Resource Management in a Hybrid Cloud Infrastructure

Kaustav Choudhury
Department of Computer Science & Engineering
Heritage Institute of Technology

Diptam Dutta
Department of Computer Science & Engineering
Heritage Institute of Technology

Kasturi Sasmal
Department of Computer Science & Engineering
Heritage Institute of Technology

ABSTRACT
In this paper, a resource management technique is proposed to handle the request of Virtual Machines (VM's) as per the need of users, consisting of resources (mips, vm image size, network bandwidth, number of cpu's) containing cloudlets which in turn are cloud-based application services (content delivery, social networks) commonly deployed in data centers. There are three priority mechanisms governing the user's requests namely low, medium and high priority requests. Here, the priority is given to the amount of VM's requested. To implement the above concept, the Hybrid Cloud model is used by which the benefits of both the private and public clouds can be reaped. This model has its advantages that it proves to be cost-effective as the resources are effectively utilized from private clouds and only when exhausted are taken from public clouds which is cheaper.

General Terms
Cloud Computing, Distributed and Parallel Systems.

Keywords
Cloud Computing, Hybrid Cloud, Resource Management.

1. INTRODUCTION
Cloud Computing can be viewed as an abstraction of software and hardware resources which are outsourced to an organization or a user on a pay-per-usage basis.

The architecture followed by a Cloud infrastructure is that of the combination of parallel and distributed type where the resources are dynamically provisioned over the Internet through Service Level Agreements between the consumer and the Cloud Service Provider. [1]

Cloud Computing is based upon service-based resources. Resource management is the effective deployment of an organization's resources when they are required. Resource management system in data centers are supported by Service Level Agreement (SLA)-oriented resource allocation methods. The SLA Resource Allocator serves as the interface between the Cloud Infrastructure and the external users or broker. The user interacts with the Cloud Management Systems through the broker who acts on the users behalf for the submission of service requests from anywhere in the world to the Cloud Infrastructure. The resource provisioning stage is the time when the cloud broker makes a decision to provision the resources on-demand and to allocate the VMs to cloud providers for utilizing these resources. Here, all the requests of users for the resources are addressed by the Cloud Services Manager (CSM) which manages all resources and their corresponding costs. Next, the CSM interacts with the Virtual Infrastructure Manager (VIM) for meeting the user's resource requests. The VIM next interacts with the Data Center Broker to fulfill these requests from the available existing resources. The Cloud Brokers act as intermediaries between the end users and cloud providers. The multiple VMs can concurrently run applications based on different operating system environments on a single physical machine. The multiple virtual machines are started and stopped in a dynamic fashion to meet the accepted services requests. Physical hosts, Data Center Broker and the VIM comprise a Cloud. These are the key components of a Cloud at the Infrastructure as a Service layer. This layer is also known as the Hardware as a Service layer. When a user requests for resources, the Resource Manager accepts the request and tries to fulfill them by allocating these resources first from the private cloud. More preference of resource utilization is given to private cloud as they reside within the organization itself. If the resources are not available in the private cloud (due to the limited scalability of private cloud) then the Resource Manager allocates the resources from the public cloud by interacting with its CSM. [2]

2. RELATED WORK
In [2] the authors have categorized the user's request into two types based on their resource requirements that is critical data processing and data security. These requests are assigned according to priority, if the user's need to perform critical data processing or when security demand is high then the request is classified as a high priority and if the request is to run non critical tasks then it classified as low priority. The Resource Manager recognizes the suitable cloud to be used to fulfill a request. The high priority request always accesses resources from the private cloud itself, because it has confidential (secure) information. The next low priority requests can be fulfilled from either public cloud or private cloud. But if the private cloud resources are available, it must be used first as these resources are possessed by the enterprise and should be utilized.

3. PROPOSED WORK AND ARCHITECTURAL ASPECTS
In this paper a proposal is made to implement a priority-based Resource Manager which will address the request of virtual machines (VMs) as per the need of users, consisting of resources (mips, vm image size, network bandwidth, number of cpu's) containing cloudlets which in turn are cloud-based application services (content delivery, social networks) generally deployed in data centers [3].

There are three priority mechanisms governing the user's requests namely low, medium and high priority requests. Low and medium priority requests for 1 or 2 VM's respectively will be sent to the private cloud and high priority requests (i.e. 3 VM's) to the private cloud first (2 VMs) and then the

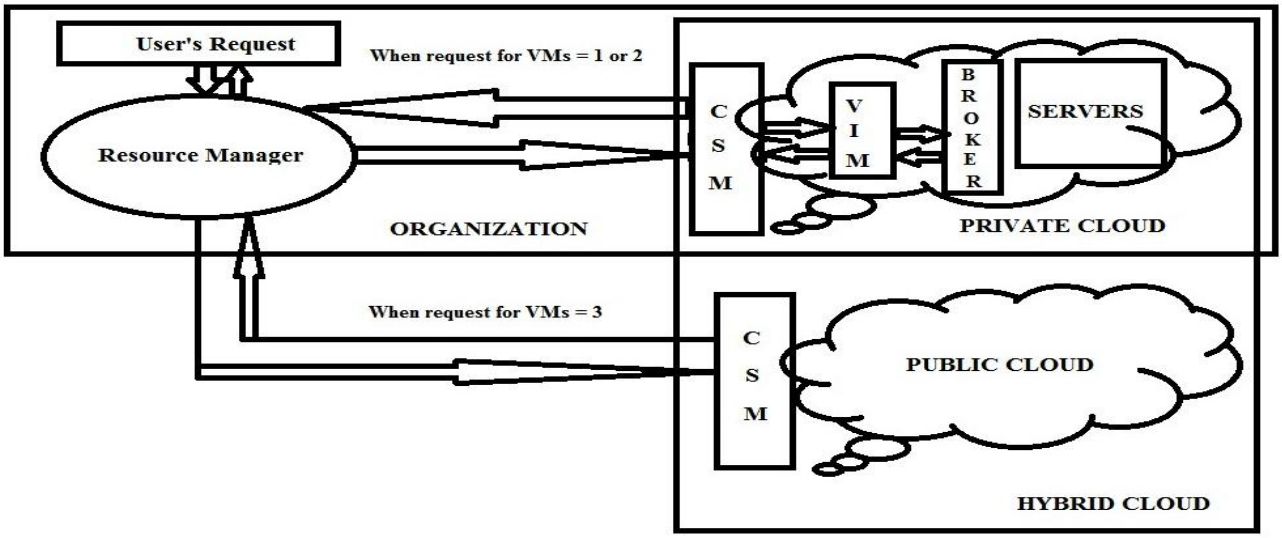

Fig 1: Architecture of the Resource Manager

remaining to the public cloud. Here, the priority is given to the private cloud server on the basis of the amount of VM's requested as the private cloud servers belong to the enterprise and should be utilized first being also more secure.

The implication is that private cloud servers lie within an organization, the cost of which is the liability of the organization. Hence, the scalability of the private cloud servers is less. Now, the problem starts when the need of the organization increases and there is a deficit in ample amount of resources. Overloading may take place due to excessive demands where the cloud server gets exhausted of resources. To overcome this limitation, help is taken from public cloud servers which reside outside the organization. The excess requests are redirected to the public cloud servers which increases the overall scalability of the cloud. The public cloud providers offer services at a cheaper cost which would be rather expensive if the same had to be incurred by the enterprise.

Also it must be taken into account that private cloud servers are more secure compared to public cloud servers. This is due to the presence of unauthenticated users in the public cloud as most of these servers are freely available for use to the general public. So, the request for VMs is first met from the private cloud and when exhausted the request is redirected to the public cloud server.

To implement the above concept, the Hybrid Cloud model is used by which the benefits of both the private and public clouds can be reaped. The Hybrid Cloud environment consists of multiple internal and external providers. The Hybrid Cloud is a combination of both public and private clouds. It can help to provide on-demand, externally provisioned scalability, higher security and efficiency.

This model has its benefits that it proves to be cost-effective as the resources are effectively utilized from private clouds and only when exhausted are taken from public clouds which is cheaper. Also, the scalability of the private cloud is increased which ensures that the user's requests are met.

The architecture of the Resource Manager is depicted above in the Fig.1.

Legend: CSM – Cloud Services Manager, VIM – Virtual Infrastructure Manager.

4. PROPOSED ALGORITHM

4.1 Algorithm for Resource Manager

Step 1: Create the Java package priorityBasedResourceMgr.

Step 2: Create the class priorityResourceMgr which inherits the HybridCloud class.

Step 3: Take number of VMs as input from the user.

Step 4: Call the ResourceMgr ( ) function to do the necessary allocation as per the no. of VM's.

Step 5: Function ResourceMgr (int req_vms)

5.1: Initialize variable 'vmcnt' for counting the number of VMs in the private cloud, equal to zero.

5.2: Request Type 1: Low Priority Request

If the requested number of VMs i.e. 'req_vms' is equal to 1

Call the function pvtcldCreateVM ( ).

End If

5.3: Request Type 2: Medium Priority Request

If the requested number of VMs i.e. 'req_vms' is equal to 2

Call the function pvtcldCreateTwoVM ( ).

End If

5.4: Request Type 3: High Priority Request

If the requested number of VMs i.e. 'req_vms' is equal to 3

Call the function pvtcldCreateTwoVM ( ).

Count the number of VMs created in the private cloud.

If 'vmcnt' is less than allotted space for private cloud

Print "No space in Private Cloud Server.

Hence, redirecting to Public Cloud Server".

Call function pubcldCreateVM.

End If

Call function total_debt ( ) to display the total expenditure.

End If

5.5: End Function ResourceMgr ( ).

4.2 Algorithm for creating the Hybrid Cloud Infrastructure

Step 1: Create the Private Cloud with 1 VM

1.1: Initialize the CloudSim package.

1.2: Initialize the CloudSim library.

1.3: Create a Datacenter. Datacenters are the resource providers in CloudSim.

1.4: Create a Datacenter Broker.

1.5: Create one virtual machine, define the properties and then add it to the vm list.

1.6: Submit vm list to the broker.

1.7: Create one Cloudlet, define its properties and then add it to the cloudlet list.

1.8: Submit cloudlet list to the broker.

1.9: Start the simulation.

1.10: Print results when simulation is over also with the debt of each user to each datacenter.

Step 2: Create the Private Cloud with 2 VMs in the same way as above following Steps 1.1 through 1.10.

Step 3: Create the Public Cloud with required amount of VMs in the same way as above following Steps 1.1 through 1.10.

Step 4: Write a function to return the number of VMs created in private cloud.

Step 5: Write a function to return the total debt for the hybrid cloud.

Step 6: Create a Power Data Center for Private Cloud.

6.1: Create a list to store the machine.

6.2: Create PEs (Processing Elements) and add these into a list.

6.3: Create Host with its id and list of PEs and add them to the list of machines.

6.4: Create an object of the DatacenterCharacteristics class that stores the properties of a data center: architecture, OS, list of Machines, allocation policy: time- or space-shared, time zone and its price (G$/Pe time unit).

6.5: Finally, create a PowerDatacenter object.

Step 7: Create a Power Data Center for Public Cloud and make required changes to the DatacenterCharacteristics object.

Step 8: Write the function to create a broker.

Step 9: Write the function to print the Cloudlet objects.

Note: In this algorithm the space of the Private Cloud is restricted to 2 VMs. It can be increased according to the user's implementation.

The above algorithms need to be converted to their Java counterpart following the guidelines of Cloudsim 3.0 [4].

5. TOOLS USED

Cloudsim 3.0 [6], [7] is used and integrated within Eclipse (Indigo) in Windows 7 environment. Cloudsim helps to effectively simulate a real cloud infrastructure with options of flexible scalability in a single computing node and test the model repeatedly [3].

The Cloudsim toolkit is downloaded and installed following some steps [5]. Eclipse Indigo [8], a Java editor to write and run the Java codes is downloaded and installed. Windows 7 is the operating system.

6. RESULTS PRODUCED

Run the Java program for the Resource Manager algorithm with the following cases.

Case 1: When number of VMs = 1

Enter the number of virtual machines you want to create:- 1

Initialising...

Starting CloudSim version 3.0

Datacenter_0 is starting...

Broker is starting...

Entities started.

0.0: Broker: Cloud Resource List received with 1 resource(s)

0.0: Broker: Trying to Create VM #0 in Datacenter_0

0.1: Broker: VM #0 has been created in Datacenter #2, Host#0

0.1: Broker: Sending cloudlet 0 to VM #0

400.1: Broker: Cloudlet 0 received

400.1: Broker: All Cloudlets executed. Finishing...

400.1: Broker: Destroying VM #0

Broker is shutting down...

Simulation: No more future events

CloudInformationService: Notify all CloudSim entities for shutting down.

Datacenter_0 is shutting down...

Broker is shutting down...

Simulation completed.

Simulation completed.

The Output is:-

Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time

0 SUCCESS 2 0 400 0.1 400.1

*****Datacenter: Datacenter_0*****

User id Debt

3 35.6

****

Finished Creating 1 VM

Case 2: When number of VMs = 2

Enter the number of virtual machines you want to create:- 2

Initialising...

Starting CloudSim version 3.0

Datacenter_0 is starting...

Broker is starting...

Entities started.

0.0: Broker: Cloud Resource List received with 1 resource(s)

0.0: Broker: Trying to Create VM #0 in Datacenter_0

0.0: Broker: Trying to Create VM #1 in Datacenter_0

0.1: Broker: VM #0 has been created in Datacenter #2, Host#0

0.1: Broker: VM #1 has been created in Datacenter #2, Host#0

0.1: Broker: Sending cloudlet 0 to VM #0

0.1: Broker: Sending cloudlet 1 to VM #1

4000.1: Broker: Cloudlet 0 received

4000.1: Broker: Cloudlet 1 received

4000.1: Broker: All Cloudlets executed. Finishing...

4000.1: Broker: Destroying VM #0

4000.1: Broker: Destroying VM #1

Broker is shutting down...

Simulation: No more future events

CloudInformationService: Notify all CloudSim entities for shutting down.

Datacenter_0 is shutting down...

Broker is shutting down...

Simulation completed.

Simulation completed.

The Output is:-

Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time

0 SUCCESS 2 0 4000 0.1 4000.1

1 SUCCESS 2 1 4000 0.1 4000.1

*****Datacenter: Datacenter_0*****

User id Debt

3 71.2

****

Finished Creating 2 VMs

Case 3: When number of VMs = 3

Enter the number of virtual machines you want to create:- 3

Initialising...

Starting CloudSim version 3.0

Datacenter_0 is starting...

Broker is starting...

Entities started.

0.0: Broker: Cloud Resource List received with 1 resource(s)

0.0: Broker: Trying to Create VM #0 in Datacenter_0

0.0: Broker: Trying to Create VM #1 in Datacenter_0

0.1: Broker: VM #0 has been created in Datacenter #2, Host#0

0.1: Broker: VM #1 has been created in Datacenter #2, Host#0

0.1: Broker: Sending cloudlet 0 to VM #0

0.1: Broker: Sending cloudlet 1 to VM #1

4000.1: Broker: Cloudlet 0 received

4000.1: Broker: Cloudlet 1 received

4000.1: Broker: All Cloudlets executed. Finishing...

4000.1: Broker: Destroying VM #0

4000.1: Broker: Destroying VM #1

Broker is shutting down...

Simulation: No more future events

CloudInformationService: Notify all CloudSim entities for shutting down.

Datacenter_0 is shutting down...

Broker is shutting down...

Simulation completed.

Simulation completed.

The Output is:-

Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time

0 SUCCESS 2 0 4000 0.1 4000.1

1 SUCCESS 2 1 4000 0.1 4000.1

*****Datacenter: Datacenter_0*****

User id Debt

3 71.2

****

Finished Creating 2 VMs

Number of VMs created in Private Cloud Server = 2

No space in Private Cloud Server. Hence redirecting to Public Cloud Server

Initialising...

Starting CloudSim version 3.0

Datacenter_0 is starting...

Broker is starting...

Entities started.

0.0: Broker: Cloud Resource List received with 1 resource(s)

0.0: Broker: Trying to Create VM #0 in Datacenter_0

0.1: Broker: VM #0 has been created in Datacenter #2, Host#0

0.1: Broker: Sending cloudlet 0 to VM #0

4000.1: Broker: Cloudlet 0 received

4000.1: Broker: All Cloudlets executed. Finishing...

4000.1: Broker: Destroying VM #0

Broker is shutting down...

Simulation: No more future events

CloudInformationService: Notify all CloudSim entities for shutting down.

Datacenter_0 is shutting down...

Broker is shutting down...

Simulation completed.

Simulation completed.

The Output is:-

Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time

0 SUCCESS 2 0 4000 0.1 4000.1

*****Datacenter: Datacenter_0*****

User id Debt

3 15.12

****

Finished Creating 1 VM in the Public Cloud

The total no. of VMs created is 3

The total debt is:- 86.32

*As it can be noted from the above output that, if all three VMs were taken from the private cloud server then the total debt would have been **106.8** which has been reduced to **86.32**.*

Note: The above Java programs should be created in a package and should reside in the Cloudsim project folder.

7. CONCLUSION

In this work a resource management technique is proposed for a Hybrid Cloud model which produces satisfactory results in decreasing the overall expenditure of the organization. This proves highly beneficial for a particular organization consuming cloud based services and also for the Cloud Service Providers to provide cloud services in a more effective and cheaper way. In the current scenario, just decreasing the IT expenditure is not enough for an organization to mark a niche in the market. It has to be responsible in cutting down the ever increasing energy requirements by employing suitable greener technologies. In this context, the future work can be to devise such a strategy of dynamic resource allocation which will reduce the overall energy consumption rate of the data centers in the Cloud and to strive in making a greener Cloud Infrastructure. Such a strategy may involve in careful inspection of the CPU utilization, adjusting the system voltage and migrating the tasks in heavily loaded servers to the idle server resources. In this way total resource utilization is possible and energy can be saved.

8. REFERENCES

[1] Kaustav Choudhury, Argha Roy, Diptam Dutta. *Visualizing a Cloud using Eucalyptus and Xen*. International Journal of Advanced Research in Computer Science and Software Engineering. Volume 3, Issue 3, March 2013, pp.482-487.

[2] Rajkamal Kaur Grewal, Pushpendra Kumar Pateriya. *A Rule-Based Approach for Effective Resource Provisioning in Hybrid Cloud Environment*. International Journal of Computer Science and Informatics. Volume 1, Issue 4, 2012, pp.101-106.

[3] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose and Rajkumar Buyya. *CloudSim - A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms*. SOFTWARE - PRACTICE AND EXPERIENCE 2011, Wiley Online Library, 24 August 2010, pp.23–50.

[4] Cloudsim Toolkit Example packages.

[5] "HOW TO INSTALL CLOUDSIM 3.0 AT ECLIPSE", http://ajithphd.blogspot.in/2012/02/how-to-install-cloudsim-30-at-eclipse.html

[6] "CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services", https://code.google.com/p/cloudsim/

[7] "Cloudsim Download Page", http://code.google.com/p/cloudsim/downloads/.

[8] "Eclipse Indigo Download Page", http://www.eclipse.org/downloads/packages/release/indigo/sr2.