CHAPTER 5

COST AND PERFORMANCE ANALYSIS OF CMAAS
As discussed in chapter 1, the motive is not only to present a solution but also, give an idea of probable benefits and costs involved in having CMaaS. The benefits of CMaaS are already been discussed in chapter 3 section 4, where, the desired features of CMaaS were presented. The desired features are mostly sparsely present in current scenario monitoring. While talking of costs, it is very necessary to understand the way how cost calculations are done in cloud. Hence, a perpetual cost model is being discussed before cost analysis. Similarly, a performance model is presented before the performance analysis is given. The complete simulation model is being discussed prior to cost and performance models in section 5.1 onwards.

5.1 The Simulation Setup

5.1.1 About Cloudsim

Most of the simulation work done in computer science makes use of simulators like ns2 [http://www.isi.edu/nsnam/ns/] and ns3 [http://www.nsnam.org/] for network related simulations, Globus toolkit [https://www.globus.org/toolkit/] for grid systems. Other simulators like MatLab [www.mathworks.in/products/matlab/], Vissim [www.vissim.com/] etc. are also widely used for simulating empirical environments. As mentioned by Calheiros, R. N. et al. (2011) Cloudsim is being used widely by researches to simulate algorithms and structural aspects of cloud computing. This section discusses various aspects of Cloudsim which enabled the analysis of CMaaS architecture. The approach towards testing or simulation framework is as given in figure 5.1. The model was introduced with job parameters which were consisting of number of users posing request and duration of service etc. The model output certain statistics that were analyzed and inferences were drawn. The core model was made by using Cloudsim which was used to simulate the model for cost and performance analysis. The layered Cloudsim architecture reveals the major layers placing broker, cloudlets,
virtual machines, cloudlet execution etc. The architecture of Cloudsim is well presented in paper by Calheiros, R. N. et al. (2011). Fig 5.2 explains the layered Cloudsim architecture. Cloudsim supports modelling and simulation of virtualized datacenters. The Cloudsim is made over the GridSim layer where basic building blocks are like Network topology, message delay calculation, event handling, sensor, Cloud coordinator, Datacenter, VM provisioning, CPU, Memory, Storage are bandwidth allocations. The major interface of study in Cloudsim are cloudlet and virtual machines which are used to execute cloudlets under a provisioned VM and study the analytical aspects. Calheiros, R. N. et al. (2011) also discusses the aspects of modelling a cloud market, network behavior, federation of clouds, dynamic workload, power consumption at data center and dynamic entity creation. Many projects have come up using Cloudsim, like Cloudsim Ex, WorkflowSim, Dynamic Cloudsim etc as listed at [http://www.cloudbus.org/Cloudsim/].

Figure 5.1: Testing architecture.
5.1.2 Setting up the Cloudsim

For the requirement of modelling performance, the major steps in the clouds in environment were made. Cloudsim as mentioned at url:http://www.cloudbus.org/cloudsim/ was installed on eclipse [http://www.eclipse.org/]. Major modifications in various classes where made so as to adapt to the performance evaluation. The classes which were extended were:

- DataCentreBroker (DCBroker).
- Cloudlet (MyCloudlet).

Major classes (referred as helper classes) are made to complete the simulation, they are:

- UserRequest
- LongTerm
- SimParametres
The Cloudsim project being mainly initiated by R. Buyya, R.N Calheiros and N. Groyer. Cloudsim with latest version Cloudsim 3.0.3 was firstly launched in 2010 with version 2.1 at Melbourne university cloud labs (initially grid labs www.cloudbus.com/ intro.html). The major features Cloudsim are as listed in [cloudbus.org/Cloudsim]. The download link (http://code.google.com/p/Cloudsim/downloads/) provides with an open source free download capability for anyone. The procedure for installing Cloudsim is also very simple. Figure 5.3 to 5.8 gives the complete class diagram of the setup. The DCBroker class is majorly used to take cloudlets and submit them to particularly allocated VMs. The DCHost uses particular provisioner to a broker and in turn to the user. The DCHost also creates and destroys VMs as and when required. The DCBroker can at any point of time de-allocate the VMs which are allocated to a particular user. Overall, the DCHost is the class used to allocate VMs, set main configurations of the VMs like process elements, RAM size, memory etc. It also helps in scheduling and provisioning of other aspect like jobs and bandwidth. Overall DCHost is the class which sets the datacenter present in the Cloudsim core classes. Only few functionalities of the core classes are used and not all. To bring in the monitor server as in 4.1, the datacenter broker is also extended into DCMonitor. The prime role of DCMonitor is to recieve log cloudlets and process them. MyCloudlet classes are mainly used to create cloudlets and logs at cloud. Other helper classes like SimParameters and UserRequest are used to maintain simulation parameters as discussed later in this section) and maintaining user request parameters. The performance model and cost model uses two topologies namely shared VM monitoring and dedicated VM monitoring as discussed in section 5.3.1.
Figure 5.3: class diagram of Simulation setup (only for those classes which have been modified from Cloudsim).
Figure 5.4: The main class.
Figure 5.5: cloudlet class used in simulation.

Figure 5.6: the broker and monitor classes.

Figure 5.7: the generic simulation class
Figure 5.8: the class holding simulation parameters and methods.
5.1.3 Setting up Cloudsim and workload

The simulation was done on both topology 1 and topology 2 (as discussed in section 5.3.1 Performance model). The inputs in both topologies are as listed in table 5.1. The input length of user cloudlets, start times of each cloudlet varies uniformly (pl see Jerry banks (2005)) between 0-2000 for the former and 0 to 28800 (1 day) for each day. Also the number of users logged in per day varies from 0 to the number of users in the organization. The major reason for taking uniform distribution is that, we are mostly concerned to know the overall effects of the monitoring processes on the user request processing and in case of any other empirical distributions (pl see Jerry banks (2005)) the trends might show non-natural behavior, which might lead us to wrong interpretations. The right way will be to look at a normal monthly workload of an organization and study the results (see figure 5.12). However, we also took few workloads from sources so as to give a real touch and compare the stochastic results with the real workload results. The cases of simulation and results are presented in chapter 7.

Table 5.1a: Host configurations

<table>
<thead>
<tr>
<th>No of hosts</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual core</td>
<td>1000MIPS each</td>
</tr>
<tr>
<td>Linux x86, Xen</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>2GB</td>
</tr>
<tr>
<td>storage</td>
<td>1000000000</td>
</tr>
<tr>
<td>bw</td>
<td>10000</td>
</tr>
<tr>
<td>Average No of Cloudlets Per user</td>
<td>10</td>
</tr>
<tr>
<td>Days</td>
<td>30</td>
</tr>
<tr>
<td>Avg no of logs per user</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5.1b: Test configurations

<table>
<thead>
<tr>
<th>Topology</th>
<th>No of VMs at</th>
<th>Monitored?</th>
<th>No of</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 VMs</td>
<td>Y</td>
<td>1 VM</td>
</tr>
<tr>
<td>2</td>
<td>2 VMs</td>
<td>Y</td>
<td>Shared</td>
</tr>
<tr>
<td>NA</td>
<td>2 VMs</td>
<td>N</td>
<td>NA</td>
</tr>
</tbody>
</table>
The simulation was run for three cases as given in table 5.1b. There were two VMs allocated to the broker and one VM for monitor (if any) at datacenter. The case of no monitoring was also taken so as to compare the results to that of CMaaS for effects on user services. The basic configuration of host is also listed in table 5.1a.

Apart from the stochastic workload a set of logged workloads from various sources were also considered while performing simulations. The need of taking workloads from sources was primarily to confirm the findings on stochastic workloads. In most of the cases the brokers and the user organizations function and interact on networks so network also needed to be considered while simulating. BRITE[http://www.cs.bu.edu/brite/] network topologies were formed considering number of nodes from very small (for small organizations) to very large (for large organizations). The workloads and their sources are given in the table 7. 1 and the network topologies are given in the table 7. 2-7.6 (In large cases only first few and last few nodes and edges are tabulated in this document).

## 5.2 The Performance Analysis

*Sample Sequence diagrams.*

It is worthwhile to view certain sequence diagrams to understand how the simulation has been performed. The figures 5.9 to 5.11 (built using ObjectAID UML diagram tool under Eclipse) holds few sequence diagrams which are meant to have an idea of the flow of the simulation using Cloudsim. Firstly, a set of parameters as discussed previously in the section (like number of users, days of simulation, cost of data and processing etc.) as well as parameters that will be discussed in section 5.2 and 5.3 (like, leased or not, shared VM or dedicated VM etc.) are maintained. Using these parameters the user cloudlets as well as log cloudlets are made. The major Host configurations are set independent of cloudlets. The user cloudlets are
submitted to the broker and the log cloudlets are submitted to the monitor respectively. And finally the cost and performance readings are taken.

Figure 5.9: Sequence diagram- submitting cloudlets to Broker.
Figure 5.10: Sequence diagram- submitting log cloudlets to Monitor.

Figure 5.11a: Sequence diagram- creating log and user cloudlets.
SimParameters number of logs per user and average number of cloudlets per user. In most of cloud architectures, the allocation of VM on a service is made through the brokers as marked by Calheiros, R. N. et al (2011). The brokers in CMaaS will be architecturally similar to the Cloud Service Server (figure 4.2b). The brokers decide upon the scheduling and allocation of resources to the user requests based on specific priorities and policies. Upon introduction of CMaaS, it becomes obvious that a policy must be incorporated whether to provide MaaS from the specially allotted VM and other resources at the datacenter or to provide MaaS on the VMs that are already allotted to the brokers specifically for the particular organization. However, in both the cases the brokers will be the center point of various monitoring activities. The complete testing will be done on two topologies that will be implemented using Cloudsim.
5.2.1 Performance Model

Shared VM Allocation and Dedicated VM Allocation for Access Monitoring

Since the hybrid monitoring (as CMaaS is) will need further compute support at both organizational servers and datacenters it needs to be assessed whether the CMaaS doesn’t become an over burden. The part of burden at organization level can be ignored as it is in the organizations interest to have CMaaS. However, the accessing of extra cost and performance issues experienced by the organization because of the additional resources taken into account of CMaaS from datacenters has to be analyzed for conformance. Topology 1 (figure 5.13a) will be the one in which the monitor will run at datacenter (which is our proposed setup) and the broker will have its own set of VMs. The monitor will be running and will submit cloudlets to the separately reserved VM for a particular organization. The VMs for monitor can be internally managed or can be negotiated by the broker itself. The second topology (figure 5.13b) is the one where the monitoring is done at the broker’s part. Here the log

Figure 5.12: Trends of usage per day used in simulation.
cloudlets will be submitted to the VMs which are allotted to the broker and will be shared by all other user cloudlets along with the log cloudlets.

Figure 5.13a: Topology1: Reserved VM for monitoring.

Figure 5.13b: Topology2: Shared VM for monitoring.
5.3 Cost Analysis

5.3.1 Cost Model

The testing framework will begin in the similar fashion to that of a basic architecture diagram as shown in figure 5.1. The model will be introduced with test data in terms of a) cost parameters containing the basic pricing of various entities at the datacenter and b) the job parameters which will be consisting of number of users posing requests and duration of service etc. The model will output certain statistics that can be analyzed and inferences could be drawn. The core model will be made by using Cloudsim. Cloudsim was used to simulate the model for cost analysis. The layered Cloudsim architecture reveals the major layers placing Broker, Cloudlets, Virtual Machines, Cloudlet execution etc. Majorly, in cost analysis the topologies (figure 5.13) can be easily implemented and studied without bothering much about scheduling, SLA and QoS parameters. The complete testing will be done on two topologies that will be implemented using Cloudsim. Topology 1 (figure 5.13a) will be the one in which the monitor will run at datacenter (which is our proposed setup) and the broker will have its own set of VMs. The monitor will be running and will submit cloudlets to the separately reserved VM for a particular organization. The VMs for monitor can be internally managed or can be negotiated by the broker itself. The second topology (figure 6b) is the one where the monitoring is done at the broker’s part. Here the log cloudlets will be submitted to the VMs which are allotted to the broker and will be shared by all other user cloudlets along with the log cloudlets. Later on in the inferences it will be observed that this topology is not as cost effective as it appears to be on the prima facie.

Cost calculation

Basically, cloud pricing are made in two ways, pay-as-you-go as described by Ibrahim S. et al. (2011) and leased prices as described by Borja Sotomayor et al. (2009). The exact way of the cost calculation may not be same with every provider. Hence, we are bound to follow a fine grained calculation, considering almost every aspect of computing into our cost calculations. Although, we have taken utmost care of not sparing any criteria, it might be that we may have
missed one or two. Even then, it be rest assure that the left ones have almost negligible significance on the overall cost.

Pay-as-you-go:

Cost of $i^{th}$ cloudlet,

$$U_i = Ue_i + Ud_i$$

where, $Ue_i$ is the execution cost of user cloudlet, $Ud_i$ is the data transfer cost of $i^{th}$ user cloudlet.

Cost of $j^{th}$ log cloudlet,

$$L_j = Le_j + Ld_j + Ls_j$$

Where, $Le_j$ is the execution cost of $j^{th}$ logcloudlet, where, $Ld_j$ is the execution cost of $j^{th}$ log cloudlet, $Ls_j$ is the data transfer cost of $j^{th}$ log cloudlet

$Le_j, Ue_i=$ length of cloudlet * cost of processing of logs and user cloudlets

$Ld_j, Ud_i=$ input data transfer cost +output data transfer cost

= (input size +output size)*cost of bandwidth

We will be taking average input and output sizes for user and log cloudlets (see table).

$Ls_j =$ output size * storage cost at VM

For a system dealing with $a$ number of user requests and on an average $b$ number of cloudlets per users then total number of user cloudlets will be

$$n = a \times b$$

If on an average one user generates $c$ number of logs (that needs to be processed) then number of log cloudlets will be
\[ l = a \times c \]

In general, the total cost will be

\[ C = \sum_{i=1}^{n} U_i + \sum_{j=1}^{l} L_j \]

Considering the cost incurred due to some unknown or non-assured cases will be included as overhead cost. The overhead cost \( o \) for every log cloudlets have to be added for each log.

Hence,

\[ C = \sum_{i=1}^{n} U_i + \sum_{j=1}^{l} L_j + (l \times o) \]

Substituting all, we get

\[ C = \sum_{i=1}^{n} (Ue_i + Ud_i) + \sum_{j=1}^{l} (Le_j + Ld_j + Ls_j) + (l \times o) \]

**Leased:**

In the leased case, the datacenter host is taken on lease basis. The lease can depend basically on number of days. However, since the daily number of user requests may vary significantly, the leasing can be based on number of VMs being used or allocated. The leased cost will not cover the data transfers of user cloudlets. In case of leased datacenter, the cost of that will incur due to user requests thus, will depend only on that of data transfer cost.

Hence,
\[ U_i = U d_i \]

The cost incurred due to logs will be similar to the pay-as-you-go case. This is because we are studying the overall cost implications due to monitoring, and at present we can’t make assumptions of a leased monitoring.

Thus,

The overall cost in case of leased will be

\[ C = \sum_{i=1}^{n} U d_i + \sum_{j=1}^{l} (L e_j + L d_j + L s_j) + (l \times o) + DC\text{cost} \]

The overall costing methodology is depicted in figure 5.14

![Cost calculation at datacenter](image)

Figure 5.14: Cost calculation at datacenter.
The datacenter cost ($DC_{cost}$) are calculated in case of leased hosts. Thus, for every VM on an assigned SLA will incur cost in the leased case. However, the execution costs of cloudlets will not be included along with the storage costs.

If the leased price of each VM is taken as $e_k$ for $r^{th}$ VM then 

$$e_k = mc_r * sc_r * e_{ck}$$

Where, $mc, sc, ec$ are memory cost, storage cost for $r^{th}$ VM and execution cost for the $k^{th}$ day respectively.

Therefore, the total cost incurred at the datacenter for $m$ days leasing $v$ number of VMs will be:

$$DC_{cost} = \sum_{r=1}^{v} (mc_r + sc_r) + \sum_{k=1}^{m} e_{ck}$$

**The Simulation**

The simulation was done on both topology 1 and topology 2 (figure 5.13). The inputs in both topologies are as listed in table 5.2. The input length of user cloudlets, start times of each cloudlet varies uniformly between 0-2000 for the former and 0 to 28800 (1 day) for each day. Also the number of users logged in per day varies from 0 to number of users in the organization. The major reason for taking uniform distribution is that, we are mostly concerned to know the trends of cost implications and in case of any other empirical distributions the cost implications might show non-natural trends which might lead us to wrong interpretations.
Table 5.2a: Host configurations

<table>
<thead>
<tr>
<th>No of hosts</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual core</td>
<td>1000MIPS each</td>
</tr>
<tr>
<td>Linux</td>
<td>x86, Xen</td>
</tr>
<tr>
<td>RAM</td>
<td>2GB</td>
</tr>
<tr>
<td>storage</td>
<td>100000000</td>
</tr>
<tr>
<td>bw</td>
<td>100000</td>
</tr>
<tr>
<td>Average No of Cloudlets Per user</td>
<td>10</td>
</tr>
<tr>
<td>Days</td>
<td>30</td>
</tr>
<tr>
<td>Avg no of logs per user</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5.2b: Test configurations

<table>
<thead>
<tr>
<th>Methodology</th>
<th>No of VMs</th>
<th>Monitored?</th>
<th>No of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pay as you go</td>
<td>2</td>
<td>Y</td>
<td>1 VM</td>
</tr>
<tr>
<td>Pay as you go</td>
<td>2</td>
<td>Y</td>
<td>shared</td>
</tr>
<tr>
<td>Pay as you go</td>
<td>2</td>
<td>N</td>
<td>NA</td>
</tr>
<tr>
<td>Leased</td>
<td>2</td>
<td>Y</td>
<td>1 VM</td>
</tr>
<tr>
<td>Leased</td>
<td>2</td>
<td>Y</td>
<td>shared</td>
</tr>
<tr>
<td>Leased</td>
<td>2</td>
<td>N</td>
<td>NA</td>
</tr>
</tbody>
</table>

Table 5.3: price parameters

<table>
<thead>
<tr>
<th>Item</th>
<th>ratio</th>
<th>Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>storage cost</td>
<td>0.0010</td>
<td>0.0001/byte</td>
</tr>
<tr>
<td>bw cost</td>
<td>0.010</td>
<td>0.001/byte</td>
</tr>
<tr>
<td>proc cost</td>
<td>30</td>
<td>3/sec/VM</td>
</tr>
<tr>
<td>memory cost</td>
<td>0.0010</td>
<td>0.0001/byte</td>
</tr>
<tr>
<td>PAYUG proc</td>
<td>1.00</td>
<td>0.1/sec</td>
</tr>
</tbody>
</table>

The simulation will run for both pay-as-you-go as well as leased cases. There will be two VMs allocated to the broker and one VM for monitor (if any) at data centre. The cases for which simulations are carried out are listed in table 5.2b. The basic configuration of host is also listed in table 5.2a. The pricing per entity is listed in table 5.3.