CloudAnalyst: A CloudSim-based Tool for Modelling and Analysis of Large Scale Cloud Computing Environments

---

MEDC Project Report

Bhathiya Wickremasinghe
(mkbw@pgrad.unimelb.edu.au)
Student No: 318282
22/6/2009

Project Supervisor:
Assoc. Prof. Rajkumar Buyya

Contents

Abstract

The advancement of Cloud technologies in the last few years has opened up new possibilities to Internet applications developers. Previously deployment and hosting of an application was one of the first and main concerns when designing an application for the Internet. But with the advent of the Cloud, now it is possible to solve this problem more economically and more flexibly using the powerful infrastructure services provided by a Cloud service provider on an as-required basis.

In this paper we introduce a novel tool, CloudAnalyst, along with a new approach to simulate such large-scaled applications on the Cloud with the purpose of studying the behaviour of such applications under various deployment configurations. Such a study would benefit the application designers greatly in identifying the optimal configuration for their application. But more importantly such a study will generate valuable insights in to designing Cloud infrastructure services in areas such as coordination between Data Centers, load balancing algorithms and possible value added services such as Service Brokers to coordinate between Data Centers to optimise the application performance and cost to the owners.

1 Introduction

1.1 Background

Cloud computing is a fast growing area in computing research and industry today. It has the potential to make the not so new idea of 'computing as a utility' a reality in the near future.[1]

With the advancement of the Cloud, there are new possibilities opening up on how applications can be built on the Internet. On one hand there are the cloud service providers who are willing to provide large scaled computing infrastructure at a cheaper price which is often defined on usage, eliminating the high initial cost of setting up an application deployment environment, and provide the infrastructure services in a very flexible manner which the users can scale up or down at will. On the other hand there are large scaled software systems such as social networking sites and e-commerce applications gaining popularity today which can benefit greatly by using such cloud services to minimize costs and improve service quality to the end users. But when bringing these two ends together there are several factors that will impact the net benefit such as the distribution (geographic) of the user bases, the available Internet infrastructure within those geographic areas, the dynamic nature of the usage patterns of the user base and how well the cloud services can adapt or dynamically reconfigure itself, etc. Doing a comprehensive study on this overall problem in the real world will be extremely difficult, and the best approach to study such a dynamic and massively distributed environment is through simulation.

There have been many studies using simulation techniques to investigate behaviour of large scale distributed systems such as the GridSim and CloudSim projects at the University of Melbourne. This project investigates into extending these techniques to study the behaviour of large scaled Internet application in a cloud environment and proposes a new simulation tool "**CloudAnalyst**" that can be used for simulating this type of large scaled applications along with a novel approach for such studies.

1.2 Aims of the project

In our initial background research on the topic, it became apparent that there are many good simulation frameworks that can be leveraged to simulate an Internet application on cloud environment. But simulating something of this nature, especially at a large (global) scale is a complex task with many unknowns and many parameters and options that need consideration. Therefore the aim of this project is more to define an approach to such studies rather than do a definitive study on the subject. We propose the tools, algorithms and a generic approach for the study and perform an initial study to demonstrate the benefit of our proposal. But a definitive study on the topic should be the result of using the tools and approach we are proposing over a reasonable period of time, refining and fine tuning the tools and approach along the way.

The frontrunner simulating Cloud environments is the CloudSim toolkit being developed at the GRIDS laboratory at the University of Melbourne. But some extensions are required to CloudSim to simulate a large scaled Internet application. It also became apparent that rather than just extending the CloudSim toolkit as a toolkit, it will be more beneficial to build a simulation tool using CloudSim, separating the simulation experimentation from a programming task. Such a tool will enable users to quickly set up simulations and summarize results in useful formats, and will appeal to a wider audience. Then the experiences and feedback from the users can be used to great effect to improve the framework as a tool as well as an approach.

Therefore the aims of the project can be summarised as below:

- Investigate in to existing simulation techniques for studying large scale distributed systems
- Leverage suitable existing simulation techniques and tools and define an approach for

effectively simulating large scaled Internet applications on Cloud environment
- Develop the initial version of a tool required for such an approach with a sufficiently flexible design that leaves room for further refinement and extension
- Perform initial experiments using the tools and approach

1.3 Related Work

1.3.1 CloudSim [2]

CloudSim is a framework developed by the GRIDS laboratory of University of Melbourne which enables seamless modelling, simulation and experimenting on designing Cloud computing infrastructures. CloudSim is a self-contained platform which can be used to model data centers, service brokers, scheduling and allocation policies of a large scaled Cloud platform. It provides a virtualization engine with extensive features for modelling the creation and life cycle management of virtual engines in a data center. CloudSim framework is built on top of GridSim framework also developed by the GRIDS laboratory.

The CloudAnalyst is built directly on top of CloudSim framework leveraging the features of the original framework and extending some of the capabilities of CloudSim.

1.3.2 GridSim [3]

GridSim toolkit was developed by Buyya et al to address the problem of near impossibility of performance evaluation of real large scaled distributed environments (typically Grid systems but also P2P networks) in a repeatable and controlled manner. The GridSim toolkit is a Java based simulation toolkit that supports modelling and simulation of heterogeneous Grid resources and users spread across multiple organizations with their own policies. It supports multiple application models and provides primitives for creation of application tasks, mapping of tasks to resources and managing such tasks and resources.

1.3.3 simjava [4]

SimJava is the underlying event based simulation toolkit used in both CloudSim and GridSim.

1.4 Terminology and Abbreviations

Data Transmission Latency	This document uses "Transmission Latency" to mean the network delay (based on geographical distance, operation of network equipment etc.) between two points. This can be considered equivalent to half of the ping round-trip time.
Data Transfer Time	Data Transfer Time is the time taken by a given amount of data to be transported from one point to another. This is taken to be equivalent to the available bandwidth divided by the size of the unit of data.
Response Time	The time taken by an Internet application defined as the time interval between sending the request and receiving a response
VM	Virtual Machine
VMM	Virtual Machine Monitor

The rest of the document is organised as follows: In Chapter 2, we introduce the tool we are proposing, "CloudAnalyst" and discuss the features and benefits of such a tool. In Chapter 3 we provide the detailed technical design of the simulator and Chapter 4 provides a brief user guide to the tool

Then in Chapter 5 we show how the simulator can be applied to studying a large-scaled application on the Cloud such as a social networking site and demonstrate how various scenarios can be analysed using the new simulation tool. The detailed results of the simulations are produced with comparisons and detailed analysis. Then we conclude this report in with Chapter 6.

2 CloudAnalyst

2.1 Introduction

As already mentioned there are several extremely good toolkits that can be used to model a simulated environment to study the behaviour of a large scaled application on the Internet. But it became apparent that having an easy to use tool with a level of visualisation capability is even better than just a toolkit. Such a tool separates the simulation experiment set up exercise from a programming exercise and enables a modeller to concentrate on the simulation parameters rather than the technicalities of programming. It also enables the modeller to execute simulations repeatedly with modifications to the parameters quickly and easily. A graphical output of the simulation results enables the results to be analysed more easily and more efficiently and it may also help in quickly highlighting any problems with the performance and accuracy of the simulation logic.

Therefore we decided to develop a simulation tool before starting the experiment.

2.2 Features of the Simulator

There are several highly desirable features of a tool similar to the one described in the above section.

2.2.1 Ease of use

Ease of setting up and executing a simulation experiment is the main point of having a simulation tool. The simulator needs to provide an easy to use graphical user interface which is intuitive yet comprehensive.

2.2.2 Ability to define a simulation with a high degree of configurability and flexibility

Perhaps the most important feature is the level of configurability the tool can provide. A simulation, especially of the nature of modelling something as complex as an Internet Application depends on many parameters and most of the time the values for those parameters need to be assumed. Therefore it is important to be able to enter and change those parameters quickly and easily and repeat simulations.

2.2.3 Graphical output

A picture is said to be worth a thousand words. Graphical output in the form of tables and charts is highly desirable to summarise the potentially large amount of statistics that is collected during the simulation. Such effective presentation helps in identifying the important patterns of the output parameters and helps in comparisons between related parameters.

2.2.4 Repeatability

Repeatability of experiments is a very important requirement of a simulator. The same experiment with the same parameters should produce similar results each time the simulation is executed. Otherwise the simulation becomes just a random sequence of events rather than a controlled experiment.

It is also helpful to be able to save an experiment (the set of input parameters) as a file and also be able to save the results of an experiment as a file.

2.2.5 Ease of extension

As already mentioned simulating something like the Internet is a complex task and it is

unlikely a 100% realistic simulation framework and a set of input parameters can be achieved in a few attempts. Therefore the simulator is expected to evolve continuously rather than a program that is written once and for all and then used continuously. Therefore the simulator architecture should support extensions with minimal effort with suitable frameworks.

2.3 Simulation Output / What is being Measured

Following are the statistical measures produced as output of the simulation in the initial version of the simulator.

- Response time of the simulated application
  - Overall average, minimum and maximum response time of all user requests simulated
  - The response time broken down by user groups, located within geographical regions
  - The response time further broken down by the time showing the pattern of change over the duration of a day
- The usage patterns of the application
  - How many users use the application at what time from different regions of the world, and the overall effect of that usage on the data centers hosting the application
- The time taken by data centers to service a user request
  - The overall request processing time for the entire simulation
  - The average, minimum and maximum request processing time by each data center
  - The response time variation pattern during the day as the load changes
- The cost of operation

2.4 Technologies Used

- Java – The simulator is developed 100% on Java platform, using Java SE 1.6.
- Java Swing – The GUI component is built using Swing components.
- CloudSim – CloudSim features for modelling data centers is used in CloudAnalyst.
- SimJava – Sim Java is the underlying simulation framework of CloudSim and some features of SimJava are used directly in CloudAnalyst.

3 CloudAnalyst Design

The CloudAnalyst is built on top of CloudSim tool kit, by extending CloudSim functionality with the introduction of concepts that model Internet and Internet Application behaviours.

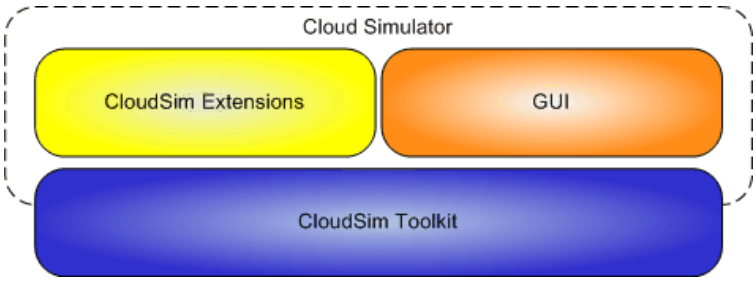


Figure 1. CloudAnalyst built on top of CloudSim toolkit

3.1 Use of CloudSim Toolkit

3.1.1 Functionality Leveraged as Is

CloudSim toolkit covers most of the activities taking place within a Data Center in detail. This includes:

- Simulating Data Center hardware definition in terms of physical machines composed of processors, storage devices, memory and internal bandwidth
- Simulating virtual machine specification, creation and destruction
- The management of virtual machines, allocation of physical hardware resources for the operation of virtual machines based on different policies (e.g. time-shared and space-shared)
- Simulating the execution of user programs or requests (Cloudlet/Gridlet) on the virtual machines

These features are directly used in the CloudAnalyst.

3.1.2 New Extensions Introduced

In addition to the Data Center operations provided by the CloudSim toolkit, following additional functionality is required for the CloudAnalyst and hence had to be built on top of CloudSim.

- **Application users** – Autonomous entities are required to act as traffic generators and their behaviour needs to be configurable.
- **Internet** – The data transmissions across the Internet needs to be realistically modelled with network delays and bandwidth restrictions.
- **Simulation defined by time period** – CloudSim as a toolkit is designed to process a pre-defined series of events (e.g. submission of n-number of cloudlets.) But for our purpose we need to convert the simulation to a time-frame limited execution where events are continuously generated by users until a pre-defined time period expires.
- **Service Brokers** – CloudSim already has the concept of DataCenterBrokers which performs a dual role in VM management in multiple data centers and routing traffic to appropriate data centers. But for CloudAnalyst these two main responsibilities were segregated and assigned to two different entities. The DataCenterController (described below) extends DataCenterBroker and is primarily responsible for the VM management within a single data center and load balancing of VM's etc within that single data center. The new entity CloudAppSeriveBroker was introduced to handle the responsibility of managing the routing of user requests between data centers based on different service brokerage policies.
- **GUI** - CloudAnalyst also introduces a comprehensive GUI which can be used to configure the simulation at a high level of detail. The GUI enables users to set up and

execute simulation experiments easily and in a repeatable manner which also benefits from highlighting the performance and accuracy of simulation logic thus automatically leading to overall improvement.

- **Ability to save simulations and results** – The CloudAnalyst also allows users to save a simulation configuration as a xml.file and also the exporting of results into PDFl format.

3.2 CloudAnalyst Domain Model and Main Components

Figure 2 summarises the main components and domain entities of the CloudAnalyst.

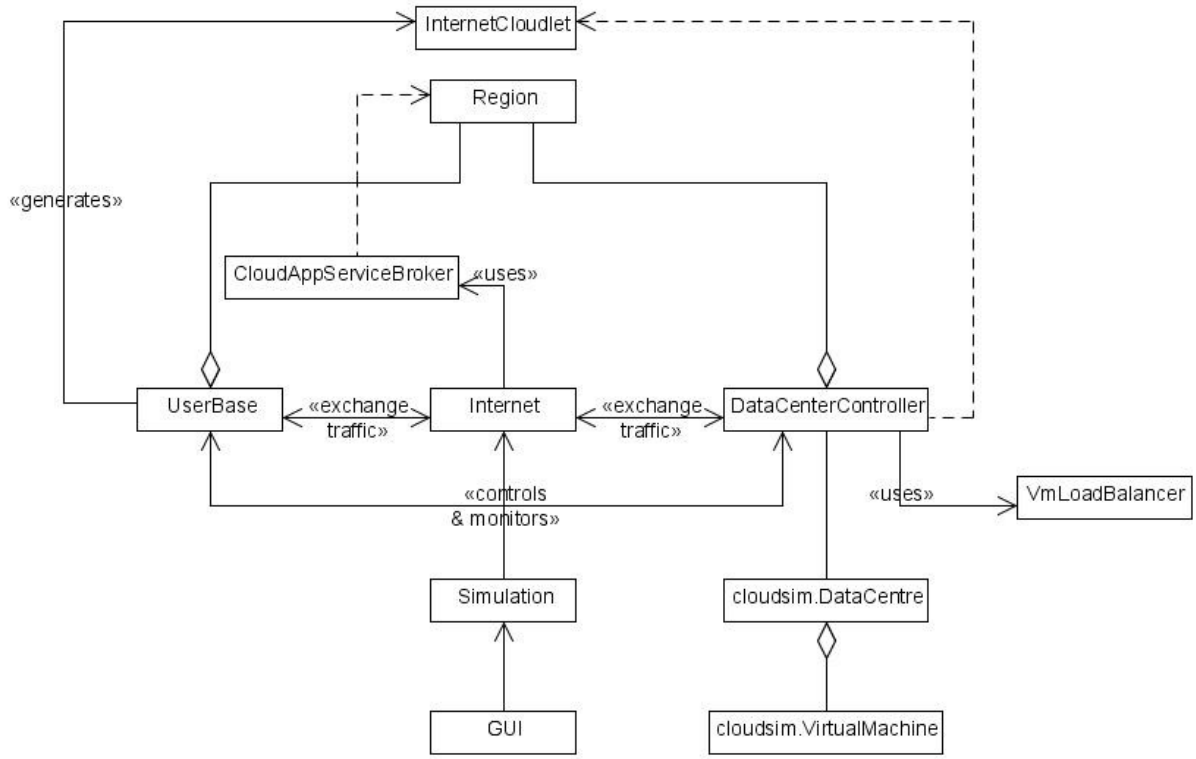


Figure 2. CloudAnalyst Domain

The following section describes these components and concepts in depth. The three main extensions introduced in CloudAnalyst to CloudSim toolkit are the User Base, Data Center Controller and the Internet. But before explaining those it is appropriate to introduce the concept of "region".

3.2.1 Region

In the CloudAnalyst the world is divided in to 6 'Regions' that coincide with the 6 main continents in the World. The other main entities such as User Bases and Data Centers belong to one of these regions. This geographical grouping is used to maintain a level of realistic simplicity for the large scaled simulation being attempted in the CloudAnalyst.

3.2.2 Internet

The CloudAnalyst Internet is an abstraction for the real world Internet, implementing only the features that are important to the simulation. It models the Internet traffic routing around the globe by introducing suitable transmission latency and data transfer delays. The transmission latency and the available bandwidth between the 6 regions are configurable.

3.2.3 Cloud Application Service Broker

The traffic routing between User Bases and Data Centers is controlled by a Service Broker that decides which Data Center should service the requests from each user base. Current version of CloudAnalyst implements three types of service brokers each implementing a different routing policy.

1. Service Proximity based routing.

   In this case the proximity is the quickest path to the data center from a user base based on network latency. The service broker will route user traffic to the closest data center in terms of transmission latency.
2. Performance Optimized routing

   Here the Service Broker actively monitors the performance of all data centers and directs traffic to the data center it estimates to give the best response time to the end user at the time it is queried.
3. Dynamically reconfiguring router

   This is an extension to Proximity based routing, where the routing logic is very similar, but the service broker is entrusted with the additional responsibility of scaling the application deployment based on the load it is facing. This is done by increasing or decreasing the number of VMs allocated in the data center, according to the current processing times as compared against best processing time ever achieved.

The implementation of Service Broker policies is discussed in detail in section 3.5.2.

3.2.4 User Base

A User Base models a group of users that is considered as a single unit in the simulation and its main responsibility is to generate traffic for the simulation.

A single User Base may represent thousands of users but is configured as a single unit and the traffic generated in simultaneous bursts representative of the size of the user base. The modeller may choose to use a User Base to represent a single user, but ideally a User Base should be used to represent a larger number of users for the efficiency of simulation.

3.2.5 InternetCloudlet

An InternetCloudlet is a grouping of user requests. The number of requests bundled into a single InternetCloudlet is configurable in CloudAnalyst. The InternetCloudlet carries information such as the size of a request execution command, size of input and output files, the originator and target application id used for routing by the Internet and the number of requests.

3.2.6 Data Center Controller

The Data Center Controller is probably the most important entity in the CloudAnalyst. A single Data Center Controller is mapped to a single cloudsim.DataCenter object and manages the data center management activities such as VM creation and destruction and does the routing of user requests received from User Bases via the Internet to the VMs. It can also be viewed as the façade used by CloudAnalyst to access the heart of CloudSim toolkit functionality.

3.2.7 VmLoadBalancer

The Data Center Controller uses a VmLoadBalancer to determine which VM should be assigned the next Cloudlet for processing. Currently there are three VmLoadBalancers implementing three load balancing policies which can be selected as required by the modeller.

1. Round-robin Load Balancer – uses a simple round-robin algorithm to allocate VMs
2. Active Monitoring Load Balancer – this version load balances the tasks among available VM's in a way to even out the number of active tasks on each VM at any given time.
3. Throttled Load Balancer – this ensures only a pre-defined number of Internet Cloudlets are allocated to a single VM at any given time. If more request groups are present than

the number of available VM's at a data center, some of the requests will have to be queued until the next VM becomes available.

3.2.8 GUI

The GUI is implemented as a set of screens that enable the user to:

1. Define the Simulation parameters
    a. Define in detail the characteristics of a Data Center including the detailed hardware specification of the server farm
    b. Define application deployment specifications such as how many virtual machines should be allocated in which data centers and the detailed specification of those virtual machines
    c. Define the user bases and their characteristics such as the number of users, the peak and off-peak hours of usage and the frequency of traffic generation
    d. Define Internet specific characteristics including network latency and available bandwidth
    e. Simulator performance related parameters such as grouping factors for user requests when messages are sent from UserBases and when messages are assigned to Virtual Machines in the Data Center
2. Save and load simulation configurations
3. Execute simulations with the option of cancelling a simulation once started
4. View and save the results of the simulation with graphical outputs where appropriate

3.3 Detailed Design

3.3.1 Class Design

The following class diagram describes the main set of classes of the CloudAnalyst. These classes are responsible for the modelling and execution of the simulations. The GUI is designed loosely coupled from this main simulation framework and hence shown as a package in the main diagram, but expanded and displayed in detail in the next diagram.

The classes UserBase, Internet and DataCenterController all extend the clousim.CloudSim class from the CloudSim toolkit and are active simulation entities that run as separate threads. The rest of the DataCenter functionality (VM management, task execution) is called by the DataCenterController via the cloudsim.DataCenter.

The main classes and their responsibilities are as follows:

- GuiMain – The main class of the GUI. Displays the GUI and acts as the front end controller for the application managing screen transitions and other UI activities.
- Simulation – Responsible for holding the simulation parameters and creating and executing the simulation.
- UserBase – Models a user base and generates traffic representing the users.
- DataCenterController – Controls the data center activities as explained above.
- Internet – Models the Internet and implements the traffic routing behaviour.
- InternetCharacteristics – Maintains the characteristics of the internet including the latencies and available bandwidths between regions, the current traffic levels and current performance level information for the data centers.
- CloudAppServiceBroker and its implementations – Models the service brokers
- VmLoadBalancer and its implementations – Models the load balancing logic used by data centers in allocating requests to VMs.
- UserBaseUIElement, DataCenterUIElement, MachineUIElement – holds information about user bases, data centers and machines for the UI until Simulation use them to create the respective simulation entities.

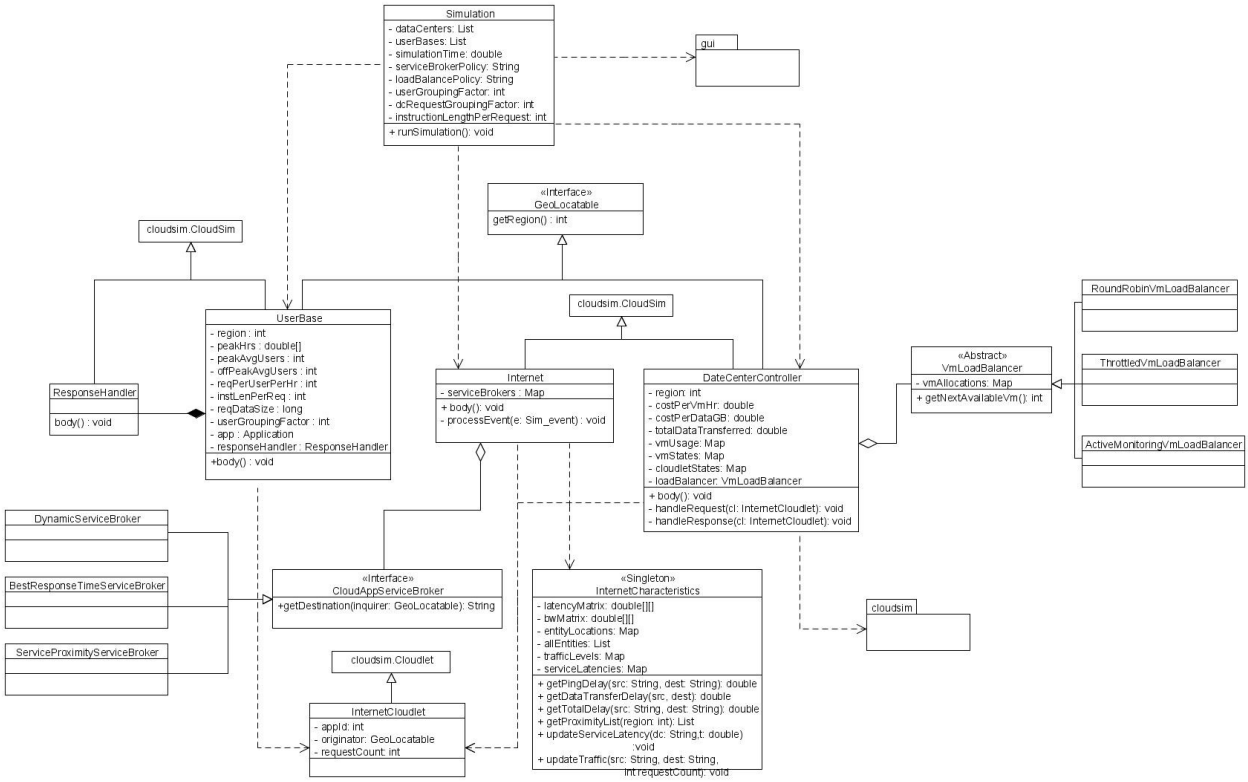


Figure 3. Class Diagram Showing the Main Classes of CloudAnalyst

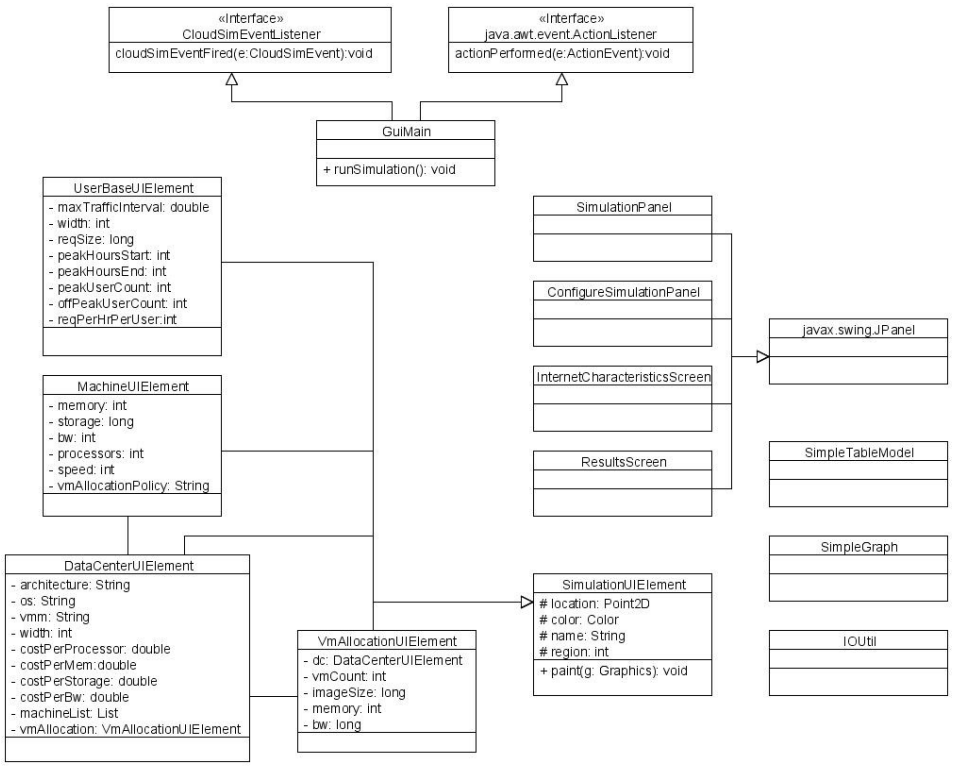


Figure 4. CloudAnalyst GUI Class Diagram

3.3.2 Sequence Diagrams

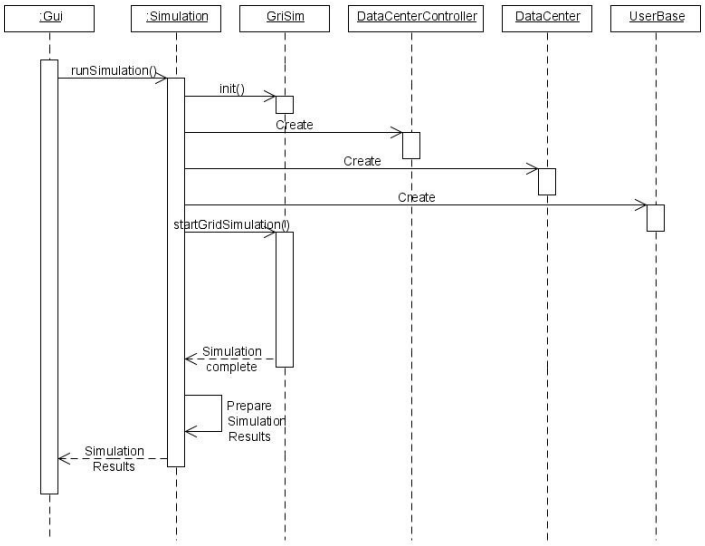


Figure 5. Sequence: Starting a Simulation

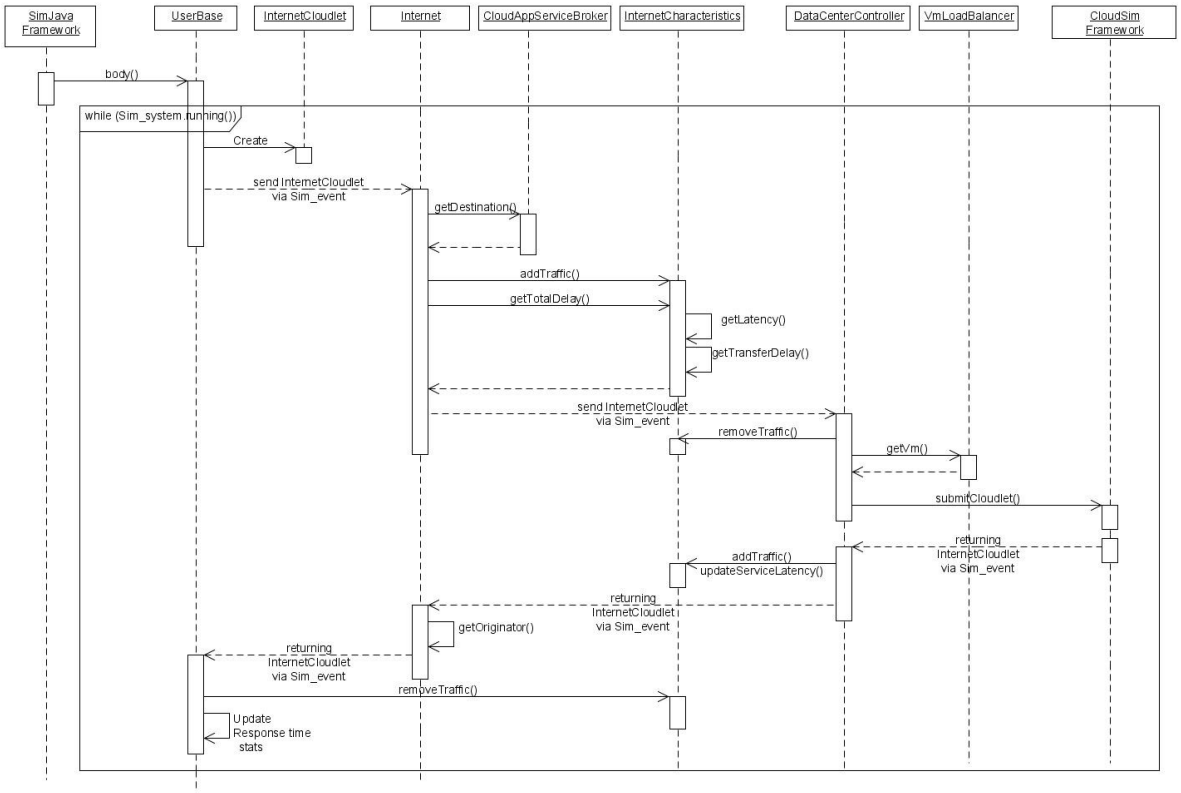


Figure 6. Sequence: Simulation Execution

3.4 Important Design Considerations

3.4.1 Routing of User Requests

In the real world scenario, a user of an Internet application is only interested in the application itself and is unaware how the application is deployed or the requests are being served. In other

words he would not care if the web page he is viewing in the browser of his personal computer is downloaded from a server in Europe or North America. The time taken for the download may be of interest to him or her, but generally the user will not be interested in the deployment specific details of the application. This behaviour is modelled in the CloudAnalyst using application id's and the routing logic employed by the Internet component as described below:

1. When a UserBase generates an InternetCloudlet, it will specify the application id for the application it is intended and also include the name of the UserBase itself as the originator for routing back the responses. Then the InternetCloudlet is sent to the Internet with a zero delay and tagged as a REQUEST.
2. The Internet upon receiving the message and seeing it tagged as a REQUEST, consults the Service Broker (a class implementing CloudAppServiceBroker interface) entity to decide which DataCenterController the InternetCloudlet should be sent to. The Service Broker maintains a list of DataCenterControllers, indexed by region and selects the best DataCenterController based on the Service Brokerage policy it is implementing.
    a. For Service Proximity Policy – The broker selects the data center that has the least network delay (without considering any processing time by the data center).
    b. For Best Response Time policy – The broker maintains a list of the latest request processing times by each data center. Then it projects the best response time by adding the appropriate network delay to the processing time and selects the data center that would give the least projected total response time.
    c. For Dynamic Configuration Policy – The routing policy is the same as for Service Proximity Policy.
    (See section 3.5.2 for details of the algorithms used.)
3. Then the Internet sends over the InternetCloudlet message to the selected DataCenter Controller adding a delay equivalent to the network delay given by the InternetCharacteristics.

For response messages the routing is much simpler.

1. The DataCenterController hands over the response InternetCloudlet to the Internet, tagged as a RESPONSE.
2. The Internet seeing it tagged as a RESPONSE, uses the 'originator' field of the cloudlet to identify the message destination and sends it over to the correct UserBase, adding the appropriate network delay to the message.

3.4.2 Calculating the Data Transmission Delay

The data transmission delay is calculated using the following formula:

$$T_{total} = T_{latency} + T_{transfer}$$

Where $T_{latency}$ is the network latency and $T_{transfer}$ is the time taken to transfer the size of data of a single request (D) from source location to destination. $T_{latency}$ is taken from the latency matrix (after applying Poisson distribution on it for distributing it) held in the InternetCharacteristics.

$$T_{transfer} = D / Bw_{peruser}$$

Where $Bw_{peruser} = Bw_{total} / N_r$

Bw_{total} is the total available bandwidth (held in the InternetCharacteristics) and N_r is the number of user requests currently in transmission. The InternetCharacteristics also keeps track of the number of user requests in-flight between two regions for the value of N_r.

3.4.3 Grouping Events to Simulate Large Scaled Operations

The nature of the simulation attempted with CloudAnalyst is extremely large with potentially thousands of users and millions of events. Therefore it is important to use event grouping to reduce the processing power and time required for the simulation but doing so should have minimal impact on how realistic the simulation is. Therefore the CloudAnalyst adopts the

following strategies for grouping of events:

1. The UserBase is the most apparent level of event grouping. The UserBase generates a single traffic event representing all the users of the UserBase. These traffic bursts are generated at intervals depending on the 'Requests Per User Per Hour' configuration parameter.
2. In the real world scenario each user request is transported through the Internet to the data center individually. Therefore each request is roughly equal in size when considered for routing etc. To emulate this situation, the traffic requests generated as above is organised in to a different level of grouping before being packaged in InternetCloudlets. This second level of grouping is based on the 'User Grouping Factor' configuration parameter. This re-grouping ensures the size of all (at least majority) InternetCloudlets passing through the Internet is consistent.
3. Once the requests are received at the DataCenterController it may again sub divided the requests in a single InternetCloudlet in to multiple sub InternetCloudlets based on the 'DC Request Grouping Factor'. In the most realistic scenario each request should be assigned to a VM individually and processed separately. But that again results in a large number of simulation events. This can be reduced by using a DC Request Grouping Factor higher than 1. But this causes the time taken for the processing of a single request to be lengthened proportionally grouping factor. This effect can be compensated by adjusting the 'Instruction Length per Request' to be smaller proportionally to the size of the DC grouping factor.

   The DC Request Grouping Factor may realistically be approximated as the number of threads a single VM can handle at the same time, or the size of the thread pool of the server application running on the VM.

If the user so chooses, the simulation can be run without grouping requests, the downside of doing so is the time taken for the simulation is quite high.

Another important design decision is in calculating the data center processing time of a request when the requests are grouped. When the DataCenterController receives a request InternetCloudlet it breaks up the request in to a number of sub InternetCloudlets as explained in step 3 above. Each of these sub cloudlets are then assigned to VM's by the load balancer and the original request is completed only when all the sub cloudlets are processed and returned to the controller. But this total duration is the time for processing all the requests in the InternetCloudlet. If the DataCenterController waits till this point to send back the response to the UserBase (for the original request) then the final response recorded by the UserBase is the total processing duration plus the transmission delay for a single request. Therefore the DataCenterController is designed to send back the response to the original request on the receipt of the first response sub cloudlet (instead of waiting for all sub cloudlets to return.) But all sub cloudlets are sent through the processing cycle to simulate the loading of the data center even after the response to the original request has been sent back. Experimenting with the code proved this to be the most fair strategy to handle this situation.

3.4.4 Distributing Parametric Measures using Poisson Distribution

In reality events such as arrival of data packets at a server do not occur at fixed intervals; nor does measures like ping round-trip time return the same result each time tested. In the real world such measures have been statistically modelled quite successfully using Poisson distribution. Therefore the following configuration parameters are used only as mean values for Poisson distributions which generate the actual values used in the simulation.

1. User base size (peak and offpeak)
2. Interval between requests generated by a UserBase
3. Network latency
4. Available bandwidth

3.4.5 Modelling Internet Bandwidth

Modelling bandwidth for a complex network like the Internet is an extremely difficult task. The CloudAnalyst "available bandwidth" configuration should not be viewed as the Internet bandwidth between regions. It is a hypothetical measure of the "**available**" bandwidth between regions. That is the bandwidth available between regions purely for the simulated application, disregarding all other traffic.

Then when calculating the network data transfer delay, the InternetCharacteristics takes into consideration the current level of application traffic and the available bandwidth in deciding how long it should take to complete the transmission of data.

3.5 Algorithms Used

3.5.1 VM Load Balancing Algorithms

As mentioned in section 3.2.7 The data centers use the VMLoadBalancer to load balance requests between the available virtual machines, and in the current version there are 3 variants. The first – round-robin is straight forward and is not explained further here. The algorithms used in the other two are explained below.

3.5.1.1 Throttled Load Balancer

Following is the algorithm used by the ThrottledVmLoadBalancer.

1. ThrottledVmLoadBalancer maintains an index table of VMs and the state of the VM (BUSY/AVAILABLE). At the start all VM's are available.
2. DataCenterController receives a new request.
3. DataCenterController queries the ThrottledVmLoadBalancer for the next allocation.
4. ThrottledVmLoadBalancer parses the allocation table from top until the first available VM is found or the table is parsed completely.
   If found:
   a. The ThrottledVmLoadBalancer returns the VM id to the DataCenterController
   b. The DataCenterController sends the request to the VM identified by that id.
   c. DataCenterController notifies the ThrottledVmLoadBalancer of the new allocation
   d. ThrottledVmLoadBalancer updates the allocation table accordingly
   If not found:
   e. The ThrottledVmLoadBalancer returns -1.
   f. The DataCenterController queues the request
5. When the VM finishes processing the request, and the DataCenerController receives the response cloudlet, it notifies the ThrottledVmLoadBalancer of the VM de-allocation.
6. The DataCenerController checks if there are any waiting requests in the queue. If there are, it continues from step 3.
7. Continue from step 2.

The throttling threshold maintained by this algorithm is 1. It can be modified easily to make the threshold a configurable value.

3.5.1.2 Active Monitoring Load Balancer

This load balancing policy attempts to maintain equal work loads on all the available VMs. The algorithm used is quite similar to the throttled case:

1. ActiveVmLoadBalancer maintains an index table of VMs and the number of requests currently allocated to the VM. At the start all VM's have 0 allocations.
2. When a request to allocate a new VM from the DataCenterController arrives, it parses the table and identifies the least loaded VM. If there are more than one, the first

identified is selected.
3. ActiveVmLoadBalancer returns the VM id to the DataCenterController
4. The DataCenterController sends the request to the VM identified by that id.
5. DataCenterController notifies the ActiveVmLoadBalancer of the new allocation
6. ActiveVmLoadBalancer updates the allocation table increasing the allocations count for that VM.
7. When the VM finishes processing the request, and the DataCenerController receives the response cloudlet, it notifies the ActiveVmLoadBalancer of the VM de-allocation.
8. The ActiveVmLoadBalancer updates the allocation table by decreasing the allocation count for the VM by one.
9. Continue from step 2.

3.5.2 Service Broker Algorithms

Currently there are 3 different implementations of Service Broker.

3.5.2.1 Service Proximity Based Routing

This is the simplest Service Broker implementation.

1. ServiceProximityServiceBroker maintains an index table of all Data Centers indexed by their region.
2. When the Internet receives a message from a user base it queries the ServiceProximityServiceBroker for the destination DataCenterController.
3. The ServiceProximityServiceBroker retrieves the region of the sender of the request and queries for the region proximity list for that region from the InternetCharacteristics. This list orders the remaining regions in the order of lowest network latency first when calculated from the given region.
4. The ServiceProximityServiceBroker picks the first data center located at the earliest/highest region in the proximity list. If more than one data center is located in a region, one is selected randomly.

3.5.2.2 Performance Optimized Routing

This policy is implemented by the BestResponseTimeServiceBroker, which extends the ServiceProximityServiceBroker.

1. BestResponseTimeServiceBroker maintains an index of all Data Centers available.
2. When the Internet receives a message from a user base it queries the BestResponseTimeServiceBroker for the destination DataCenterController.
3. The BestResponseTimeServiceBroker identifies the closest (in terms of latency) data center using the ServiceProximityServiceBroker algorithm.
4. Then the BestResponseTimeServiceBroker iterates through the list of all data centers and estimates the current response time at each data center by
    a. Querying the last recorded processing time from InternetCharacteristics.
    b. If this time is recorded before a predefined threshold, the processing time for that data center is reset to 0. This means the data center has been idle for a duration of at least the threshold time.
    c. The network delay from InternetCharacteristics is added to the value arrived at by above steps.
5. If the least estimated response time is for the closest data center, the BestResponseTimeServiceBroker selects the closest data center.
   Else, BestResponseTimeServiceBroker picks either the closest data center or the data center with the least response time with a 50:50 chance (i.e. load balanced 50:50).

All the algorithms mentioned so far in this section are just the initial versions of the algorithms. These can be further improved with more testing.

3.5.2.3 Dynamic Service Broker

Currently the Dynamic Service Broker is not fully implemented. Please see section 5.6.

4 Using the CloudAnalyst

CloudAnalyst is equipped with a comprehensive GUI built in Java Swing. This section describes briefly the screens and how to use them to set up and execute a simulation.

4.1 Setting up a Simulation

To set up a simulation you need to carry out the following steps. (Please note the screens mentioned here are explained in detail in the next section.)

1. Define user bases – Using User Base entities define the users of the application, their geographic distribution, and other properties such as the number of users, the frequency of usage and the pattern of usage such as peak hours. This is done in the Main tab of the Configure Simulation screen.
2. Define data centers – Using the Data Centers tab of the Configuration screen define the data centers you wish to use in the simulation. Define all the hardware and accounting aspects of the data centers here.
3. Allocate Virtual Machines for the application in Data Centers – Once the data centers have been created, you need to allocate virtual machines in them for the simulated application using the Main tab of the Configurations screen. A data center defined in step 2 above does not get included in the simulation unless it is allocated in this step. You can allocate multiple types of virtual machines in the same data center during this step.
4. Review and adjust the advanced parameters in the Advanced tab of the Configuration Screen.
5. Review and adjust the network latency and bandwidth matrices on the Internet Characteristics screen.

4.2 Simulator Screens

4.2.1 Main Screen with Simulation Panel

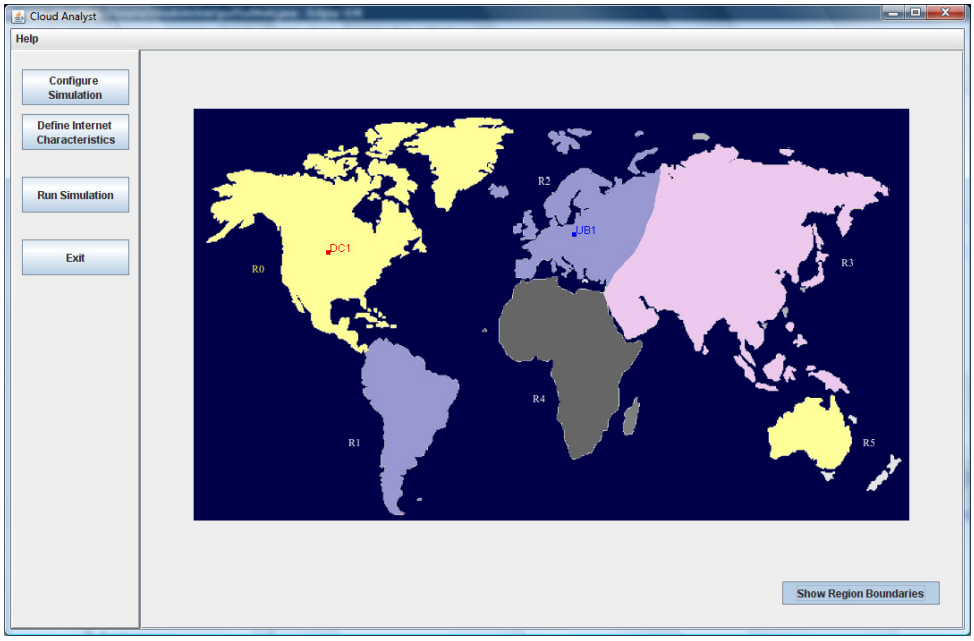


Figure 7. CloudAnalyst Main Screen

When CloudAnalyst is started the first screen displayed is the main screen. It has the simulation panel with a map of the world on the right and the main control panel on the left.

As mentioned the CloudAnalyst divides the world in to 6 regions that coincide roughly with the 6 main continents. Locations of all elements in the simulation are identified only by the region for simplicity (i.e. no x-y coordinates; all entities within are region are similar for geography specific parameters.)

Control Panel options are:

1. Configure Simulation – takes you to the Configure Simulation Screen
2. Define Internet Characteristics – takes you to the Internet Characteristics Screen
3. Run Simulation – Starts the simulation
4. Exit

At the start the simulator will be loaded with a simple default simulation.

4.2.2 Configure Simulation Screen

Configure Simulation screen has three tabs.

4.2.2.1 Main Tab

The configuration options on the main tab are:

1. Simulation time – the duration of the simulation which can be given in minutes, hours or days
2. User Bases Table – This is a table listing out all the user bases in the simulation. Each user base has following configurable fields, represented by a single row in the table.
    a. Name
    b. Region
    c. Requests per user per hour
    d. Data size per request
    e. Peak hours
    f. Average users during peak hours
    g. Average users during off-peak hours

    The Add and Remove buttons next to the table can be used to add or remove user bases from the configuration.

3. Application Deployment Configuration – This table lists how many virtual machines are allocated for the application in each data center from the Data Centers tab, along with the details of a virtual machine. The fields are:
    a. Data Center – This is a drop down listing the names of data centers created in the Data Center tab.
    b. Number of VMs – How many VMs to be allocated to the application from the selected data center
    c. Image Size – a single VM image size in bytes
    d. Memory – amount of memory available to a single VM
    e. BW – amount of bandwidth available to a single VM

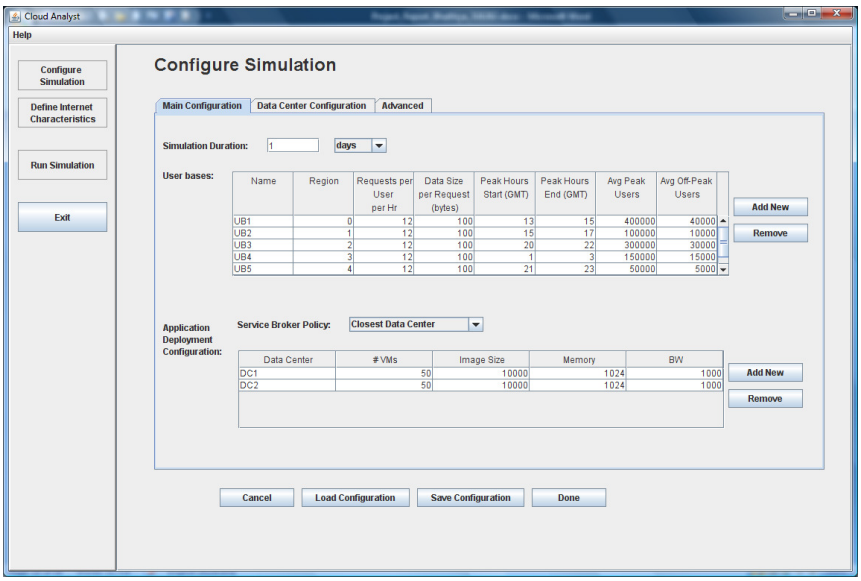


Figure 8. Configure Simulation Screen - Main Tab

4. Service Broker Policy – This drop down allows you to select the brokerage policy between data centers that decide which data center should receive traffic from which user base. The available policies are:
    a. Closest data center – The data center with the least network latency (disregarding network bandwidth) from a particular user base is sent all the requests from that user base.
    b. Optimize response time – This policy attempts to balance the load between data centers when one data center gets over loaded. Please see 3.5.2.2 for the algorithm used.

The "Save Configuration" button allows you to save the configuration created as a file. Simulation files are saved with a .sim extension. Similarly using the "Load Configuration" button you can load a previously saved simulation configuration.

4.2.2.2 Data Center Tab

The data center tab allows you to define the configuration of a data center (see Figure 8 below). The table at the top lists the data centers and using the Add/Remove buttons you can add or remove data centers to the configuration. The parameter fields are:

1. Name
2. Region
3. Architecture – Architecture of the servers used in the data center. e.g. X86
4. Operating System – e.g. Linux
5. Virtual Machine Monitor (VMM)
6. Cost per VM Hour
7. Cost per 1Mb Memory Hour
8. Storage cost per Gb
9. Data Transfer cost per Gb (both in and out)
10. Number of servers

When you select a data center from this table a second table will appear below it with the details of the server machines in the data center. The parameters for each machine can be given according to the available fields.

1. Machine Id
2. Memory

3. Storage
4. Available network bandwidth
5. Number of processors
6. Processor speed (MIPS)
7. VM allocation policy (time shared/space shared)

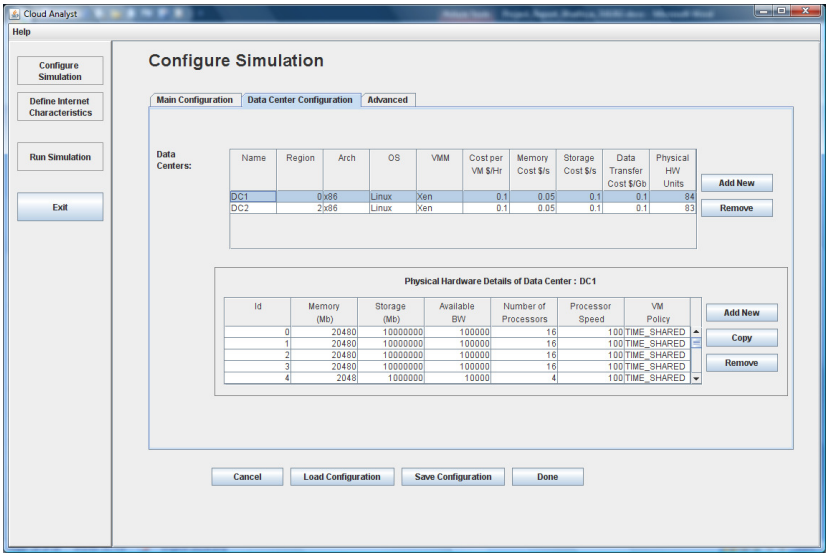


Figure 9. Configure Simulation Screen - Data Center Tab

4.2.2.3 Advanced Tab

The advanced tab contains some important parameters that apply to the entire simulation.

1. User Grouping Factor (in User Bases) – This parameter tells the simulator how many users should be treated as a single bundle for traffic generation. The number given here will be used as the number of requests represented by a single InternetCloudlet.

   In the ideal scenario this parameter should be 1, with each individual user represented independently. But that will increase the simulation time unrealistically.

2. Request Grouping Factor (in Data Centers) – This parameter tells the simulator how many requests should be treated as a single unit for processing. i.e. this many requests are bundled together and assigned to a single VM as a unit.

   Again, ideally this should be equal to 1. But it could also be viewed as the number of simultaneous threads a single application instance (VM) can handle.

3. Executable instruction length (in bytes) – This is the main parameter that affects the execution length of a request. This is the same GridletLength parameter used in GridSim.

4. Load balancing policy – the load balancing policy used by all data centers in allocating requests to virtual machines. Available policies are:
   a. Round-robin
   b. Equally Spread Current Execution Load – The load balancer keeps track of how many Cloudlets are currently being processed by each VM and tries to even out the active load.
   c. Throttled – The load balancer throttles the number of requests assigned to a single VM. See section 3.5.1.1 for the throttling algorithm.

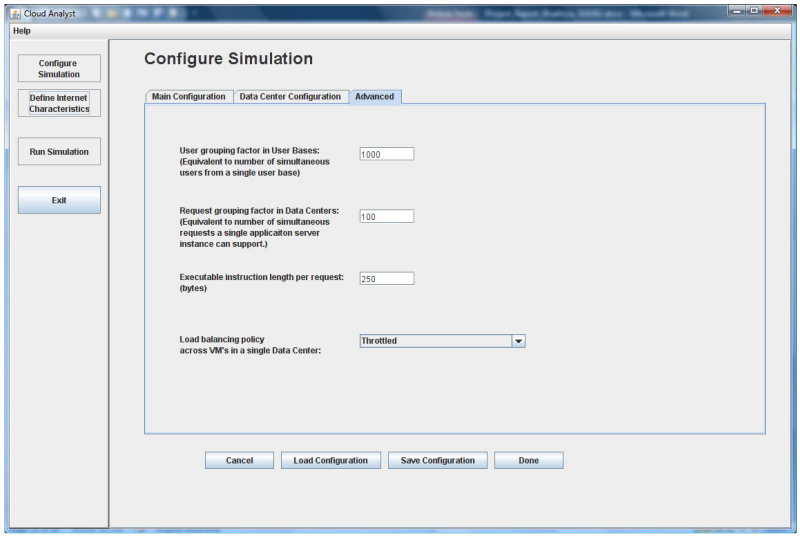


Figure 10. Configure Simulation Screen - Advanced Tab

4.2.3 Internet Characteristics Screen

The Internet Characteristics screen can be used to set the Internet latency and bandwidth parameters. It presents two matrices for these two categories.

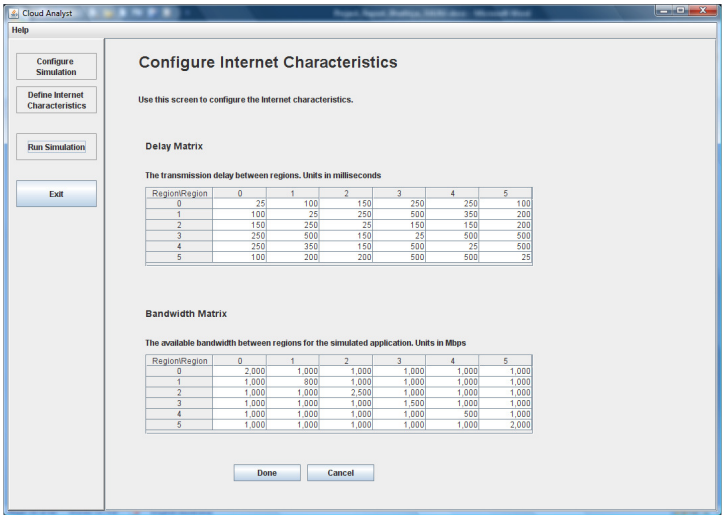


Figure 11. Internet Characteristics Screen

4.2.4 Running a Simulation

Once the above screens have been used to successfully create a simulation configuration, the user has to go back to the main screen and execute the simulation by selecting the "Run Simulation" from the control panel. This will start the simulation and the progress bar at the top of the simulation panel shows the percentage completion of the simulation. The simulation screen will display a simple animation showing which user bases are sending messages to which data centers.

A simulation can be cancelled before the completion of the run, using the cancel button at the bottom right hand corner. It may take a while after clicking the cancel button for the simulation to halt as it will continue to gather the simulation data of the requests that had been generated before cancelling but had not been completed.

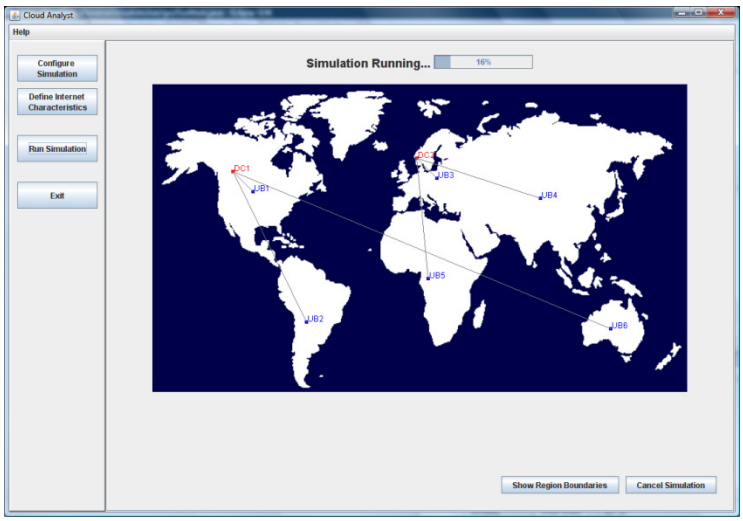


Figure 12. Simulation Panel During a Simulation

4.2.5 Results Screen

Once the simulation is completed the main response times will be displayed on the simulation panel next to each user base. The detailed results can be viewed by clicking the "View Detailed Results" button that appears at the right hand bottom corner of the screen after the simulation has completed.

The results screen will list out the data collected from the simulation. This includes:

1. Overall response time summary (for all the user bases)
2. Response time by user base in tabular format
3. Response time by user base in graphical format broken down into the 24 hours of the day
4. Request servicing time by each data center in tabular format
5. Request servicing time by data center in graphical format broken down into 24 hours of the day
6. Data center loading (number of requests serviced) in graphical format broken down in to 24 hours of the day
7. Cost details

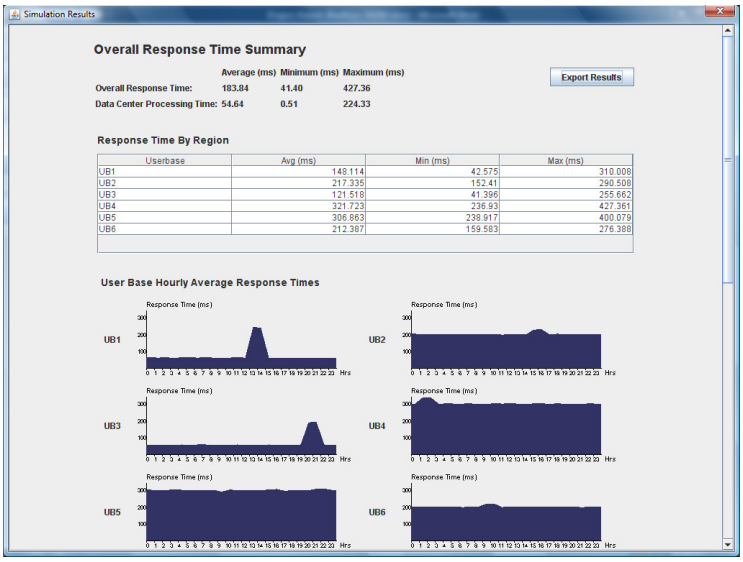


Figure 13. Results Screen

5 Simulating a Large Scaled Internet Application Running on the Cloud

A typical large scaled application type on the Internet today that could benefit from Cloud is social networking applications. E.g. Facebook, one of the most popular social networking sites has over 200 million registered users world wide. On 18/06/2009 the approximate distribution of the Facebook user base across the globe is given in the table below. [5]

Region	**CloudAnalyst Region Id**	**Users**
North America	0	80 million
South America	1	20 million
Europe	2	60 million
Asia	3	27 million
Africa	4	5 million
Ocenia	5	8 million

For our simulation let us assume a similar system but at $1/10^{th}$ of the scale of Facebook.

5.1 Simulation Configuration

We define 6 user bases representing the above 6 regions with the following parameters.

User base	Region	Time Zone	Peak Hours (Local time)	Peak Hours (GMT)	Simultaneous Online Users During Peak Hrs	Simultaneous Online Users During Off-peak Hrs
UB1	0	GMT – 6.00	7.00–9.00 pm	13:00-15:00	400,000	40,000
UB2	1	GMT – 4.00	7.00–9.00 pm	15:00-17:00	100,000	10,000
UB3	2	GMT + 1.00	7.00–9.00 pm	20:00-22:00	300,000	30,000
UB4	3	GMT + 6.00	7.00–9.00 pm	01:00-03:00	150,000	15,000
UB5	4	GMT + 2.00	7.00–9.00 pm	21:00-23:00	50,000	5,000
UB6	5	GMT + 10.00	7.00–9.00 pm	09:00-11:00	80,000	8,000

For simplicity each user base is contained within a single time zone and let us assume that most users use the application in the evenings after work for about 2 hours. Let us also assume that 5% of the registered users will be online during the peak time simultaneously and only one tenth of that number during the off-peak hours. Let us also assume that each user makes a new request every 5 minutes when online.

In terms of the cost of hosting, let us assume a pricing plan which closely follows the actual pricing plan of Amazon EC2[6], the most popular Cloud Service provider at present. The assumed plan is:

Cost per VM per hour (1024Mb, 100MIPS)	$0.10
Cost per 1Gb of data transfer (from/to Internet)	$0.10

Other parameters used are given in the table below. Please see section 4.2.2 for details on these parameters.

Parameter	**Value Used**
VM Image Size	10000
VM Memory	1024 Mb
VM Bandwidth	1000
Data Center – Architecture	X86
Data Center – OS	Linux
Data Center – VMM	Xen
Data Center – Number of Machines	20
Data Center – Memory per Machine	2048 Mb
Data Center – Storage per machine	100000 Mb
Data Center – Available BW per Machine	10000
Data Center – Number of processors per machine	4
Data Center – Processor speed	100 MIPS
Data Center – VM Policy	Time Shared
User Grouping Factor	1000
Request Grouping Factor	100
Executable Instruction Length	250

Latency Matrix values (in milliseconds):

Region/Region	**0**	**1**	**2**	**3**	**4**	**5**
0	25.0	100.0	150.0	250.0	250.0	100.0
1	100.0	25.0	250.0	500.0	350.0	200.0
2	150.0	250.0	25.0	150.0	150.0	200.0
3	250.0	500.0	150.0	25.0	500.0	500.0
4	250.0	350.0	150.0	500.0	25.0	500.0
5	100.0	200.0	200.0	500.0	500.0	25.0

Bandwidth Matrix values (in Mbps):

Region/Region	**0**	**1**	**2**	**3**	**4**	**5**
0	2000.0	1000.0	1000.0	1000.0	1000.0	1000.0
1	1000.0	800.0	1000.0	1000.0	1000.0	1000.0
2	1000.0	1000.0	2500.0	1000.0	1000.0	1000.0
3	1000.0	1000.0	1000.0	1500.0	1000.0	1000.0
4	1000.0	1000.0	1000.0	1000.0	500.0	1000.0
5	1000.0	1000.0	1000.0	1000.0	1000.0	2000.0

Now let us try to simulate this application with the CloudAnalyst and observe the behaviour.

5.2 Scenario 1 – Simple Web Application Hosted on a Single Data Center

Like with most real-world web application let us assume initially the application is deployed in a single location, in Region 0 (North America).

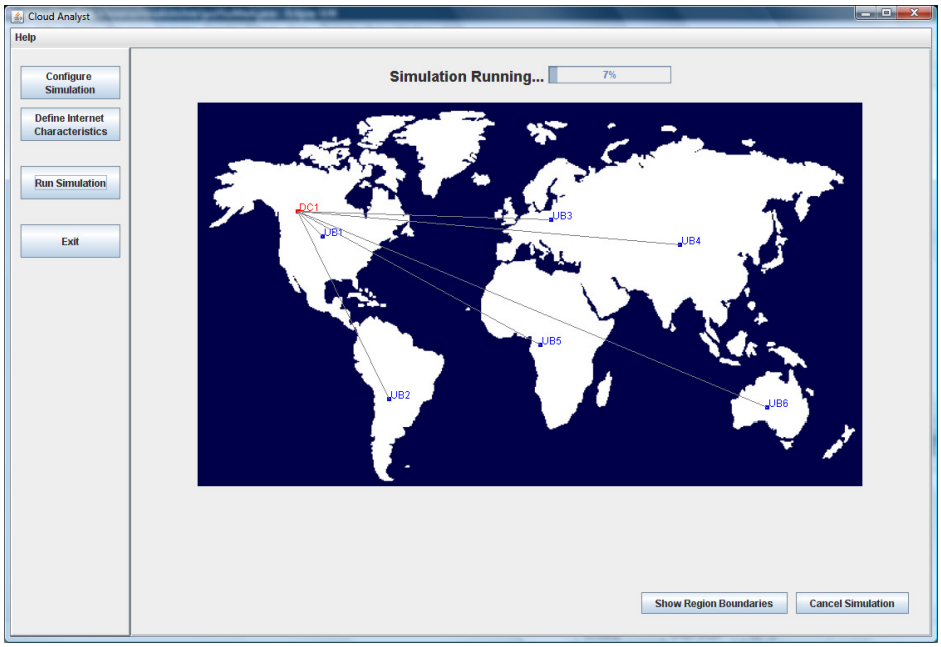


Figure 14. Scenario 1 - Single Data Center

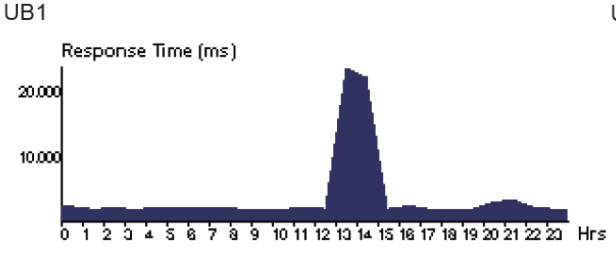
Assuming the application is deployed in 50 virtual machines (with 1024Mb of memory in each VM running on physical processors capable of speeds of 100MIPS), following is the simulation output.

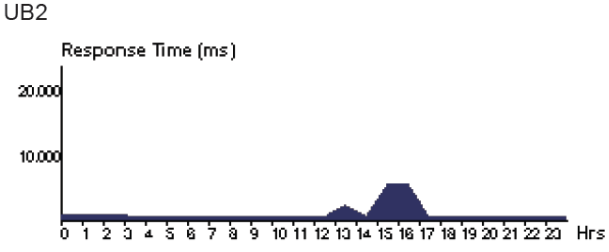
Overall Response Time

	Avg (ms)	**Min (ms)**	**Max (ms)**
Overall response time:	7965.50	223.00	39003.14
Data Center processing time:	7692.53	50.55	38938.86

Please note that these numbers are based on all the parameters mentioned in section 5.1 and therefore may not be realistic. But they serve as a sound basis for comparison between various scenarios.

The response times experienced by each user base are depicted graphically as follows:





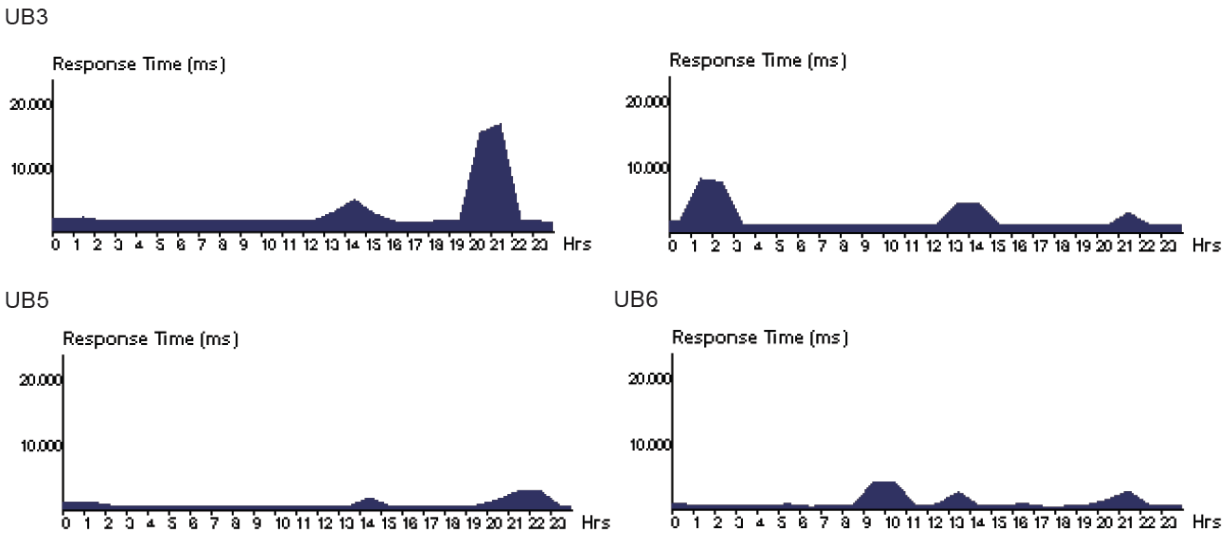


Figure 15. Scenario 1: Response Time across Regions with the Single Data Center

The spikes in response times can be seen clearly during the two hour peak period, and it can be observed how the peak loads of one user base could affect other user bases as well. For example UB1 has the peak between 13:00-15:00 GMT and all other user bases have small spikes during this same period. But impact has been less as the number of requests generated from those regions during this period is less.

During this 24 hour period the average time take by the data center to process a request and the number of requests processed is as follows:

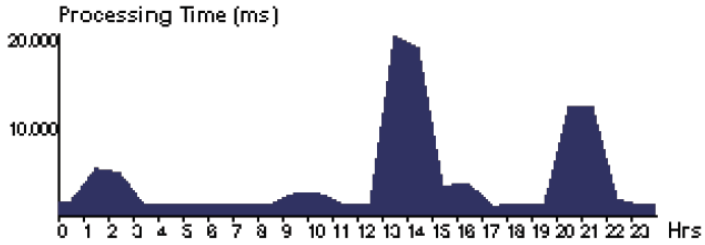


Figure 16. Scenario 1: Data Center Average Request Processing Time

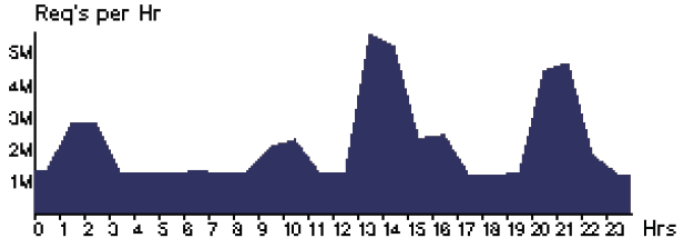


Figure 17. Scenario 1: Data Center Loading

As expected these two graphs reflect each other closely and the first one is very close to a superposition of all the region specific graphs above. During the high load of the UB1 peak from Region 0 (North America) the time for processing has dropped to almost 20 seconds and as can be seen from the region specific graph above the total response time is just over 20 seconds with the network delays added to the processing time. The overall processing time

statistics for the data center in numbers are as follows:

Average:	7692.53 ms
Minimum:	50.55 ms
Maximum:	38938.86 ms

So overall, users can expect a response time of about 7 seconds but during the day there are several periods of time when the actual response time can be expected to rise much higher.

Other important factor to the application owner would be how much it would cost for operating this application for a day. The CloudAnalyst calculates this amount to be:

Total Virtual Machine Cost:	$120.05
Total Data Transfer Cost:	$512.74
Grand Total:	$632.79

5.3 Scenario 2 – Web Application Hosted on Multiple Data Centers around the World

When applications grow in popularity on the Internet the most common approach to improve service quality is to deploy the application in several locations around the globe. So for the second scenario, while keeping the user bases the same we add one more data center, in region 2 (Europe). To keep the cost the same the 50 virtual machines are allocated 25 in each data center.

5.3.1 Case 1: Two Data Centers with 25 VMs in each

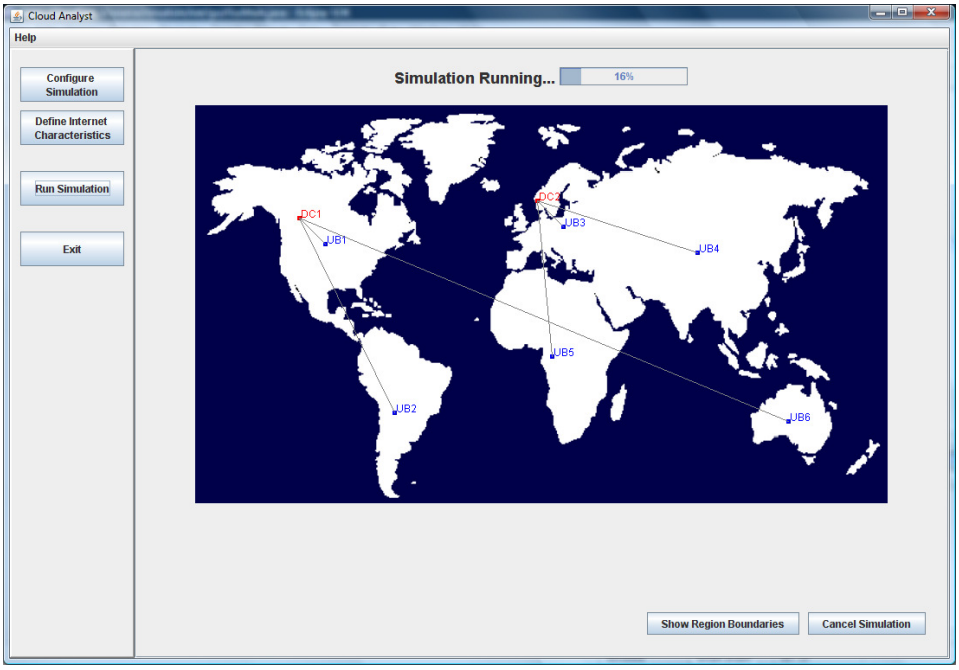


Figure 18. Scenario 2 - Application deployed in multiple Data Centers

Then the simulation results are as follows.

Overall response time and request processing times:

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	15068.35	275.17	86867.90
Data Center DC1 processing time:	16908.01	100.55	86810.59
Data Center DC2 processing time:	12480.52	125.36	45852.59

Region-wise Response Time Distribution:

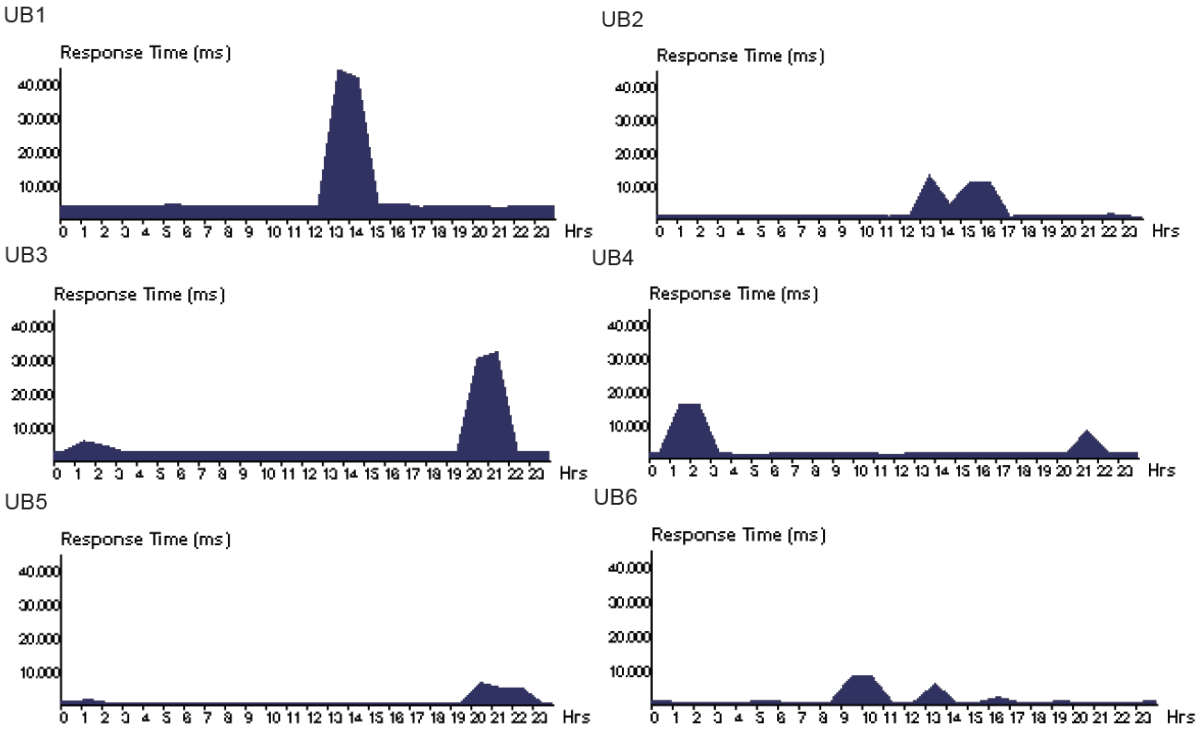


Figure 19. Scenario 2 - Region-wise Response Times

The result may not be quite what was expected by bringing the service closer to the users. The response time has roughly doubled. The pattern of response time distribution has not changed much with the main difference being the number of smaller peaks being reduced as not all the traffic is directed at the same data center this time.

The Data Center processing times and response times are as follows:

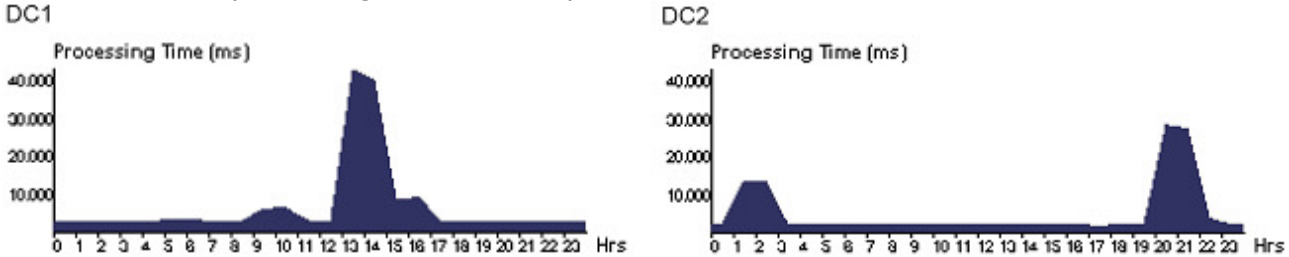


Figure 20. Scenario 2 - Data Center Processing Times

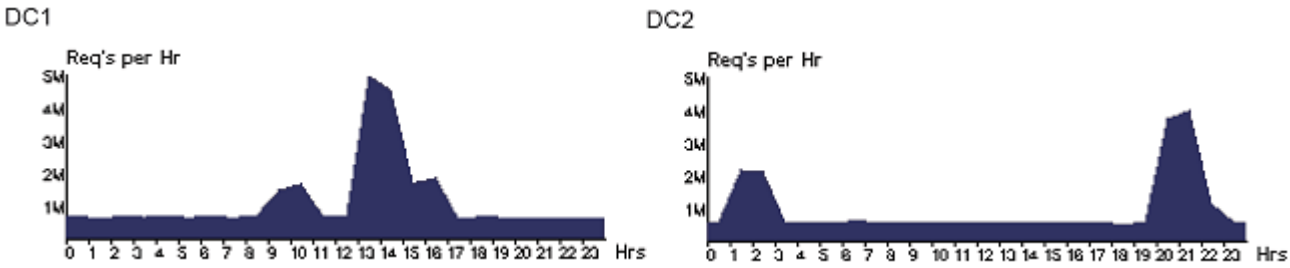


Figure 21. Scenario 2 - Data Center Loading

And the cost is:

Total Virtual Machine Cost:	$120.05
Total Data Transfer Cost:	$512.74
Grand Total:	$632.79

Which is exactly the same as before (as expected).

There are two reasons for poorer response times:

1. The data centers are overloaded during the peak times as evident from the fact that the processing time increase by about 100 times.
2. Reducing the number of virtual machines by half during each of these peak loads effectively lengthens the time taken to process by about two-folds in each data center.

Generally the first point above means the available processing capacity is not sufficient. Ideally the significant component of the total response time should be the network delays as it appears to have happened when recording the minimum response time of 275.17 ms when the minimum processing time has been 100.55ms. One main reason for this heavy loading is the low MIPS rating selected for the simulation parameters and the limited memory per VM. But since these extreme conditions high-light the patterns of activity, we continue to use these same parameters for the next few experiments as well.

In section 5.4 we produce some results taken from the simulator with more realistic server configurations.

5.3.2 Case 2: Two Data Centers with 50 VMs Each

By increasing the number of virtual machines in each data center to 50 following results was obtained.

Overall response time and processing times:

	Avg (ms)	**Min (ms)**	**Max (ms)**
Overall response time:	7337.57	223.00	30119.70
Data Center DC1 processing time:	8030.10	50.55	30054.92
Data Center DC2 processing time:	6174.12	125.01	26112.04

Region-wise Response Time Distribution:

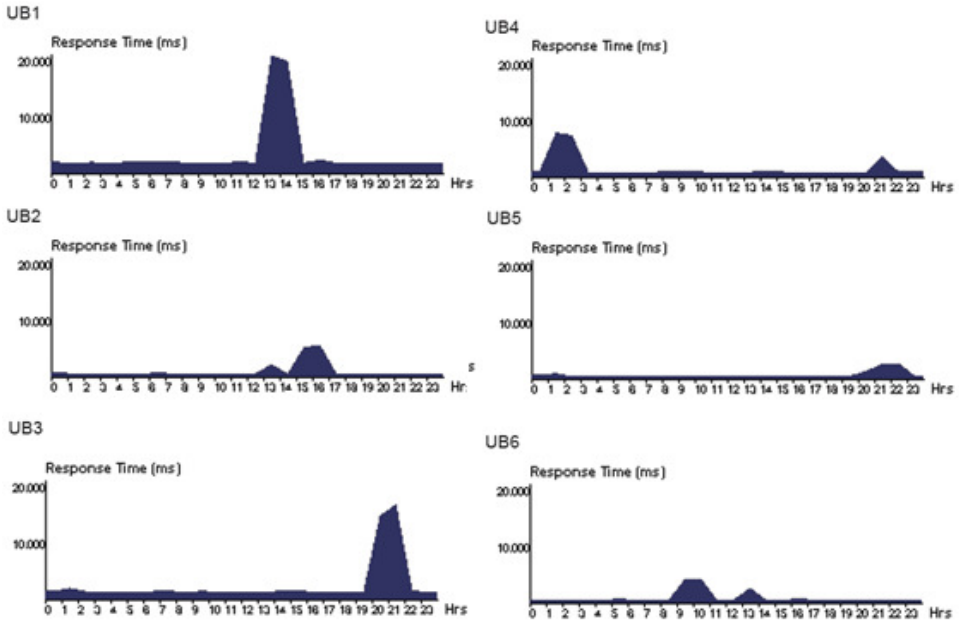


Figure 22. Scenario 2: Region-wise response time with 50 vms in each DC

Comparing these values with the Scenario 1 shows a reasonable improvement with the overall average response time improving from 7965.50 ms to 7337.57 ms.

The data center loading pattern for this scenario is as below:

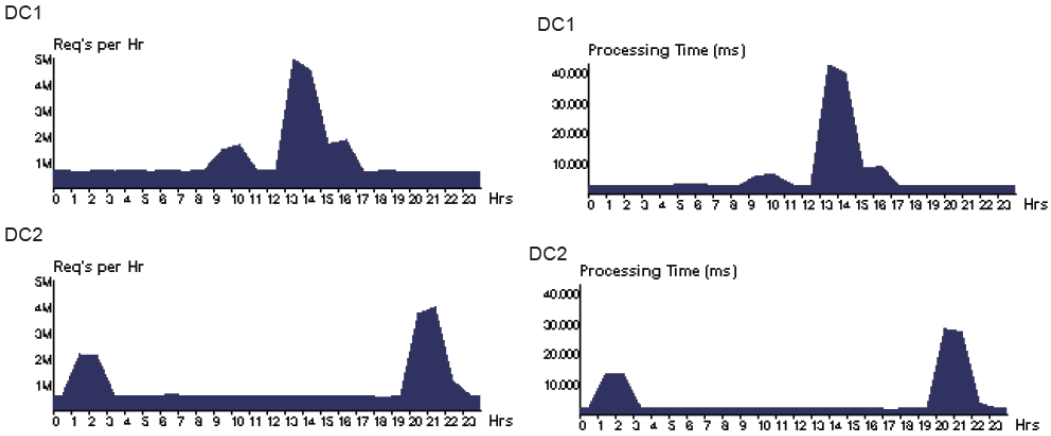


Figure 23. Scenario 2: Data Center Loading with 50VMs in each Data Center

The cost is:

Total Virtual Machine Cost:	$ 240.10
Total Data Transfer Cost:	$ 512.74
Grand Total:	$ 752.84

5.3.3 Case 3: Two Data Centers with 50 VMs Each and Sharing Load during Peak Hours

As it can be seen from the results in the previous section, clearly the heavy loads occur in the two data Centers at different periods of time. So what if some of the load at any point in time

is diverted from the most loaded data center to the lesser loaded data center? This scenario can be analysed by using the Optimized Response time service broker policy in CloudAnalyst and the results obtained are as follows.

Overall response time and processing times:

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	6716.65	228.24	39014.08
Data Center DC1 processing time:	8114.51	50.16	38952.68
Data Center DC2 processing time:	4615.20	50.31	17021.11

Region-wise response time distribution does not demonstrate any significant differences to the previous case, but the data center loading patterns show significant changes.

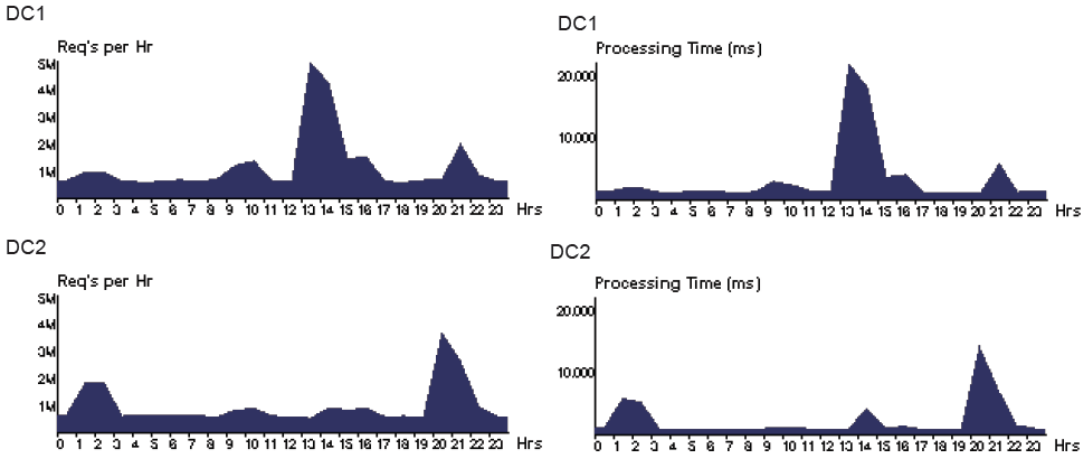


Figure 24. Data Center Loading when Sharing Peak Loads

As shown in the graphs, some amount of the peak-load has been transferred to the lesser loaded data center. The proportion of improvement gained by such load sharing is very much dependent on the algorithm used for load sharing. Please see section 3.5.2 for the algorithm used in Optimize Response Time service broker.

5.3.4 Case 4: Apply throttling to the processing of requests

Another common technique used in web application development is throttling. This scenario can be modelled using the load balancing policy as Throttled in the simulator. Then the results are as follows.

Overall response time and processing times:

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	3497.44	221.86	24049.39
Data Center DC1 processing time:	4003.38	50.19	23964.10
Data Center DC2 processing time:	2536.57	50.19	17085.07

This is almost doubling the performance of the previous case. But again, the benefit gained depends largely on the throttling algorithm. The throttling algorithm used in the simulator is explained in 3.5.1.1.

5.3.5 Case 5: Web Application Hosted on 3 Data Centers with 50VMs each

Let us now add a third data center to region 3 (Asia) with further 50 VMs allocated and observe the simulation. We use Optimise Response Time service broker policy with throttling enabled VM load balancing policy, as that produced the best results for the case of 2 data centers.

Now the overall response time and processing times:

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	3184.58	167.69	24041.95
Data Center DC1 processing time:	3909.24	50.26	23965.65
Data Center DC2 processing time:	2605.66	50.33	16979.78
Data Center DC3 processing time:	1269.12	50.37	8421.84

There are no surprises in these results; the overall response time has improved. DC1 gives the worst average response time as it faces the highest load.

5.3.6 Case 6: Web Application Hosted on 3 Data Centers with 75, 50, 25 VMs in each

So adjusting the number of VMs to DC1 – 75, DC2 – 50 and DC3 – 25, gives the following result.

Overall response time and processing times:

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	2698.13	170.78	17045.82
Data Center DC1 processing time:	2575.01	50.02	15968.87
Data Center DC2 processing time:	2439.60	50.36	16979.78
Data Center DC3 processing time:	2626.62	50.04	16309.57

Now the response time is further improved, but the most important result is that the average processing time at each data center is quite similar. This is demonstrated more clearly by the data center processing time graphs.

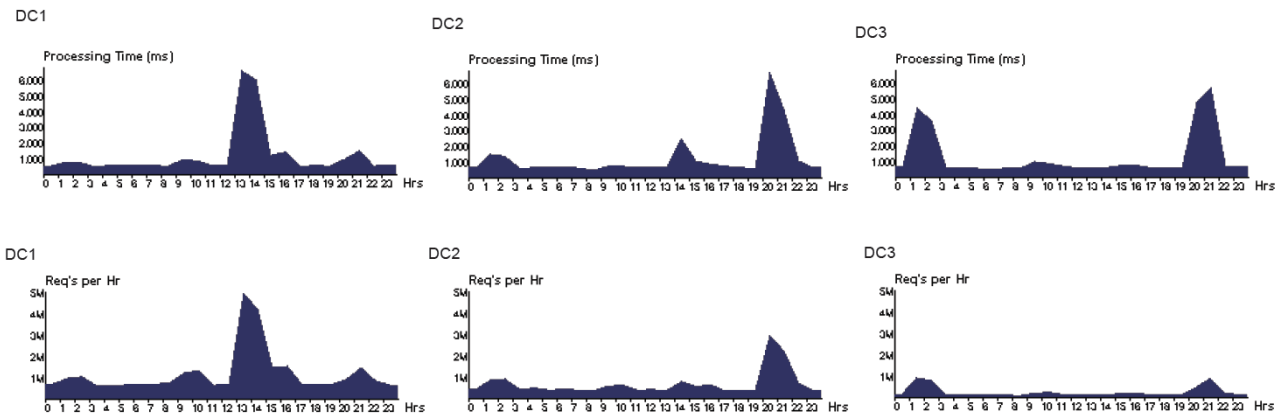


Figure 25. Data Center Loading Patterns for Case 6

From the graphs it can be seen that even though different data centers face peak traffic at different time periods, and the actual number of requests generated during those periods vary significantly, the 75-50-25 ratio of VMs have managed to keep the maximum processing time around the 6 seconds mark in all three data centers. As a result the overall response time drops below 3 seconds.

5.4 Simulation Results Summary

So after the series of experiments the results summary for the response times can be tabularised as follows:

Scenario	Scenario Description	Overall average response time (milliseconds)	Overall average time spent for processing a request by a data center (milliseconds)
1	1 data center with 50 VMs	7965.50	7692.53
2.1	2 data centers with 25 VMs each	15068.35	14872.42
2.2	2 data centers with 50 VMs each	7337.57	7176.79
2.3	2 data centers with 50 VMs each with peak load sharing	6716.65	6523.32
2.4	2 data centers with 50 VMs each with peak load sharing and throttling	3497.44	3322.38
2.5	3 data centers with 50 VMs each with peak load sharing and throttling	3184.58	3010.98
2.6	3 data centers with 75,50,25 VMs, with peak load sharing and throttling	2698.13	2537.79

Graphically it can be presented as below:

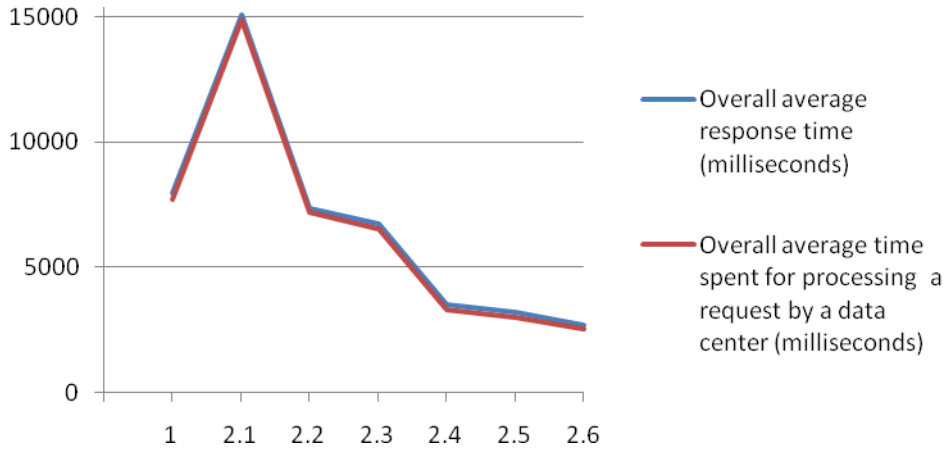


Figure 26. Summary of Simulation Results

As mentioned at the end of section 5.3.1 these results may not be realistic since we are trying to run a very large web application on a limited number of VMs running on fairly slow processors at 100 MIPS and only 1Gb of memory in each VM instance. (For comparison an

Intel Pentium III, 500MHz processor ran at 1,354 MIPS [7]). This limited configuration was used as it high-lighted the variations in usage patterns clearly.

Therefore we obtained the following results for the same experiments with more powerful hardware configurations of:

Parameter	**Value Used**
Data Center – Processor speed	10000 MIPS
VM Image Size	100000
VM Memory	10240 Mb
VM Bandwidth	10000

And the results obtained are:

	Scenario	**Overall average response time (milliseconds)**	**Overall average time spent for processing a request by a data center (milliseconds)**
1	1 data center with 50 VMs	284.98	46.79
2.1	2 data centers with 25 VMs each	249.20	119.97
2.2	2 data centers with 50 VMs each	183.85	54.65
2.3	2 data centers with 50 VMs each with peak load sharing	184.92	54.60
2.4	2 data centers with 50 VMs each with peak load sharing and throttling	157.56	28.45
2.5	3 data centers with 50 VMs each with peak load sharing and throttling	124.12	29.12
2.6	3 data centers with 75,50,25 VMs, with peak load sharing and throttling	121.07	23.96

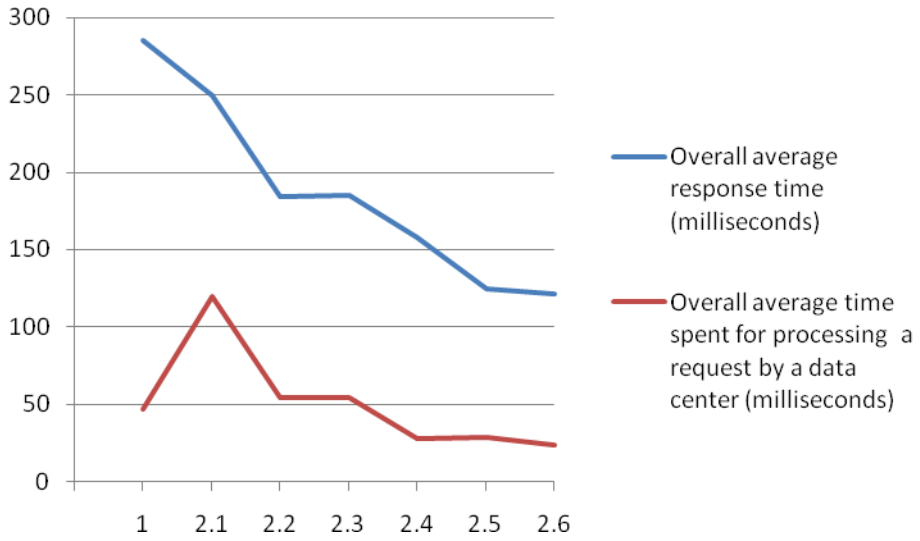


Figure 27. Summary of Simulation Results with More Powerful Hardware

These results look more realistic. E.g. when 50 VMs are divided in to bundles of 25 in to two data centers the processing time still goes up, but the overall response time goes down as the proximity of the service improves the network delays.

5.5 Main Observations from the Results

- Bringing the service closer to the users improves the quality of service (response time in this case)
- Service quality can be further improved by the application of load balancing at the application level across data centers (by different service brokerage policies) and also at virtual machine level within data centers. But the levels of improvement achieved depend largely on the load balancing algorithms employed.
- But for such improvements to be effective, sufficient capacity is required in the data centers to meet the peak demand.
- On the other hand if the peak capacity is allocated throughout, there will be a significant proportion of the time where that capacity is not fully utilized. This is not economical.

So overall, these results seem to suggest another approach that could optimize the performance while keeping the cost down.

5.6 Scenario 3 – Dynamic Re-Configuration of the Web Application Deployment Based on Usage

Based on the observations so far, the optimal solution looks to be a setup where the resources are dynamically allocated depending on the current work load. E.g. the highest load from region 0 (N. America) occurs from 13:00-15:00 GMT and during this time the data center servicing these requests (usually the data center located in region 0 itself) should have a higher number of VMs allocated. But once the peak has passed, the number of VM's could be dynamically reduced and the number in a different data center facing higher load can be increased.

This was attempted with a 'Dynamic Service Broker' in the simulator, but due to some unexpected behaviour of the CloudSim code the results obtained were not useful. Therefore we present only the concept in this section.

5.6.1 Proposed Algorithm for the Dynamic Service Broker

The DynamicServiceBroker should extend either ServiceProximityServiceBroker or the BestResponseTimeServiceBroker.

1. DynamicServiceBroker maintains a list of all data centers and another list with the best response time recorded so far for each data center.
2. When the Internet receives a message from a user base it queries the DynamicServiceBroker for the destination DataCenterController.
3. The DynamicServiceBroker uses the ServiceProximityServiceBroker/ BestResponseTimeServiceBroker algorithm to identify the destination.
4. The DynamicServiceBroker updates the best response time records if the current response time is better than previous.

In other words, the routing algorithm is the same as one of the other policies. But in addition to the above, service broker should run a separate thread to monitor the current response times of all the data centers.

1. If the current response time is increasing and is greater than the best response time for the data center plus some pre-defined threshold, the DynamicServiceBroker notifies the DataCenterController to increase the VM count by creating more VMs.
2. If the current response time is decreasing steadily for a pre-defined threshold of time the DynamicServiceBroker notifies the DataCenterController to reduce the VM count by releasing VMs.

6 Conclusion

With the advancement of Cloud technologies rapidly, there is a new need for tools to study and analyse the benefits of the technology and how best to apply the technology to large-scaled applications. A typical type of Internet application that could benefit from the flexibility of Cloud type services is social networking. Currently there are several simulation frameworks that can be extended and used to model this type of a problem, but there is no user friendly tool that can be used concentrating on the modelling effort rather than on the programming technicalities when using such frameworks. CloudAnalyst is a new tool developed to address this need, based on top of mature simulation frameworks such as SimJava, GridSim and CloudSim.

In the first part of this report we described the design of the CloudAnalyst in detail and explained the various algorithms used in different scenarios. Then in the second part, by applying the simulator to model a typical social networking type application with high usage loads, we demonstrated how different aspects of the operation of such an application can be studied using the CloudAnalyst. We showed how the simulator can be used to effectively identify overall usage patterns and how such usage patterns affect the data centers hosting the application. We also showed how the simulator can be used to study how different deployment configurations would tackle these usage patterns. In addition, the simulation exercise leads to many new insights that can be used to improve the service quality of such real world applications. One main possibility is introducing dynamic configurability through a global Cloud Service Broker, increasing or decreasing the size of the application in different locations depending on the load.

The simulation of a large scaled application on the Internet is a complex task. Therefore the CloudAnalyst is not a comprehensive solution to all such simulation needs. Currently it is the first step of a tool and an approach to studying this type of applications by simulation, and the tool and the approach is expected to evolve over the time producing improved quality of analysis along the way. In the long term these types of simulation experiments have a rich potential in identifying and experimenting with mechanisms and algorithms for improving performance in Cloud applications.

7 Future Work

CloudAnalyst is only a first step in a process that is expected to grow over a period of time. Following are some of the immediate next steps that should be adopted.

1. Resolve issues with CloudSim toolkit in dynamic reconfiguration and incorporate that to the simulator to study the dynamic behaviour.
2. Explore the VM load balancing and service load balancing (service brokerage) algorithms.
3. Incorporate failure handling mechanisms in to the simulation
4. Improve the GUI for usability
5. Improve the simulation panel animation during simulation

8 References

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic, *Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility*, Future Generation Computer Systems, Volume 25, Number 6, Pages: 599-616, ISSN: 0167-739X, Elsevier Science, Amsterdam, The Netherlands, June 2009.

[2] Rajkumar Buyya, Rajiv Ranjan and Rodrigo N. Calheiros, Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities, Proceedings of the 7th High Performance Computing and Simulation Conference (HPCS 2009, ISBN: 978-1-4244-4907-1, IEEE Press, New York, USA), Leipzig, Germany, June 21-24, 2009.

[3] R. Buyya, and M. Murshed, "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing," *Concurrency and Computation: Practice and Experience,* vol. 14, no. 13-15, pp. 1175-1220, 2002.

[4] U. o. E. Institute for Computing Systems Architecture. "simjava," 15-05-2009; http://www.dcs.ed.ac.uk/home/hase/simjava/.

[5] "Facebook," 16/6/2009; http://www.facebook.com.

[6] "Amazon Elastic Compute Cloud (Amazon EC2)," 19/06/2009; http://aws.amazon.com/ec2/.

[7] Wikipedia. "Instructions per second," 20/06/2009; http://en.wikipedia.org/wiki/Instructions_per_second.

Appendix A – Facebook Statistics

According to Facebook it has over 200 million users as of 18/06/09 and over 100million users log into Facebook daily. Following table summarises the registered user counts for some of the main countries across the globe (as of 18/06/09). [5]

Country	Registered Users
N.America	
USA	61,625,160
Canada	10,791,100
Mexico	2,937,920
South America	
Argentina	3,874,860
Brazil	853,680
Chile	3,971,240
Colombia	4,447,240
Venezuela	2,881,960
Europe	
Belgium	2,044,000
Denmark	1,633,520
France	8,701,140
Germany	2,687,720
Greece	1,391,000
Italy	8,428,100
Norway	1,532,960
Switzerland	1,250,840
Sweden	1,928,980
U.K.	16,183,520
Africa	
Kenya	309,400
Nigeria	518,500
South Africa	1,499,200
Egypt	1,413,800
Asia	
China	858,140
Hong Kong	1,801,460
India	2,693,080
Japan	524,760
Malaysia	1,708,540
Indonesia	4,936,900
Israel	1,146,300
Pakistan	688,000
Phillipines	2,107,700
Singapore	1,205,400
Sri Lanka	253,040
U.A.E	691,600

South Korea	261,860
Russia	388,200
Ocenia	
Australia	5,345,820
New Zealand	821,500