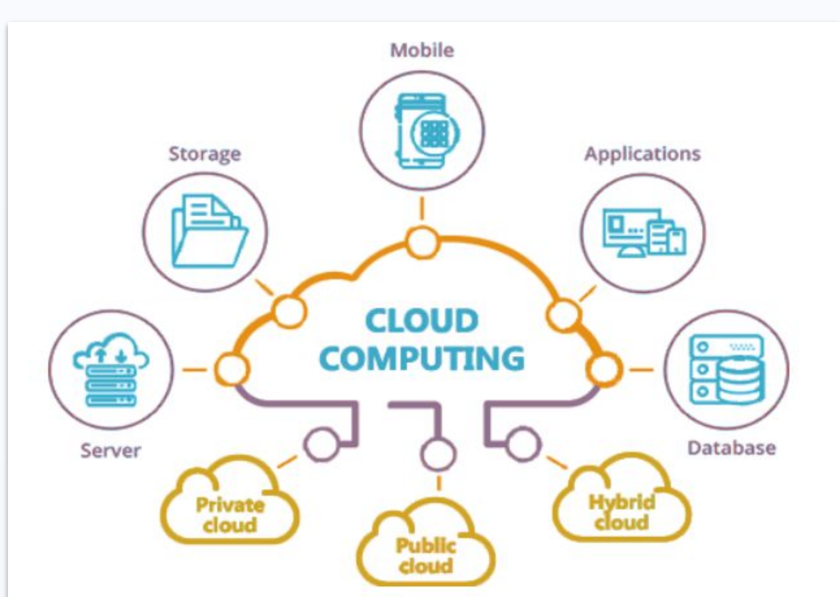
The main enabling technology for cloud computing is virtualization. Virtualization software separates a physical computing device into more separate computing devices virtually accessible and operational to meet users' requirements. The host computer is the physical hardware and provides the virtual machines (VMs). The VM, or guest, is capable of emulating different operating systems and hardware platforms via software tools. VM resource allocation is crucial to the quality of service (QoS) of cloud providers and also imposes numerous challenges, especially when the cyberattacks frequently occur nowadays.

In this project, various task scheduling and VM resource allocation policies for data centers in cloud settings are studied using modeling and simulation tools.

Cloud Computing Components

The below diagrams gives a depiction of the cloud computing process and some cloud computing utilizations. Each construct of the cloud computing process plays an important role in delivering QoS to cloud users. Having control over some of the constructs or being aware of particular processings is beneficial in optimizing the QoS and properly allocating resources. This process and many others can be practiced in CloudSim.





CloudSim

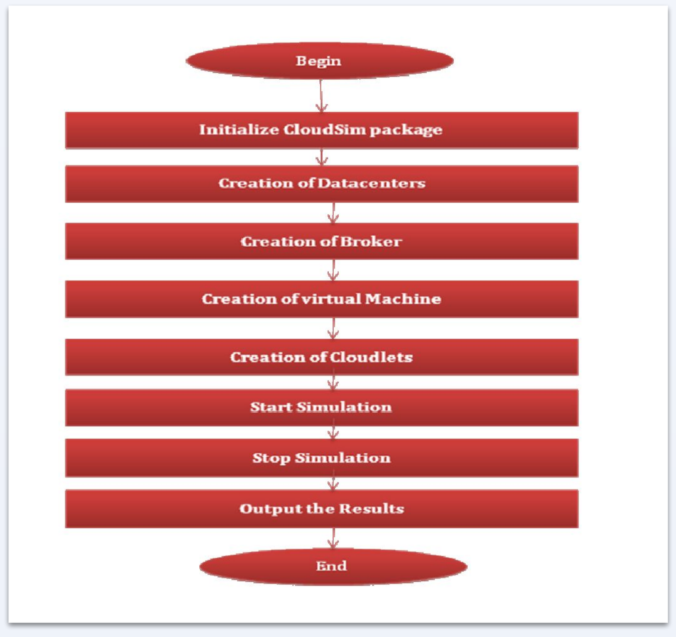
CloudSim is a simulation framework that allows modeling, simulation and experimentation of Cloud computing infrastructures. Utilizing this application enables experts and researchers to study and review simulated data using cloud computing objects as methods to control output of real functioning cloud computing technology.

Resource Allocation techniques to deliver QoS for cloud users can vary depending on which part of the cloud computing structure is being analyzed. In this experiment, we are analyzing the time it takes for the establishment of CloudLets, VMs, and DataCenter(s) in a given instance of request, whose time results are given in a rate of seconds. As shown in the flowchart figure, there are multiple steps of coding to simulate the creation of the necessary cloud computing technologies. The IDE being used to run CloudSim is Eclipse.

- Having the necessary imports are part of the CloudSim package initialization.
  - Example of some imports.

```
import org.cloudbus.cloudsim.CloudletSchedulerSpaceShared;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;


import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerSpaceShared;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
```




```
private static Datacenter createDatacenter(String name){

		// Here are the steps needed to create a PowerDatacenter:
		// 1. We need to create a list to store one or more
		//    Machines


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

		// 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should
		//    create a list to store these PEs before creating
		//    a Machine.

		//Host1(cloud host)
		List<Pe> peList1 = new ArrayList<Pe>();

		int mips = 1000; //Processing Speed

		// 3. Create PEs and add these into the list.
		//for a quad-core machine, a list of 4 PEs is required:
		peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe
id and MIPS Rating
		peList1.add(new Pe(1, new PeProvisionerSimple(mips)));
		peList1.add(new Pe(2, new PeProvisionerSimple(mips)));
		peList1.add(new Pe(3, new PeProvisionerSimple(mips)));

		//Host2(cloud host)
		//Another list, for a quad-core machine
		List<Pe> peList2 = new ArrayList<Pe>();
		//Host/CPU/HostList:
		//Simulating a quad-core machine(a list of four Processing Elements(PEs) are
required.
		peList2.add(new Pe(0, new PeProvisionerSimple(mips)));
		peList2.add(new Pe(1, new PeProvisionerSimple(mips)));
		peList2.add(new Pe(2, new PeProvisionerSimple(mips)));
		peList2.add(new Pe(3, new PeProvisionerSimple(mips)));


		//4. Create Hosts 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,
				peList1,
				new VmSchedulerSpaceShared(peList1)
			)
		); // This is our first machine
```

The createDatacenter method simulates allocating bandwidth, memory, and storage devices to hosts and VMs.

The processing elements lists are represented as one whole Host/CPU machine.

```
/** The VM list. */
	private static List<Vm> vmlist;

	private static List<Vm> createVM(int userId, int vms) {

		//Creates a container to store VMs. This list is passed to
the broker later
		LinkedList<Vm> list = new LinkedList<Vm>();

		//***VM PARAMETERS***
		long size = 875; //image size (MB)
		int ram = 1000; //VM memory (MB)
		int mips = 1000;
		long bw = 5000;
		int pesNumber = 2; //number of CPUs
		String vmm = "Xen"; //VMM name

		//create VMs
		Vm[] vm = new Vm[vms];
```

The creation of VMs and CloudLets

```
/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;
private static List<Cloudlet> createCloudlet(int userId, int cloudlets){
		// Creates a container to store Cloudlets
		LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

		//***CLOUDLET PARAMETERS***
		long length = 780;
		long fileSize = 200;
		long outputSize = 200;
		int pesNumber = 1;
		UtilizationModel utilizationModel = new
UtilizationModelFull();

		Cloudlet[] cloudlet = new Cloudlet[cloudlets];

		for(int i=0;i<cloudlets;i++){
			cloudlet[i] = new Cloudlet(i, length, pesNumber,
fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
			// setting the owner of these CloudLets
			cloudlet[i].setUserId(userId);
			list.add(cloudlet[i]);
		}

		return list;
	}
```

The output data can vary depending on the utilization of *Time Shared* or *Space Shared* allocation.
The main functionality of the *VmmAllocationPolicy* is to select the available host in a data center that meets the memory, storage, and availability requirement for a VM deployment.
The *VMScheduler* is an abstract class implemented by a Host component that models the space-shared or time-shared policies required for allocating processor cores to VMs.

```
vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerSpaceShared());


hostList.add(
			new Host(
				hostId,
				new RamProvisionerSimple(ram),
				new BwProvisionerSimple(bw),
				storage,
				peList1,
				new VmSchedulerSpaceShared(peList1)
			)
		); // This is our first machine
```

The example code gives an output of a start time(waiting time), finish time, and the turnaround time. The time results displayed in the table below represent the time required to establish each CloudLet to its DataCenter.

example output:

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
0	SUCCESS	2	0	0.78	0.1	0.88
4	SUCCESS	2	0	0.78	0.1	0.88
2	SUCCESS	2	2	0.78	0.1	0.88

CloudLet ID	Time Shared VM Scheduling		Space Shared VM Scheduling	
	Waiting time	Turnaround Time	Waiting time	Turnaround Time
0	0.1	1.17	0.1	1.17
1	0.1	1.17	0.1	1.17
2	0.1	0.78	0.1	0.78
3	0.1	0.78	0.1	0.78
4	0.1	1.17	0.1	1.17
5	0.1	1.17	0.1	1.17
6	0.1	0.78	0.1	0.78
7	0.1	0.78	0.1	0.78
8	0.1	1.17	0.1	1.17
9	0.1	1.17	0.1	1.17
Average	**0**	**1**	**0**	**1**

CloudLet ID	Time Shared Task CloudLet Scheduling		Space Shared Task CloudLet Scheduling	
	Waiting time	Turnaround Time	Waiting time	Turnaround Time
0	0.1	1.17	0.1	0.78
1	0.1	1.17	0.1	0.78
2	0.1	0.78	0.1	0.78
3	0.1	0.78	0.1	0.78
4	0.1	1.17	0.1	0.78
5	0.1	1.17	0.1	0.78
6	0.1	0.78	0.1	0.78
7	0.1	0.78	0.1	0.78
8	0.1	1.17	0.88	0.78
9	0.1	1.17	0.88	0.78
Average	**0**	**1**	**0**	**1**

DataCenter variables
- Number of DataCenters: 1
- Number of Virtual Machines: 6

CloudLet Parameters
- Number of CloudLets: 10 (Length of Table to present)
  - Remember that program iteration starts from zero(0) [0,1,2..]
- (long) length: 780
- (long) fileSize: 200
- (long) outPutSize: 200
- (int) pesNumber: 1

Virtual Machine Parameters
- Size of VM(MB): 875
- Memory of VM(MB): 1000
- Bandwidth: 5000
- pesNumber(CPUs): 2

Host Parameters (From DataCenter createDatacenter)
- Cloud Hosts: 2(From DataCenter; 'hostlist')
- Cloud User: 2(num_user main method)
- Processing Speed(MIPS): 1000
- CPU/Host[HostList]: 4
  - E.g: peList2.add(new Pe(0, new PeProvisionerSimple(mips)));
- Host Memory[*ram*](MB): 2048
- Host Storage: 1000000
- Host Bandwidth: 10000
- CPU/VM: 2(num_user['grid users'] from Main method)

CloudSim System Parameters
- System Architecture: "x86"
- Operating System: "Linux"
- Time Zone(Resource): 8.0
- Processing Cost: 2.4
- Memory Cost: 1.00
- Storage Cost: 0.50
- Bandwidth Cost: 0.0

References

- Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, *41*(1), 23-50.
- Chowdhury, M. R., Mahmud, M. R., & Rahman, R. M. (2015). Implementation and performance analysis of various VM placement strategies in CloudSim. *Journal of Cloud Computing*, *4*(1), 1-21.
- Pratap, R., & Zaidi, T. (2018, August). Comparative study of task scheduling algorithms through cloudsim. In *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 397-400). IEEE.
- Joshi, A. S., & Munisamy, S. D. (2020). Dynamic degree balanced with CPU based VM allocation policy for load balancing. *Journal of Information and Optimization Sciences*, *41*(2), 543-553.