Cloud Computing Experiment 3,4,5,6

Cloud Computing (University of Delhi)

Mounil Shah (181070058)
Shivani Pawar (181071046)
Ketaki Urankar (181071069)
Sunandan Sharma (181070064)

Final Year BTech Computer Department

Experiment 3

Aim:

To create x datacenters with x hosts, x VMs, x cloudlets in CloudSim.

Theory:

CloudSim

CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services. It is developed by the CLOUDS Lab organization and is written entirely in Java. It is used for modeling and simulating a cloud computing environment as a means for evaluating a hypothesis prior to software development in order to reproduce tests and results.

For example, if you were to deploy an application or a website on the cloud and wanted to test the services and load that your product can handle and also tune its performance to overcome bottlenecks before risking deployment, then such evaluations could be performed by simply coding a simulation of that environment with the help of various flexible and scalable classes provided by the CloudSim package, free of cost.

Following are the benefits of CloudSim:

- **No capital investment involved**. With a simulation tool like CloudSim there is no installation or maintenance cost.
- **Easy to use and Scalable**. You can change the requirements such as adding or deleting resources by changing just a few lines of code.
- **Risks can be evaluated at an earlier stage**. In Cloud Computing utilization of real testbeds limits the experiments to the scale of the testbed and makes the reproduction of results an extremely difficult undertaking.

With simulation, you can test your product against test cases and resolve issues before actual deployment without any limitations.

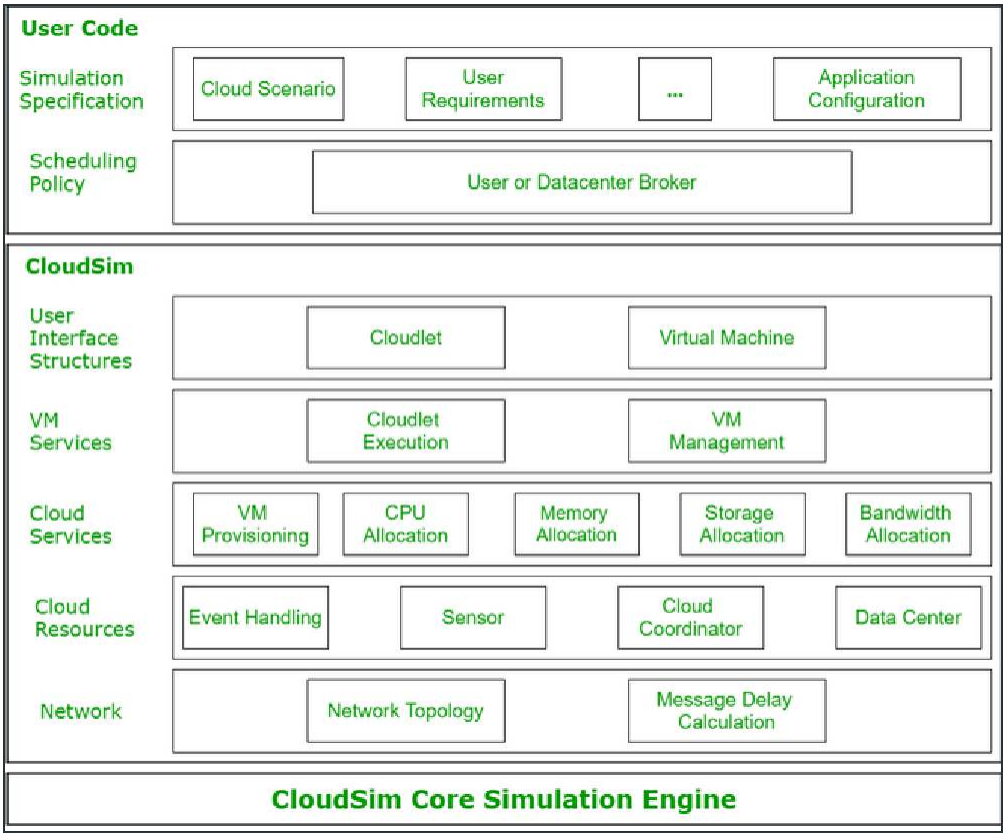
- **No need for try-and-error approaches**. Instead of relying on theoretical and imprecise evaluations which can lead to inefficient service performance and revenue generation, you can test your services in a repeatable and controlled environment free of cost with CloudSim.

Why use CloudSim?

Below are a few reasons to opt for CloudSim:

- Open source and free of cost, so it favors researchers/developers working in the field.
- Easy to download and set-up.
- It is more generalized and extensible to support modeling and experimentation.
- Does not require any high-specs computer to work on.
- Provides pre-defined allocation policies and utilization models for managing resources, and allows implementation of user-defined algorithms as well.
- The documentation provides pre-coded examples for new developers to get familiar with the basic classes and functions.
- Tackle bottlenecks before deployment to reduce risk, lower costs, increase performance, and raise revenue.

CloudSim Architecture:



CloudSim Layered Architecture

CloudSim Core Simulation Engine provides interfaces for the management of resources such as VM, memory and bandwidth of virtualized Datacenters.

CloudSim layer manages the creation and execution of core entities such as VMs, Cloudlets, Hosts etc. It also handles network-related execution along with the provisioning of resources and their execution and management.

User Code is the layer controlled by the user. The developer can write the requirements of the hardware specifications in this layer according to the scenario.

Some of the most common classes used during simulation are:

- **Datacenter**: used for modeling the foundational hardware equipment of any cloud environment, that is the Datacenter. This class provides methods to specify the functional requirements of the Datacenter as well as methods to set the allocation policies of the VMs etc.
- **Host**: this class executes actions related to management of virtual machines. It also define[illegible]al machines, as

well as allocating CPU cores to the virtual machines.

- **VM**: this class represents a virtual machine by providing data members defining a VM's bandwidth, RAM, mips (million instructions per second), size while also providing setter and getter methods for these parameters.
- **Cloudlet**: a cloudlet class represents any task that is run on a VM, like a processing task, or a memory access task, or a file updating task etc. It stores parameters defining the characteristics of a task such as its length, size, mi (million instructions) and provides methods similar to VM class while also providing methods that define a task's execution time, status, cost and history.
- **DatacenterBroker**: is an entity acting on behalf of the user/customer. It is responsible for functioning of VMs, including VM creation, management, destruction and submission of cloudlets to the VM.
- **CloudSim**: this is the class responsible for initializing and starting the simulation environment after all the necessary cloud entities have been defined and later stopping after all the entities have been destroyed.

Features of CloudSim:

CloudSim provides support for simulation and modeling of:

1. Large scale virtualized Datacenters, servers and hosts.
2. Customizable policies for provisioning host to virtual machines.
3. Energy-aware computational resources.
4. Application containers and federated clouds (joining and management of multiple public clouds).
5. Datacenter network topologies and message-passing applications.
6. Dynamic insertion of simulation entities with stop and resume of simulation.
7. User-defined allocation and provisioning policies.

Experiment:

```
arch@cloud ~/cloudsim3/cloudsim-3.0.3 16s
( java -classpath jars/cloudsim-3.0.3.jar:examples examples/org/cloudbus/cloudsim/examples/CloudSimExample6.java
Starting Cloud Computing Experiment 3

Enter number of datacenters: 2
Enter name of datacenter 0: Datacenter0
Enter number of hosts: 2
Enter name of datacenter 1: Datacenter1
Enter number of hosts: 2
Enter number of VMs: 2
Enter number of cloudlets for VM 0: 2
Enter number of cloudlets for VM 1: 2

Initialising...
Starting CloudSim version 3.0
Datacenter0 is starting...
Datacenter1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter0
0.0: Broker: Trying to Create VM #1 in Datacenter0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
1.1: Broker: Cloudlet 0 received
1.1: Broker: Cloudlet 1 received
1.1: Broker: All Cloudlets executed. Finishing...
1.1: Broker: Destroying VM #0
1.1: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter0 is shutting down...
Datacenter1 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time
    0        SUCCESS        2            0            1        0.1            1.1
    1        SUCCESS        2            1            1        0.1            1.1
Cloud Computing Experiment 3 finished
```

Created 2 datacenters with 2 hosts, 2 VMs and 2 cloudlets

Conclusion:

We successfully studied CloudSim and performed the experiment to create x datacenters with x hosts ,x VMs, x cloudlets in CloudSim.

Experiment 4

Aim:

To create x datacenter with x hosts (x per data center) Set storage capacity, ram and cost using Cloud Reports.

Theory:

CloudReports is a graphic tool that simulates distributed computing environments based on the Cloud Computing paradigm. It uses CloudSim as its simulation engine and provides an easy-to-use user interface, report generation features and creation of extensions in a plugin fashion.

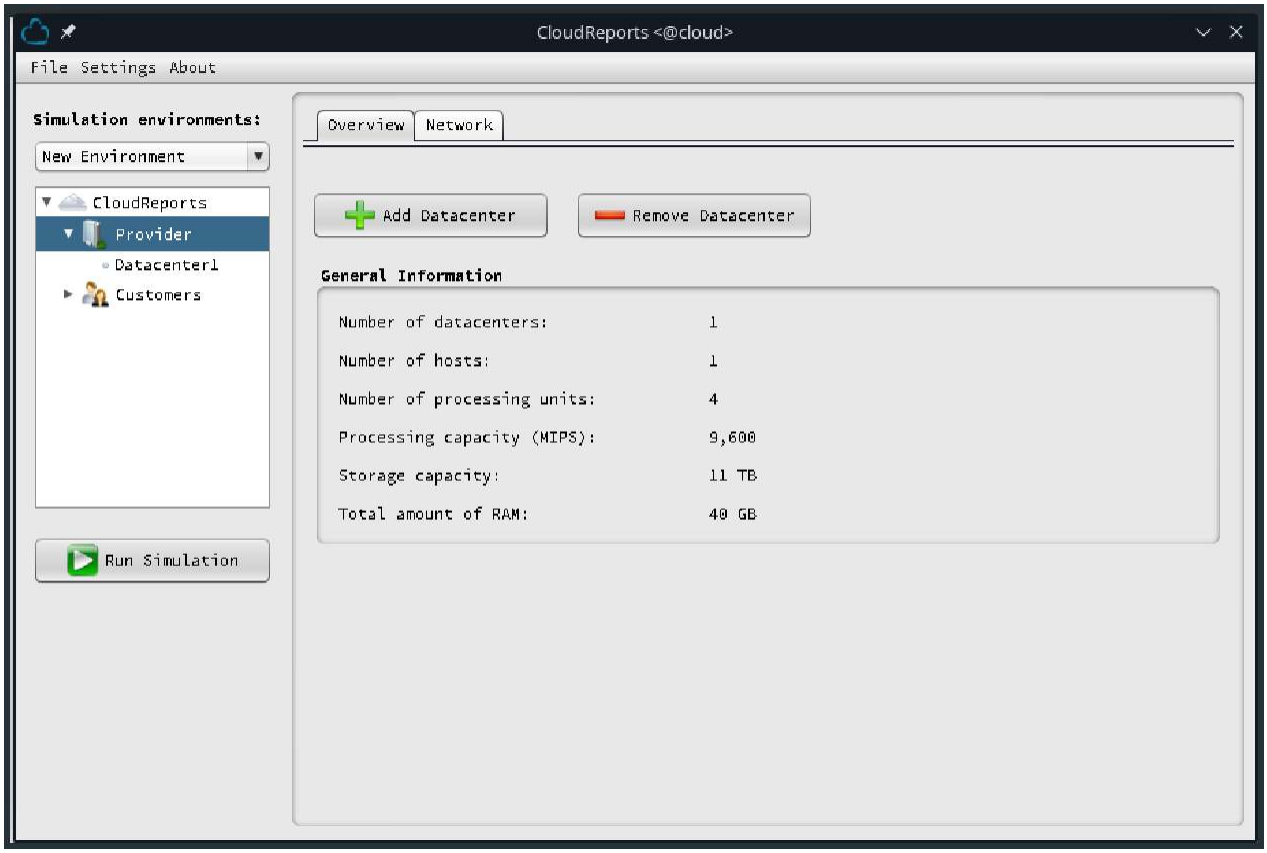
The application simulates an Infrastructure as a Service (IaaS) provider with an arbitrary number of datacenters. Each datacenter is entirely customizable. The user can easily set the amount of computational nodes (hosts) and their resource configuration, which includes processing capacity, amount of RAM, available bandwidth, power consumption and scheduling algorithms.

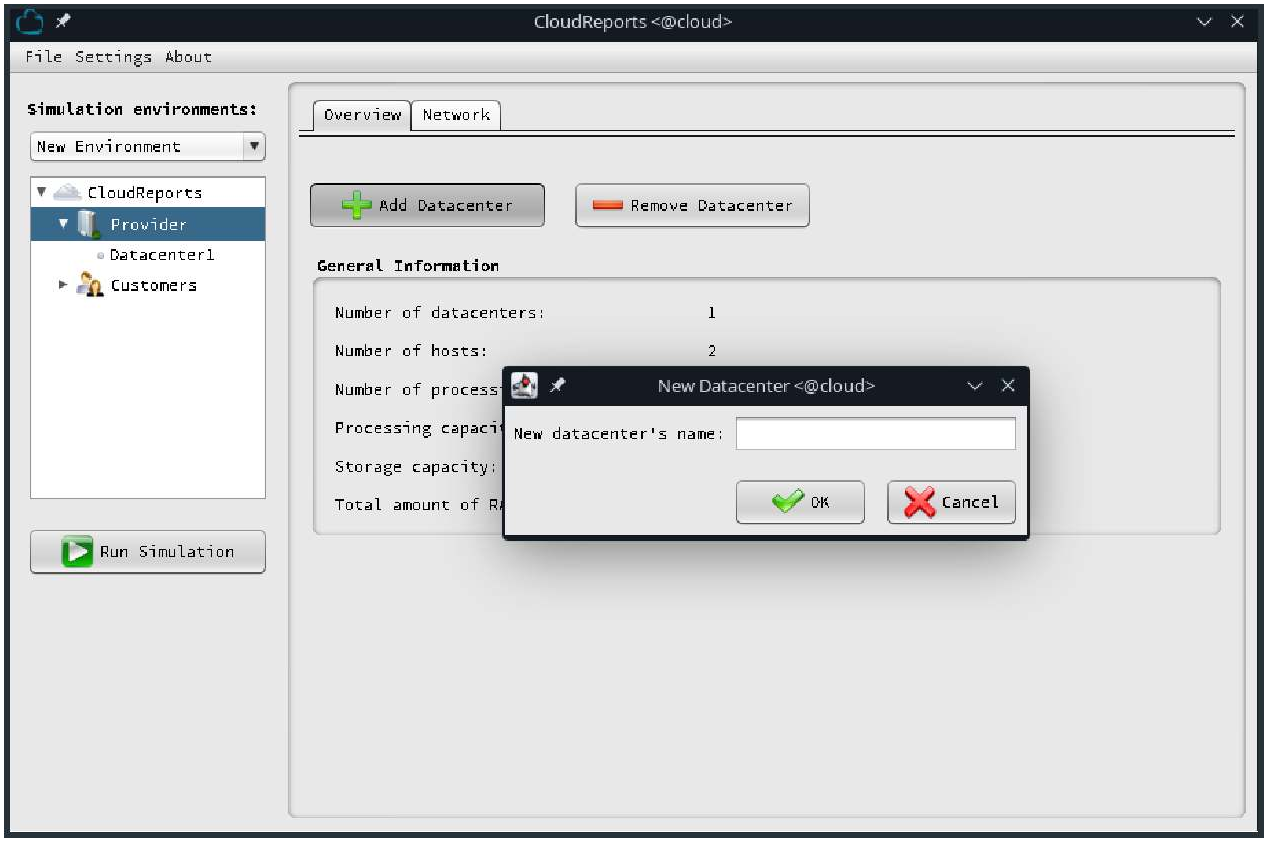
The customers of the IaaS provider are also simulated and entirely customizable. The user can set the number of virtual machines each customer owns, a broker responsible for allocating these virtual machines and resource consumption algorithms. Each virtual machine has its own configuration that consists of its hypervisor, image size, scheduling algorithms for tasks (here known as cloudlets) and required processing capacity, RAM and bandwidth.

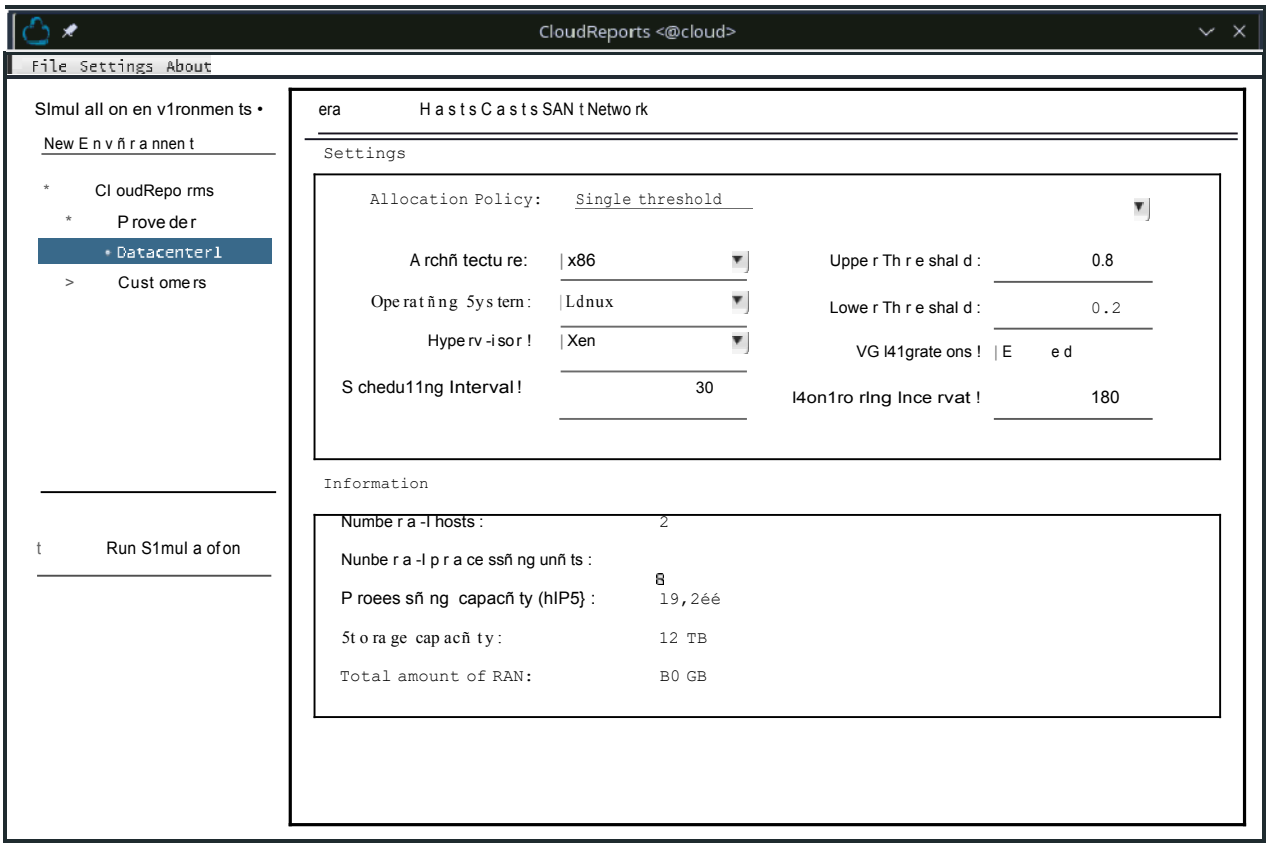
Additionally, CloudReports generates HTML reports of each simulation and raw data files that can be easily imported by third-party applications such as Octave or MATLAB.

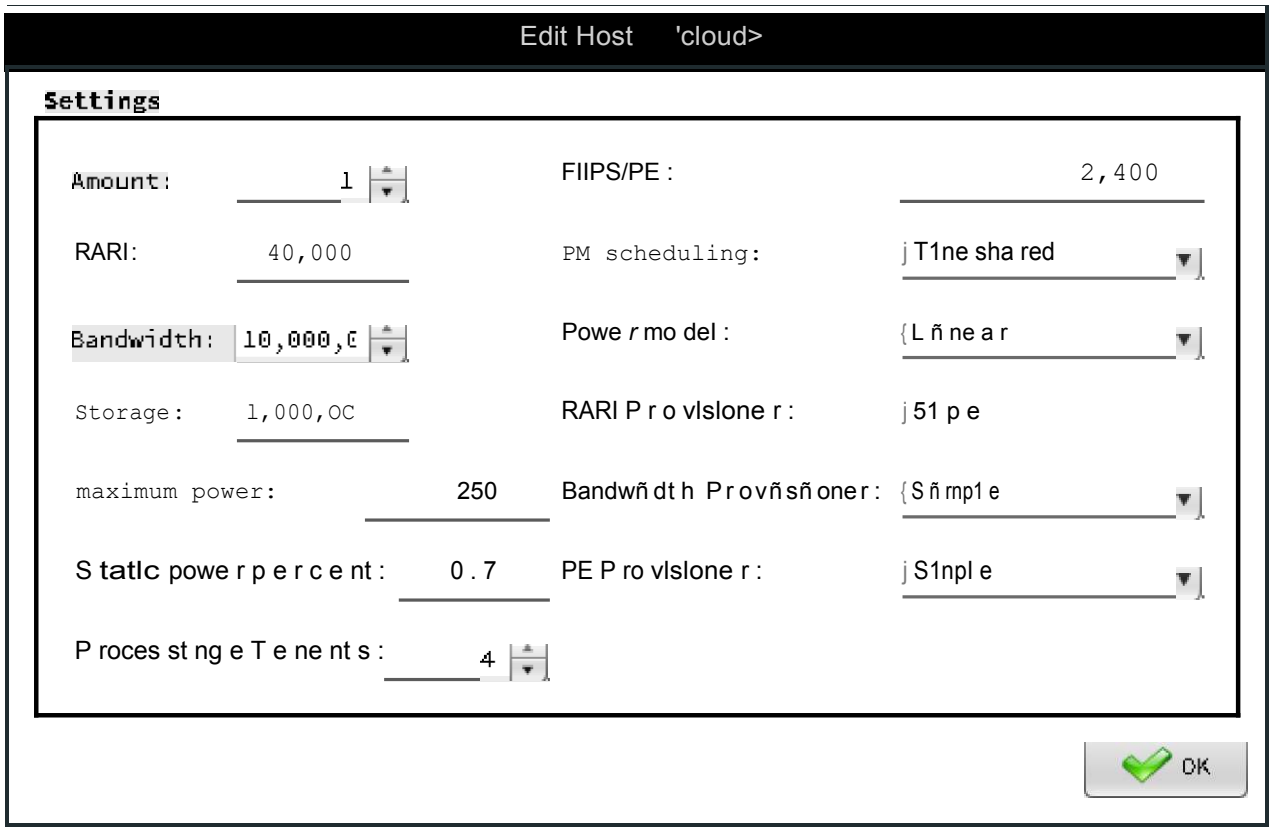
Experiment:

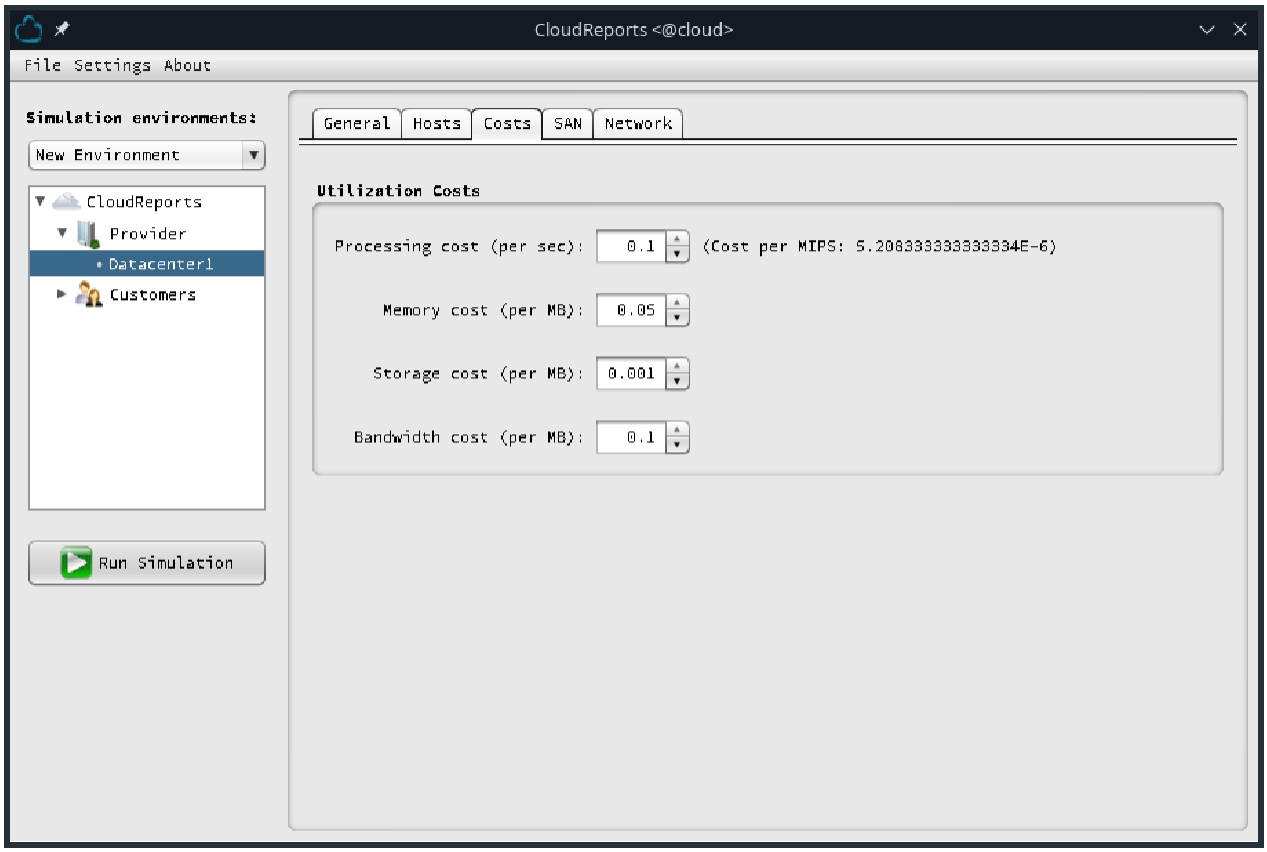
Starting CloudReport

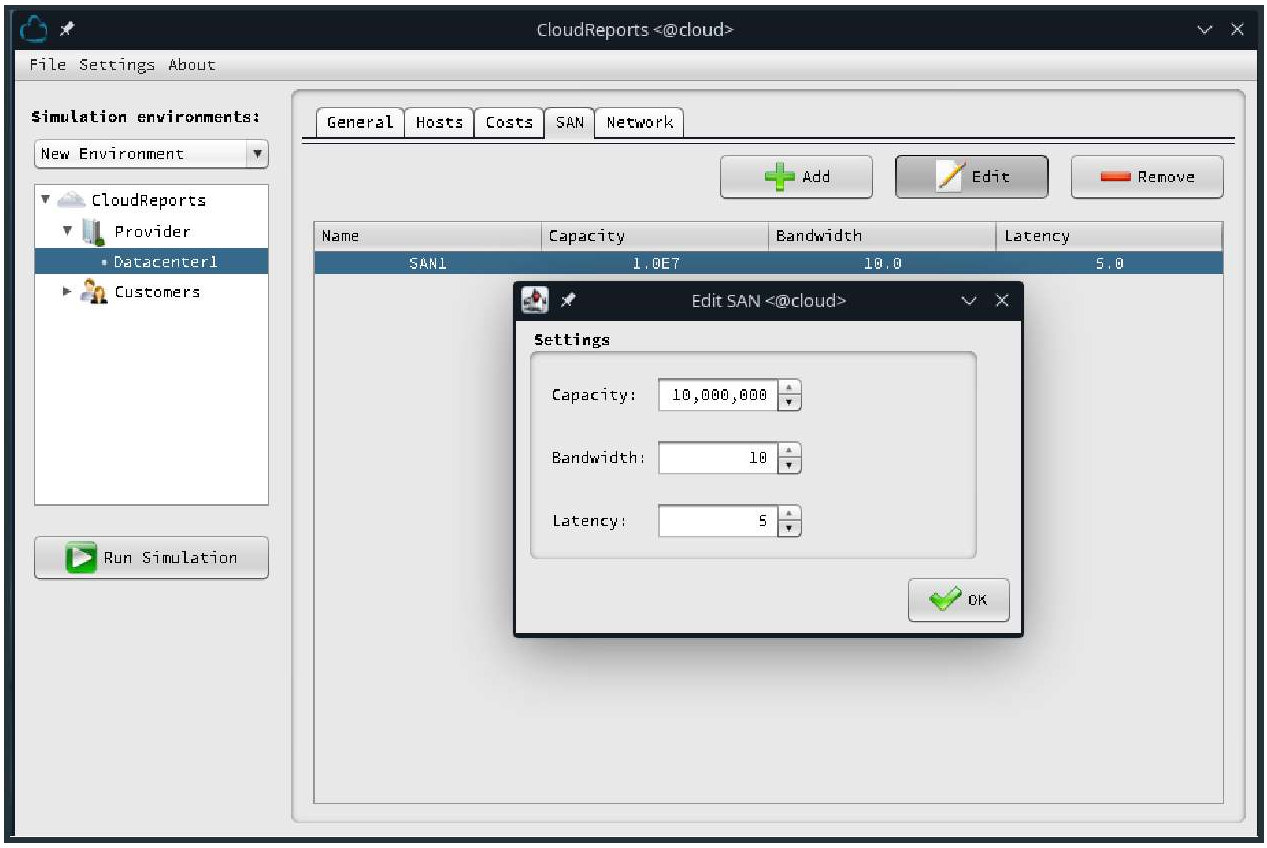


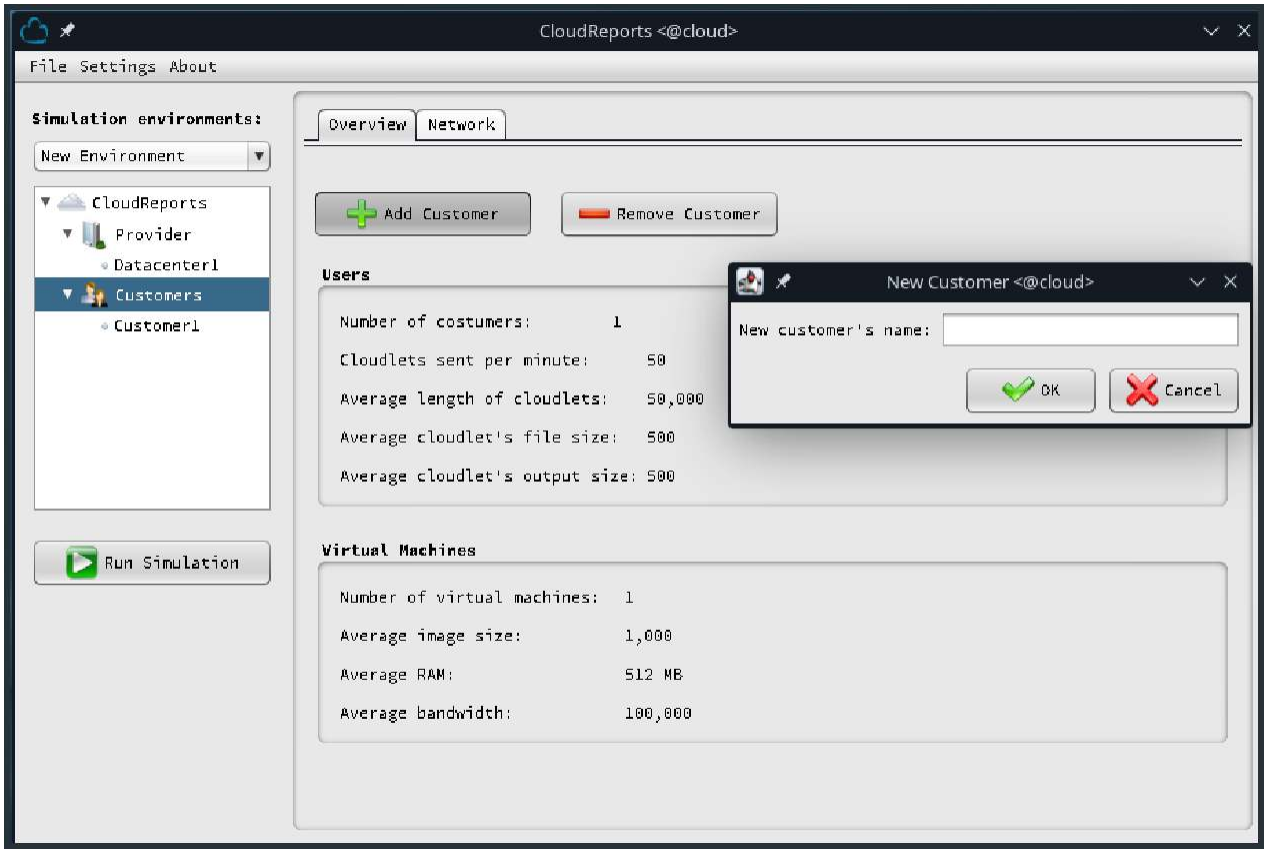


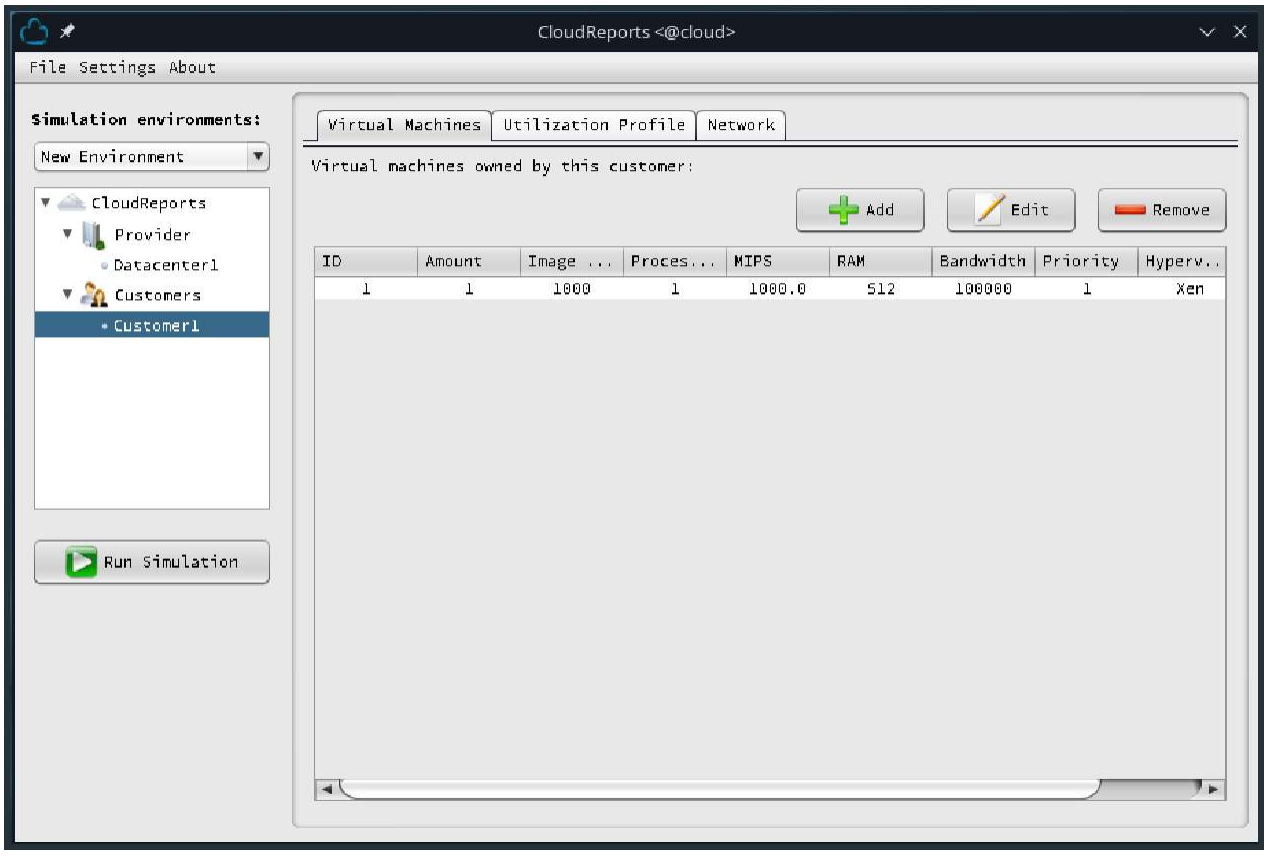


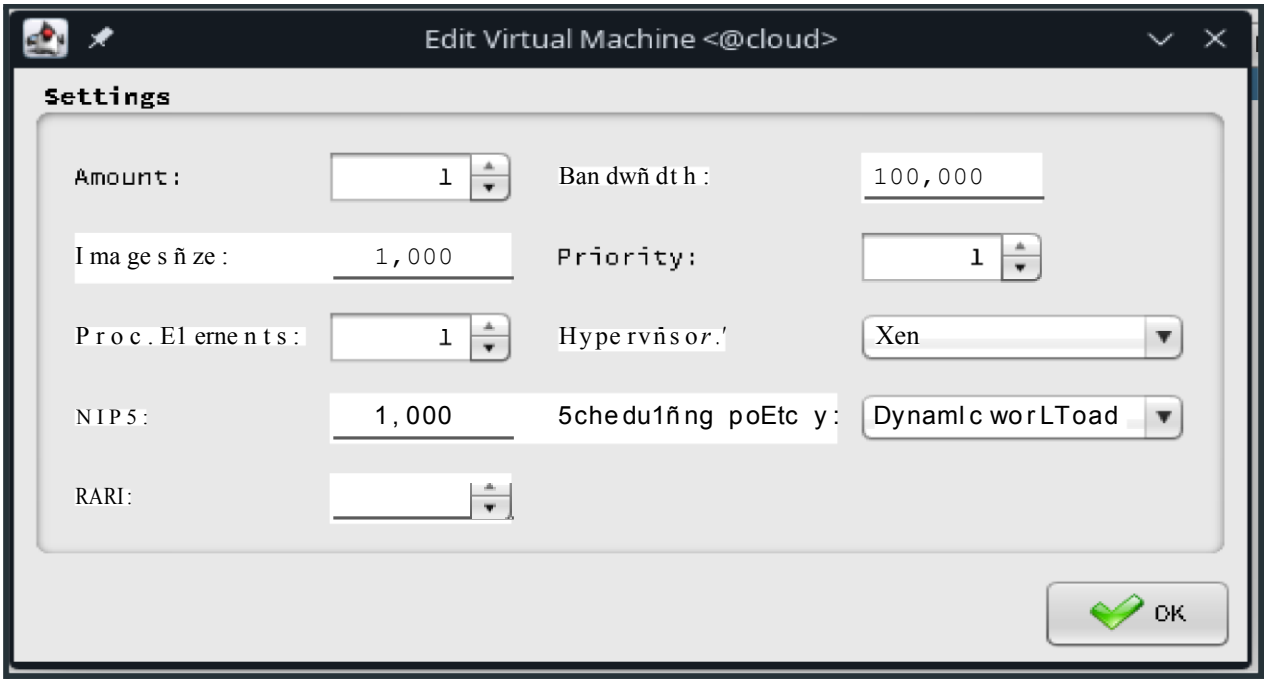


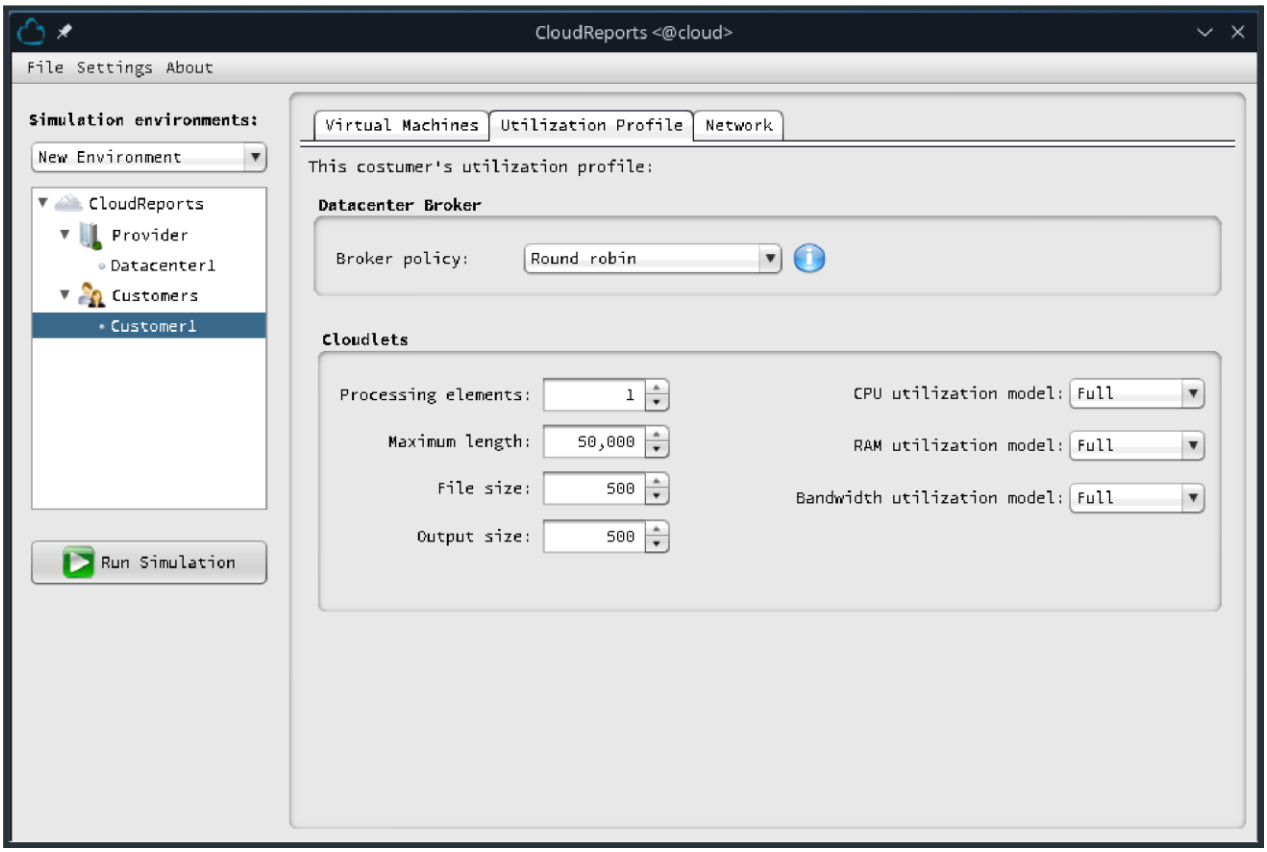












Flue S e t t ñ n gs About

Simulation environments:

e E ñ e

* CloudReports

* Provider

Datacenterl

VI rtual Nachlnes U t1 II za t1 on P raffle Ne two r k

De s c rñ p t ñ a n a f I ñ nks whose sou rce ñ s the s costume r .

Ed-it Think

Destination	Bandwñ d'zh	Latency
Datacenterl	1 , 0	'0, '0

Setts n gs

Bandwidth: 1

Latency: 0

Run Simulaqfiom

OK

File Settings About

SI musatl on envl conmen ts:

e E ñ e

*' C I oudRepo rt s

* P rovl der

• Datacenterl

* Customers

Customerl

General Hosts Costs SAN Network

Add Edit Remove

ID	Amount	VI1 Schedu I h n g	P race ss .	hIPS/PE	PE P r a vñ st one r	Ma
2	1	Tñ ne sha r e d	4	240é.é	Simple	
1	1	Th ne sha r e d	4	2400.é	Simple	

Simulations have completed 9n 01 second .

Run Simulation

Open folder

Close

4- L A '.lc°tC. e 192.168.122.65? '. e

OoudReports

General information

i nr lat a Isa° I' I°° Ie cl in .

D atace nte r1 :

a’.idei

’ +o»e s

Conclusion:

We have successfully studied CloudSim and completed the experiment of creating x data centers with x hosts by setting the storage capacity, ram and cost using cloud reports.

Experiment 5

Aim:

To study and perform Equally Spread Current Execution Load Balancing Policy in Cloud Analyst.

Theory:

CloudAnalyst

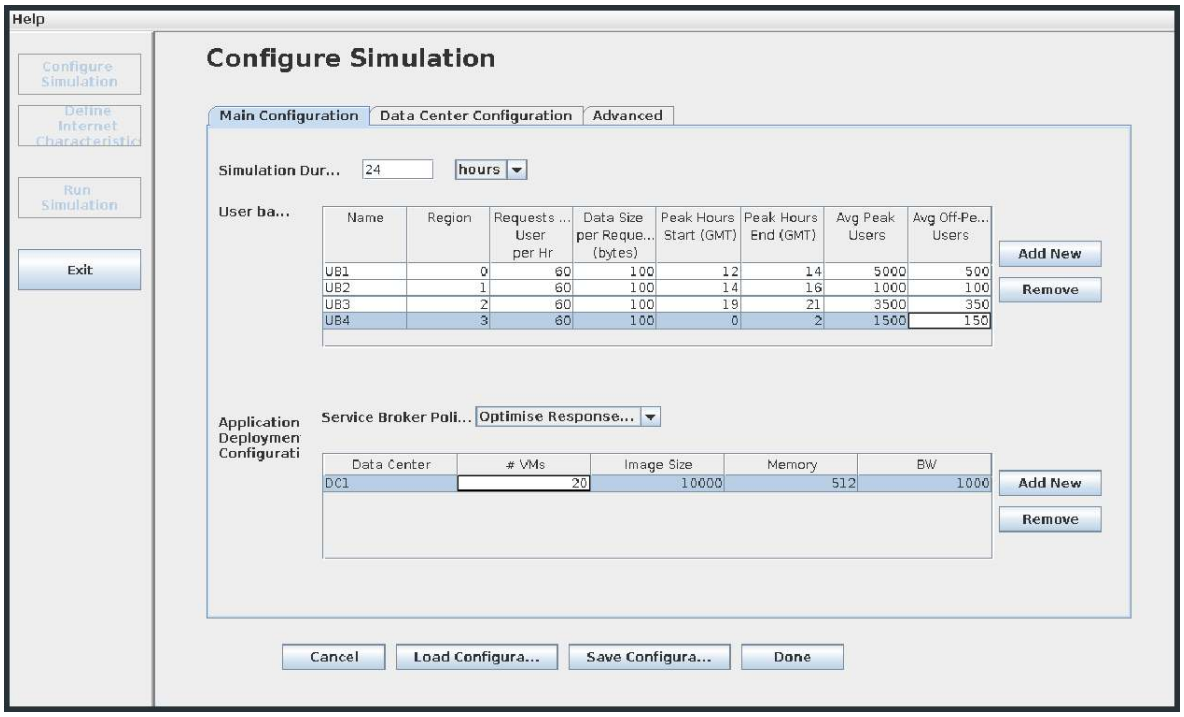
Cloud Analyst is a tool developed at the University of Melbourne whose goal is to support evaluation of social networks tools according to geographic distribution of users and data centers. In this tool, communities of users and data centers supporting the social networks are characterized and, based on their location; parameters such as user experience while using the social network application and load on the data center are obtained/logged.

Load Balancing

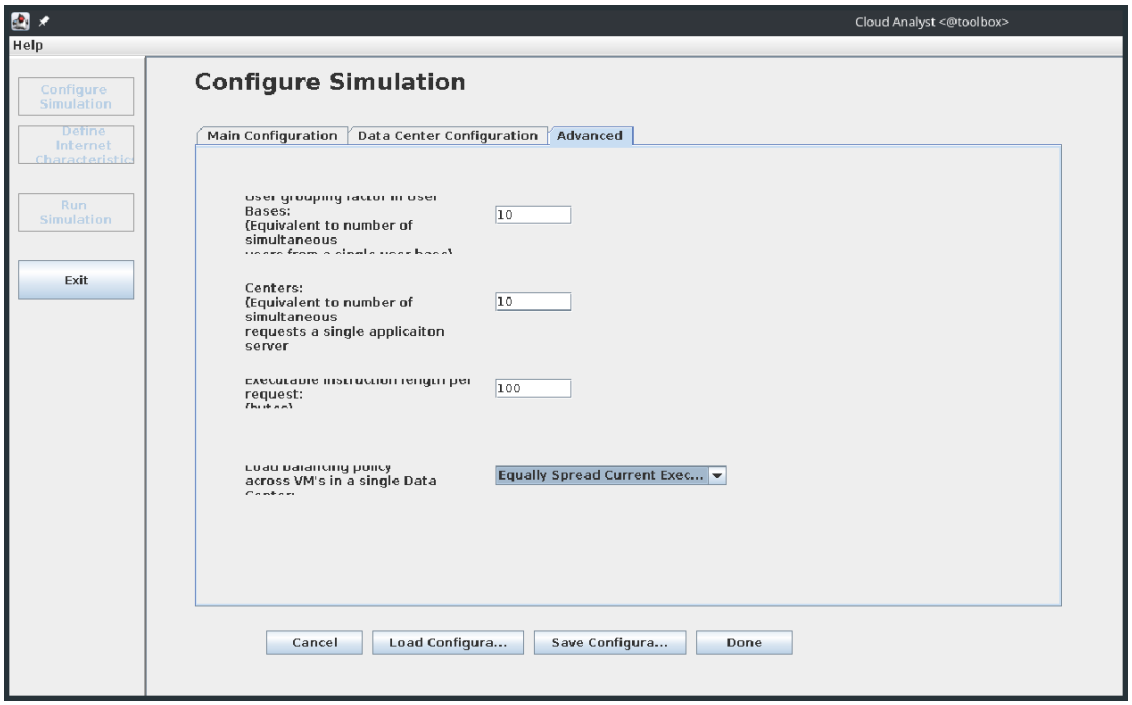
Load balancing of requests distributes workloads across multiple computing resources, such as computers, computer clusters, disk drives, network links, CPU. As technology grows faster, there are huge amounts of users on the internet so managing and fulfilling their requirements, load balancers come into picture which essentially ensure that they spread workload equally to the all available server without any delay which helps to accomplish a high user satisfaction, Maximum throughput with minimum response time. Load balancer is not only designed for cloud workload but also designed for different purposes like DNS load balancer, database load balancer, website load balancer. Load balancing done on two level in cloud computing:

- VM level mapping done between applications which are uploaded on the cloud to virtual machines, the load balancer assigns the requested VM to physical computers which balance the load of numerous applications among PCs.
- Host level mapping done between virtual machine and host resources which help to proceed multiple incoming requests of application.

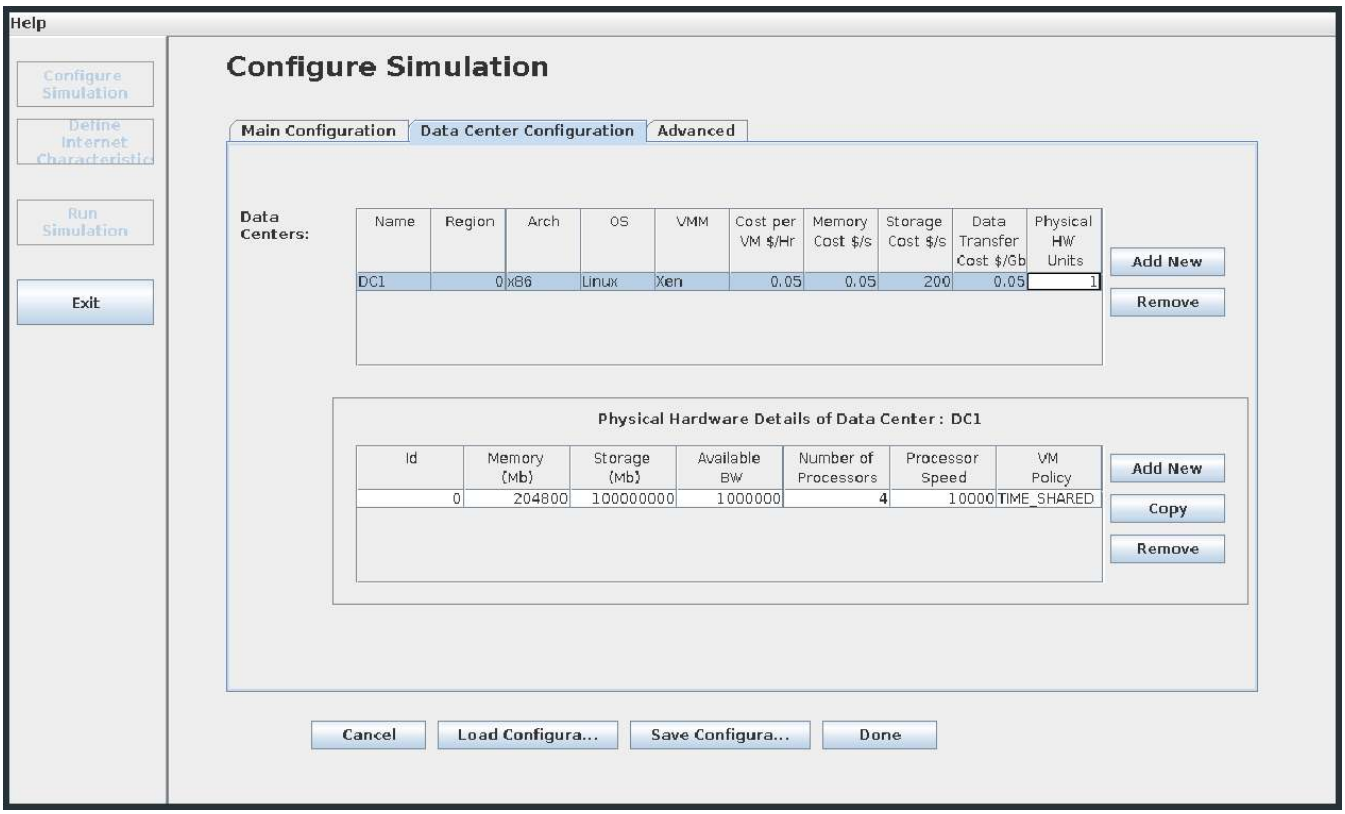
Experiment:



Configuring simulation for 4 UB and 1 datacenter



Selecting Equally Spread Current Execution Load Balancing Policy



Configuring the datacenter

Help

Configure Simulation

Define Internet Characteristics

Run Simulation

Exit

Configure Internet Characteristics

Use this screen to configure the Int ernet characteristics.

Delay Matrix

The transmission delay between regions. Units in milliseconds

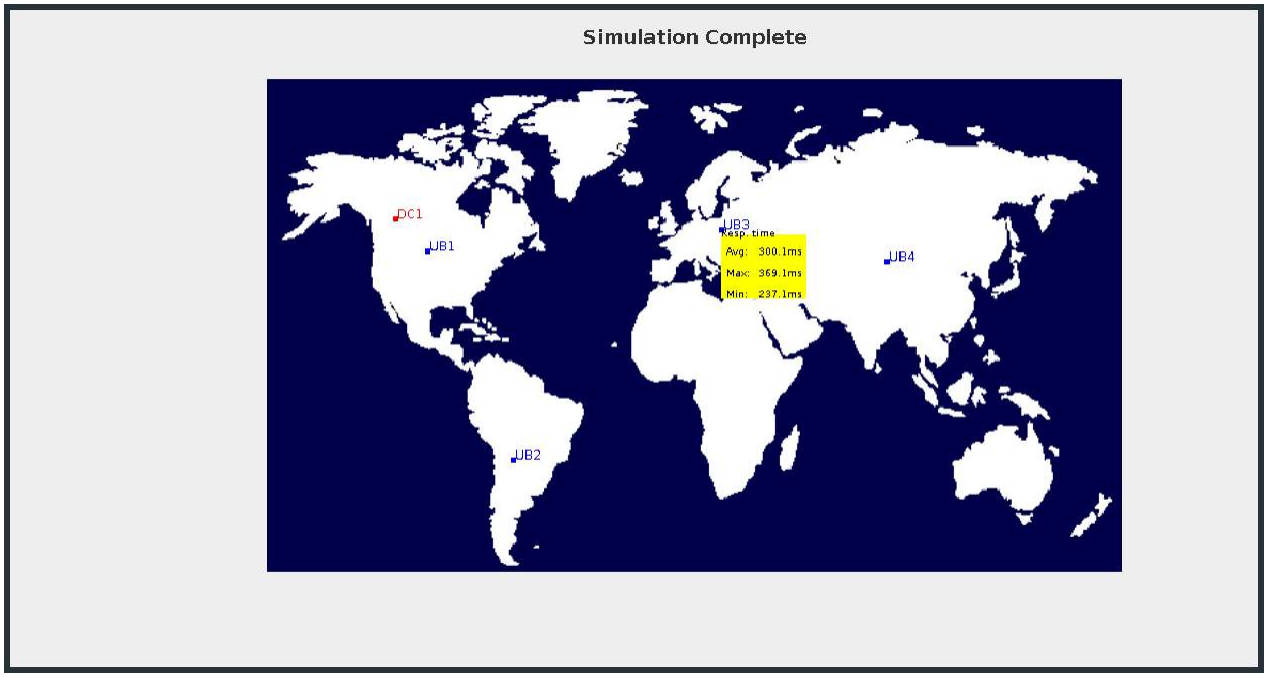
Region\Reqior	0		2	3	4	5
0	25	10C	15C	25C	25C	10C
1	10C	25	25C	50C	350	20C
2	15C	25C	25	15C	15C	20C
3	29C	b0C	UC	29	b0C	50C
4	29C	350	UC	90C	2b	50C
5	10C	20C	20C	90C	b0C	25

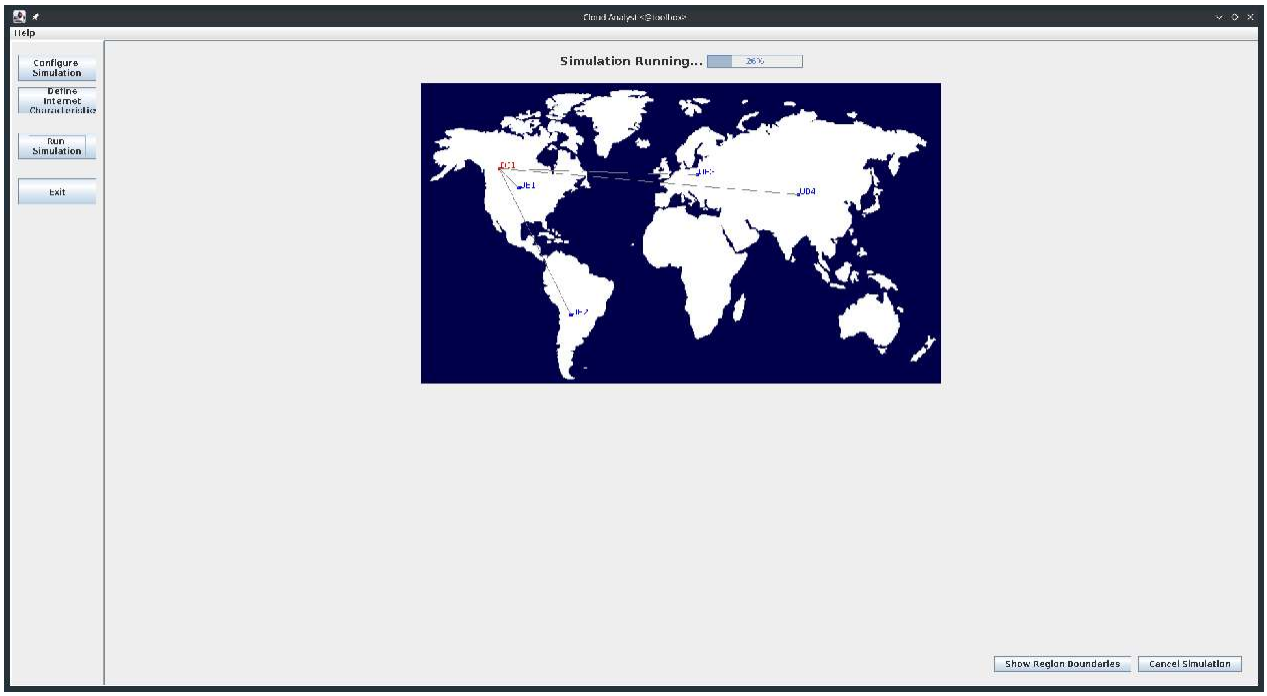
Bandwidth Matrix

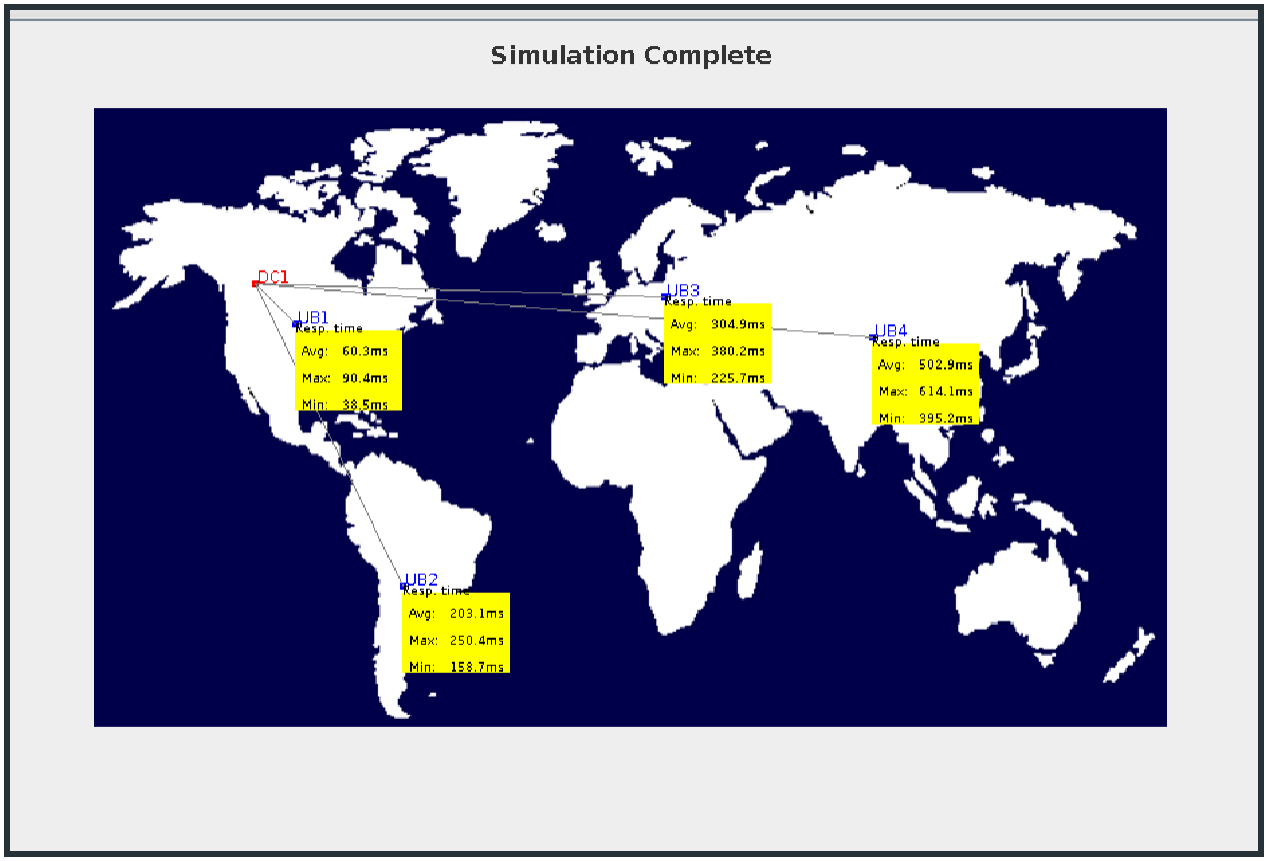
The available bandwidth between regions for the simulated appGcation. Units in Mbps

Re gion\Reg ior	0	1	2	3		
0	2,00C	1,00C	1,00C	1,00C	4	5
1	1,00C	uOC	1,00C	1,00C	1,00C	1,0000
2	1,00C	1,00C	2,90C	1,00C	1,00C	1,0000
3	1,00C	1,00C	1,00C	1,90C	1,00d	1,0000
4	1,00C	1,00C	1,00C	1,00C	90C	1,0000a

Due a el







Completed Simulation for 4 UB and 1 datacenter using Equally Spread Current Execution Load Balancing Policy

Results:

The results of the experiment performed can be found here.

Conclusion:

We successfully studied and performed the Equally Spread Execution Load Balancing Policy and recorded the results with the help of Cloud Analyst.

Experiment 6

Aim:

To study and perform Throttled Load Balancing Policy in Cloud Analyst.

Theory:

CloudAnalyst

Cloud Analyst is a tool developed at the University of Melbourne whose goal is to support evaluation of social networks tools according to geographic distribution of users and data centers. In this tool, communities of users and data centers supporting the social networks are characterized and, based on their location; parameters such as user experience while using the social network application and load on the data center are obtained/logged.

Load Balancing

Load balancing of requests distributes workloads across multiple computing resources, such as computers, computer clusters, disk drives, network links, CPU. As technology grows faster, there are huge amounts of users on the internet so managing and fulfilling their requirements, load balancers come into picture which essentially ensure that they spread workload equally to the all available server without any delay which helps to accomplish a high user satisfaction, Maximum throughput with minimum response time. Load balancer is not only designed for cloud workload but also designed for different purposes like DNS load balancer, database load balancer, website load balancer. Load balancing done on two level in cloud computing:

- VM level mapping done between applications which are uploaded on the cloud to virtual machines, the load balancer assigns the requested VM to physical computers which balance the load of numerous applications among PCs.
- Host level mapping done between virtual machine and host resources which help to proceed multiple incoming requests of application.

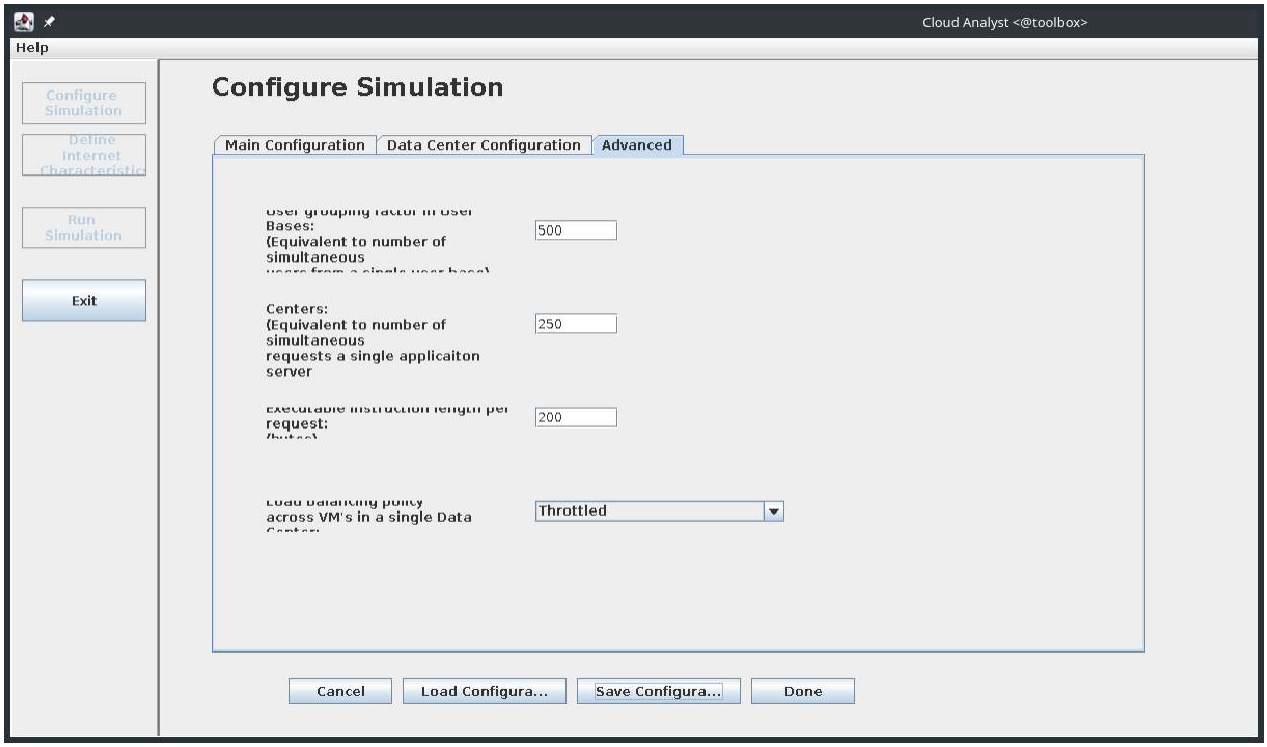
Throttled Load Balancing Policy

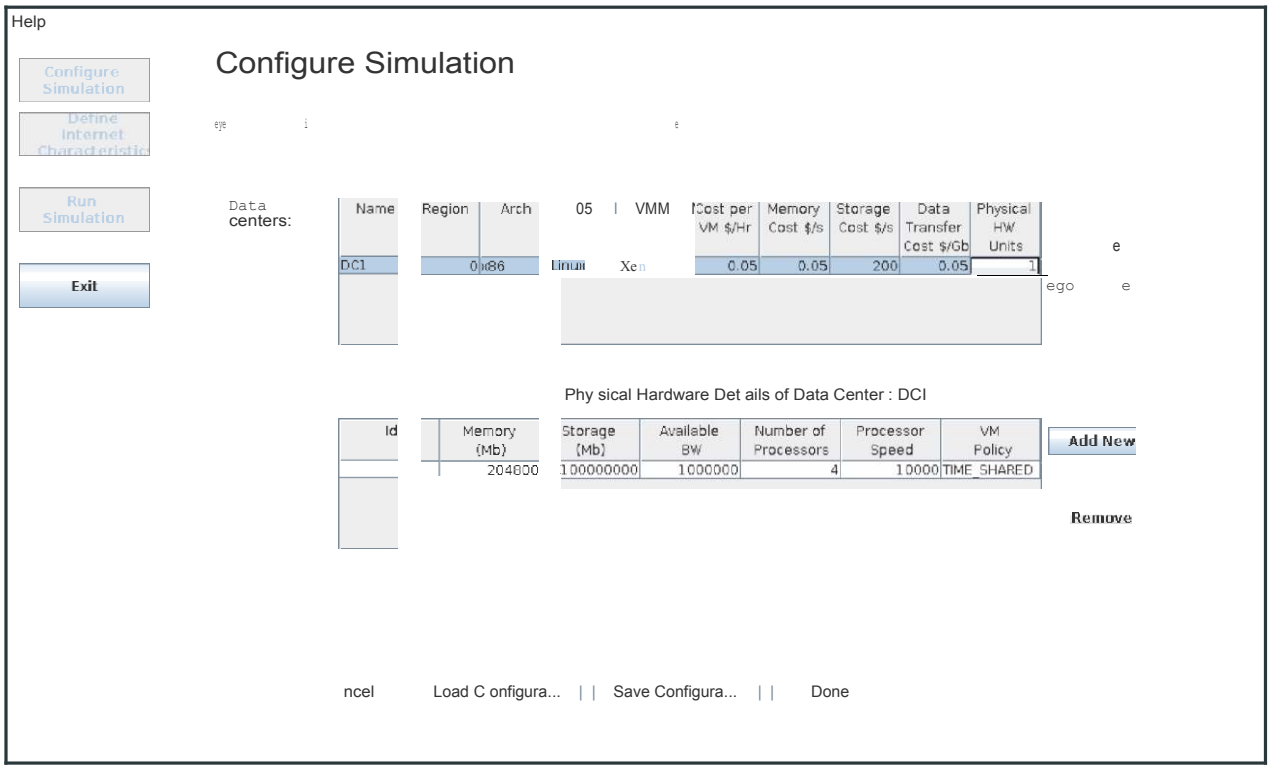
Throttled algorithm is best in performance and in response time compared to existing two policies. It is also dynamic in behavior. It assigns all incoming jobs in an efficient way to the VM. It discovers the suitable virtual machine for assigning a specific job.
The job manager is having a list of all virtual machines, using this indexed list, it allot the desired job to the suitable machine which accesses that load easily and completes the operations. If the job is well suited for a particular virtual machine in terms of size and availability of the virtual machine and that job is assigned to the suitable machine otherwise the job manager waits for the client request and puts the job in queue for fast processing. This algorithm performs well as compared to the round robin algorithm.

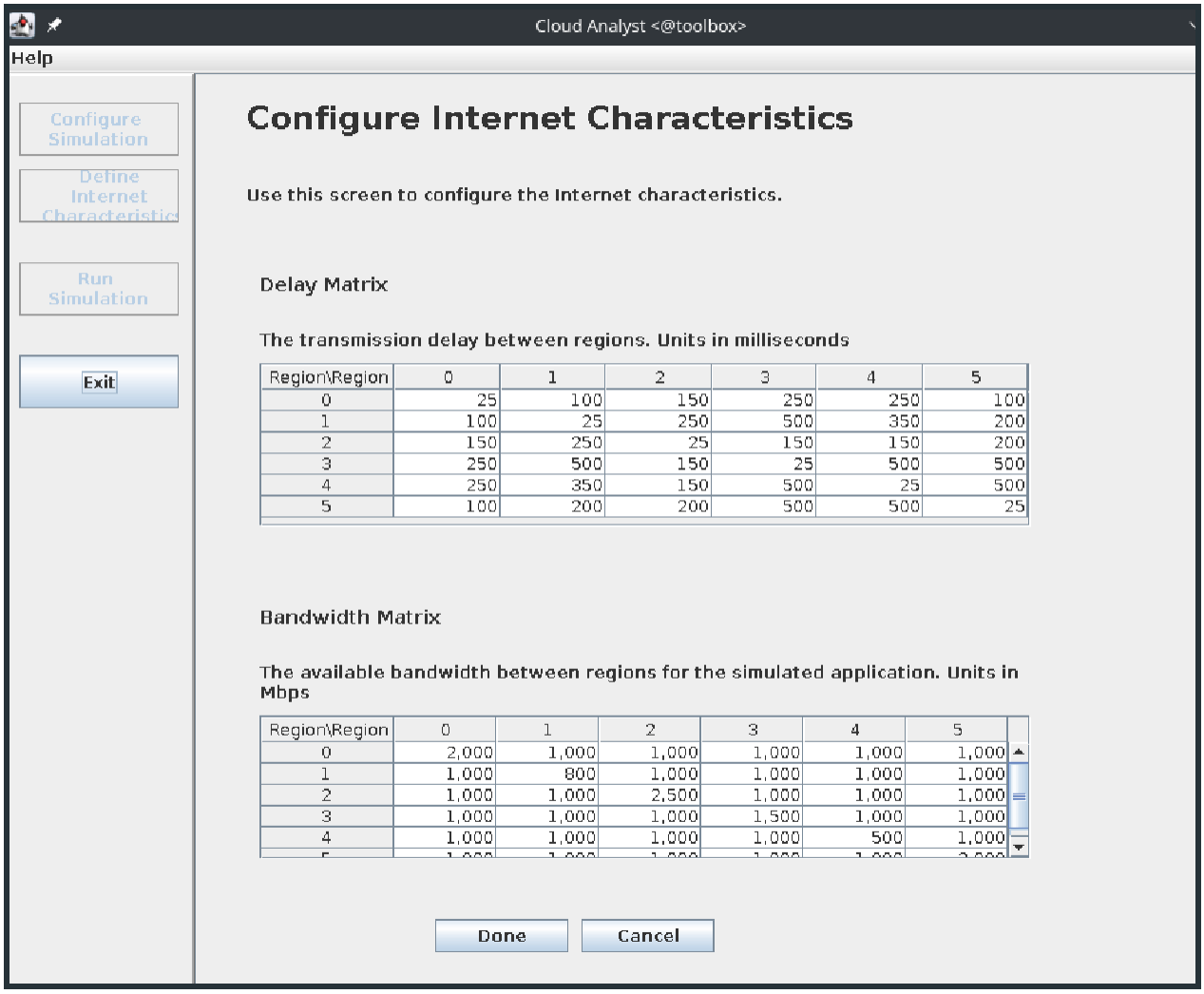
Drawbacks

- In the throttled algorithm, where the index table is resolved from the first index every time when the data center queries load balancer for allocation of VM.
- It does not take into account the advanced load balancing requirements such as processing times for each individual request.

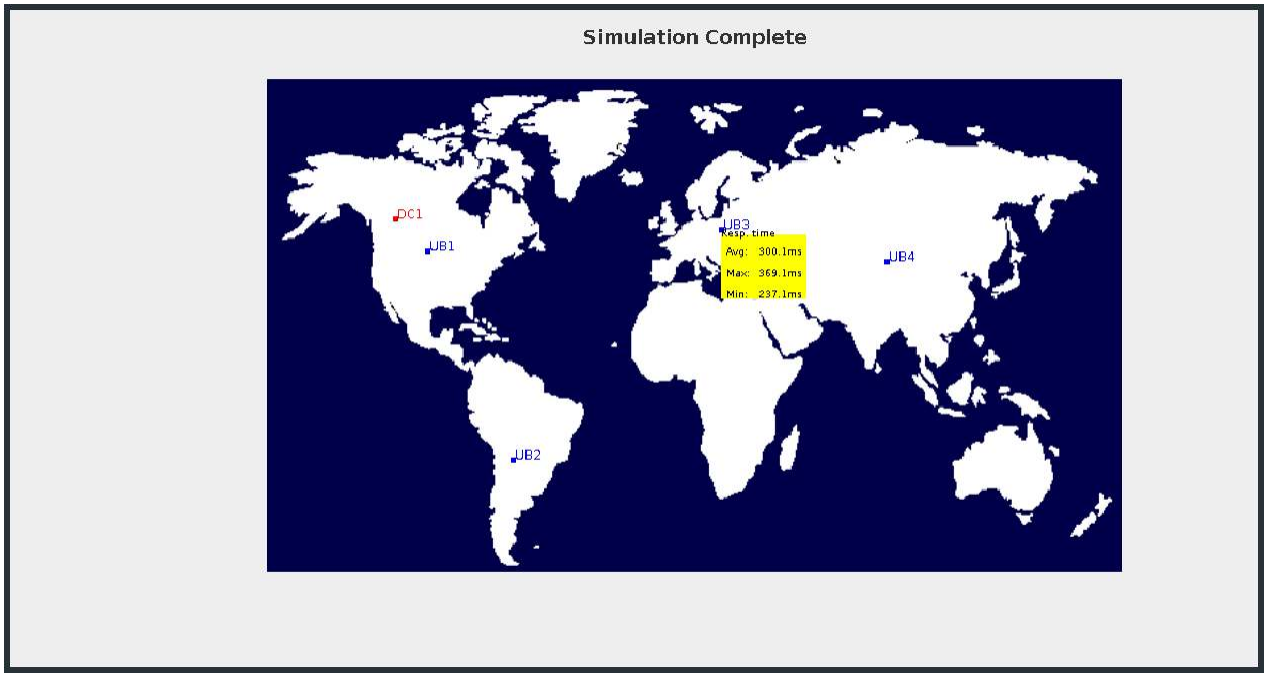
Experiment:

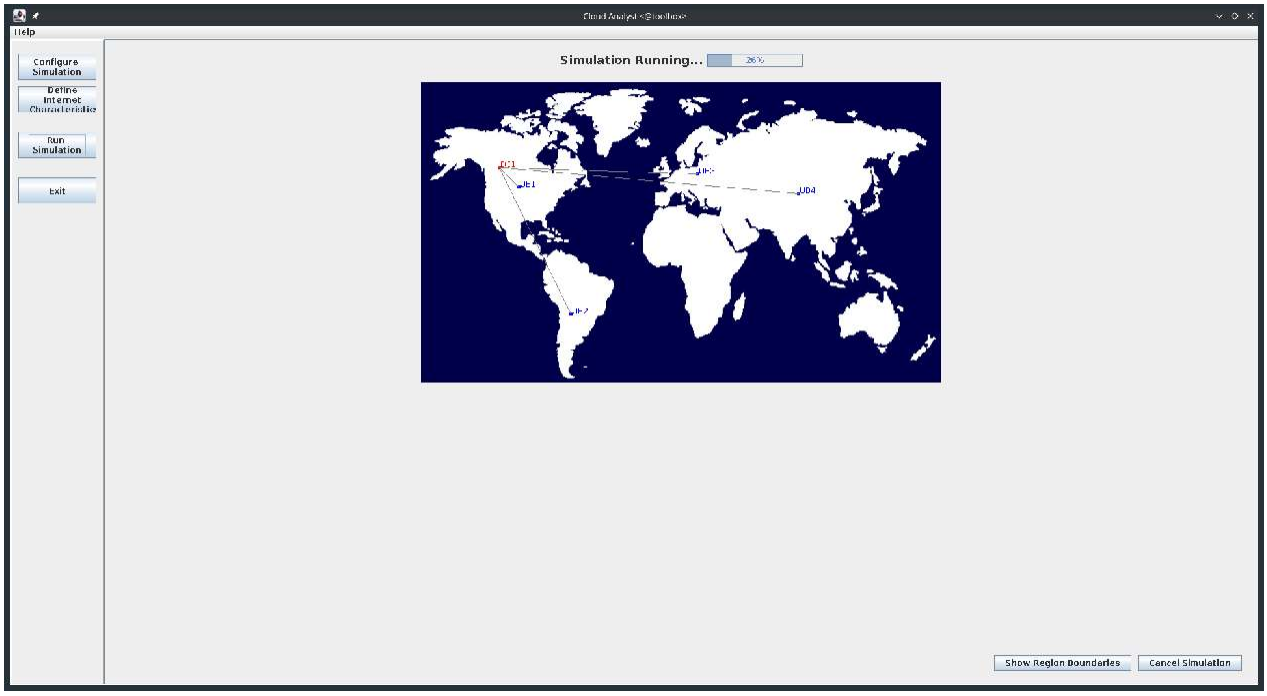


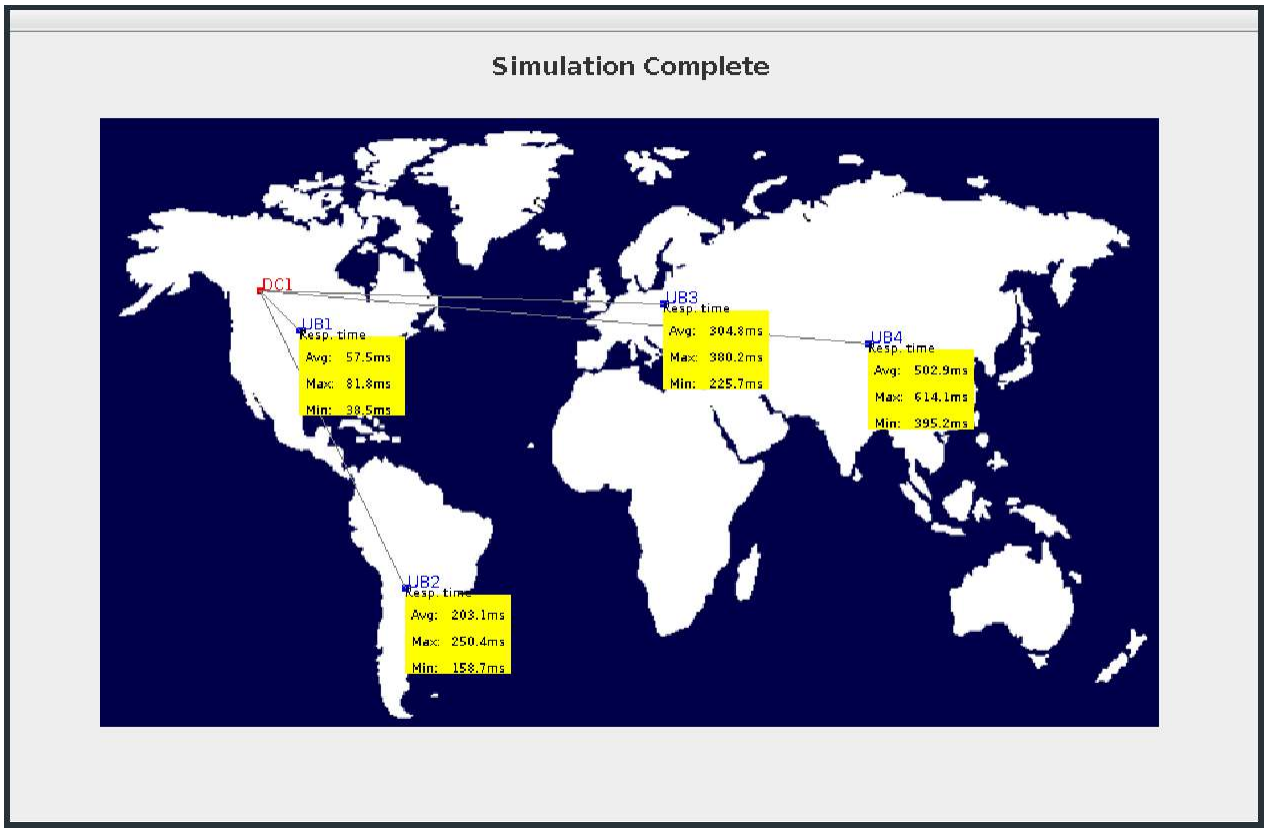




Keeping same configuration as the simulation for Equally Spread Current Execution Load Balancing Policy and selecting Throttled Load Balancing Policy







Completed Simulation for 4 UB and 1 datacenter using Throttled Load Balancing Policy

Results:

The results of the experiment performed can be found here.

Conclusion:

We successfully studied and performed the Throttled Load Balancing Policy and recorded the results with the help of Cloud Analyst.