Cost Optimized Virtual Machine Deployment in Eucalyptus for Autonomous Systems

Kamalesh Karmakar
Assistant Professor
Department of Computer Science & Engineering and Information Technology,
Meghnad Saha Institute of Technology, Kolkata, India

Anesul Mandal
Student of M.Tech,
Department of Computer Science & Engineering,
Meghnad Saha Institute of Technology, Kolkata, India.

Abstract- Cloud is a pool of virtualized computer resources. It is the next stage in the Internet's evolution, providing the means through which everything - from computing power to computing infrastructure, applications, business processes to personal collaboration - can be delivered to you as a service wherever and whenever you need. In this research, Virtual Machine's health (CPU and Memory usage) is being monitored and decisions are taken to scale them up/down. The monitoring system is running in autonomous system. Here administrator uses key based authentication for agent-less monitoring of VMs.

Keywords – VM Monitoring, VM Performance, Cost Optimization, VM Deployment

I. INTRODUCTION

This research work is based on Cost Optimized Virtual Machine(s) (VMs) [1] Deployment in Eucalyptus [2] based Private Cloud environment[3] for Autonomous System [4]. In this research Eucalyptus tool, which is available under GPL License, has been used for creating and managing private cloud. Eucalyptus is composed of three types of components, such as Cloud Controller, Cluster Controller and Node Controller. Load Balancing and VM deployment logic runs in Cloud Controller, Cluster Controller manages VM resources and VMs run in registered Node Controllers of the Cloud Environment. This work focuses monitoring and management of VM resources in an autonomous system. An Autonomous system is either a single network or a group of networks that is controlled by a common network administrator on behalf of a single entity like university of a business division.

In this research work VM's performance is being monitored. Here a user of a Cloud service is being considered as an administrator of an Autonomous system. All the VM(s) under same administrative domain are considered as Autonomous System. This research work proposes a monitoring system [5] which continuously checks the status of the performance of VM(s) to take necessary action if required.

User can analyze those VM(s) which are running in his access domain. From security standpoint, a virtualized environment faces more risk than a physical environment. Running multiple VMs on a single physical host creates a single point of failure. It may be physical server failure or network failure. So, monitoring all the VMs is necessary to ensure reliability, availability, stability and cost optimization of virtualized environment. In this work monitoring attributes are the CPU Usage and Memory Usage. This application resolves some of the well-known problems like VMs running out of memory and under-utilized CPU resource. The monitoring system is notified when memory usage is very high or very low. Similarly it is being notified when CPU usage is very high or extremely low. This helps taking necessary action on right time.

If the usage of VMs is extremely low over the time, this helps to take a decision for scale down. On the contrary if the usage of the VMs is extremely high over time that means decision should be taken for scale up. This makes VMs deployment cost optimized.

Cost optimized resource deployment [6] in Cloud means stable, cost-effective and use of optimal number of VM instances. This requires continuous tracking of performance of each VM instance. Hence to achieve this optimization a simple powerful approach is given below.

1. View the usage and trend of each VM instances.

2. Find out the under-utilized VM(s).
3. Cost-optimized reservation of Cloud resources.

Cloud service providers provide two types of plans; one is provisioned by reservation plan and other one is on-demand plan. A hybrid plan i.e. combination of both plans is effective for this research work. This paper proposes a concept for the above two points which helps achieving point 3.

There are a number of open source applications for computer system monitoring, network monitoring and infrastructure monitoring available in the market. Nagios, Zabbix and Ganglia are some of system monitoring applications and widely used. Nagios has number of plugins (NRPE, NRDP, NSClient++) which works as agent and needs to be installed and running in remote host. Zabbix works primarily via an agent that runs on each host that needs to be monitored. Ganglia is widely used for high-performance computing systems like clusters and grids. Ganglia Monitoring Daemon (gmond) is a lightweight service that needs to be installed on every host to be monitored.

The most remarkable features of these monitoring software is that, to monitor remote hosts an Agent needs to be installed on all remote hosts. The Agent collects host's health status and sends it to the monitoring server. The big advantage of the proposed architecture is that, it is very simple to install and configure. The monitored VMs are not aware of adding or removing them in the monitoring VM list of monitoring server. In this context section 2 describes the proposed architecture in details followed by experimental results in section 3.

II. Proposed Architecture

In a Cloud environment a user can reserve VM(s) for specific duration at the time of deployment from the Cloud Service Provider. When applications are running on those VMs, those VMs can be overloaded or under-utilized. But user does not have any information about which VM is idle for which VM is overloaded. Hence decision should be taken manually to scale VM resources. And for this reason cost optimization is not possible.

To decide the minimum number of VMs requirement dynamically, usage of all the VMs should be tracked. This monitoring software tracks VMs in Autonomous System. This paper proposes a model for monitoring software that has only server component and unlike other monitoring software; no agent is required to be installed in VMs. The following diagram describes cooperation of the proposed monitoring engine with Eucalyptus Cloud.

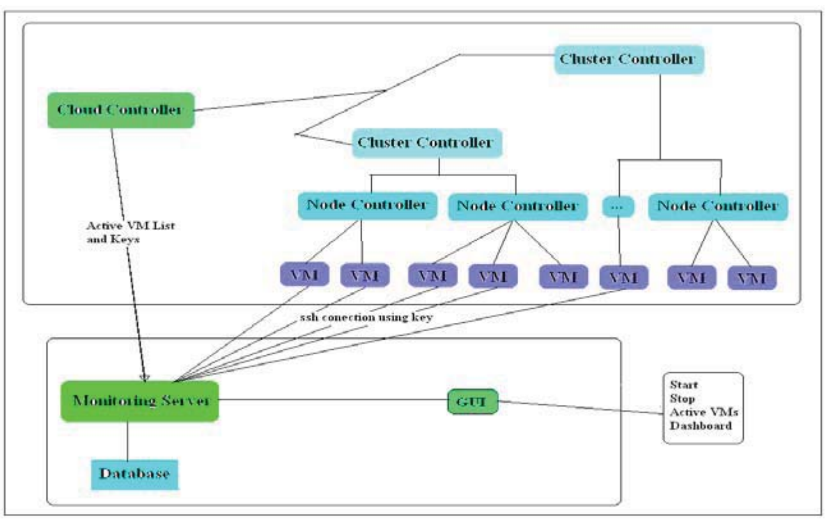

Figure 1. Proposed Architecture

When a user works with Cloud environment, he has his own admin domain where all the VM instances are running and those are associated with a key. This key is used to run the VM and to authenticate user to perform any operation on this instance. The user needs to monitor all the VMs under his control. That means he needs to track the current CPU usage and Memory usage for all the VM instances he created. This architecture works in pull mode. That means there is no need to install any software or agent to the VM and push the data to the monitoring server, rather the monitoring sever will connect to the VM instance and collect data from that VM and store it in server database. The monitoring server will collect the list of current active VMs from Cloud Controller (CC) [2] and will dynamically add to or remove from the list of monitoring VMs maintained by the monitoring server. More specifically, the monitoring server periodically collects the VM list under the current user domain from Cloud Controller and updates its own list.

The Overall workflow of the proposed architecture is shown in Figure 2.

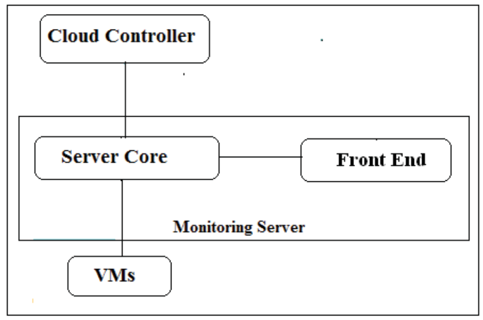

Figure 2. Overall Workflow

The monitoring server has two major components; these are core component and UI component. In this context the operation principle of the core component of the proposed architecture is described in sub-section 2.1 and that of UI component is described in sub-section 2.2.

A. Core Component –

The core component is the Monitoring Engine which is the heart of the system. This component collects active VMs list and communicates with them to get the values of different performance attributes. This component communicates with Eucalyptus Cloud Controller and individual VM. The following diagram shows the operation flow of this component. The detail functions are described below.

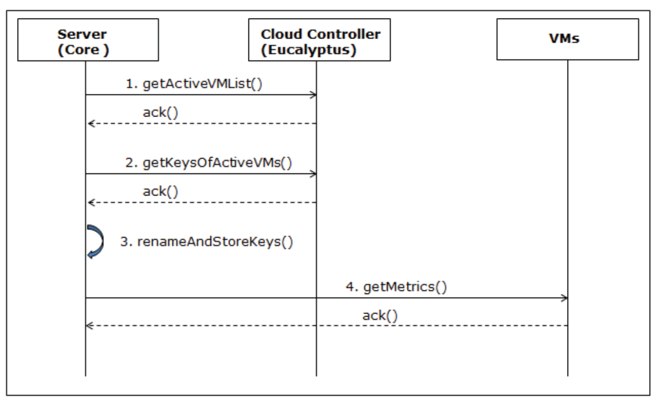

Figure 3. Operation Flow of Core Component

The functions of this component:

1) A user starts up the monitoring sever. The server communicates to the Eucalyptus Cloud Controller and gets the list of active VMs under the current user domain.
2) The server then gets the keys of all active VMs from Cloud Controller.
3) Rename the keys and store in local directory at a particular location. The name of the keys should follow a particular pattern derived from the IP of the VM.
4) The server does steps 1, 2 and 3 periodically in a particular time interval. If any new VM is found in cloud controller the server stores the IP of the VM and its keys. Any VM that exists with the server but does not exists in the new list, and then it is removed from the server list.
5) The server then connects to the remote VM through SSH. The IP of the VM is already exists with the server. For SSH connection the key of the VM is used. This ensures the authentication of the server to the remote VM and no

user/password verification for login is required. SSH provides strong authentication and secure communications over insecure channels. It executes commands in a remote host and sends the output back to the calling host. In this research work SSH uses key of the VM for remote connection, hence the remote host trusts the calling server.

6) Step 5 is carried out for all VMs available with the server. The collected metrics for CPU Usage, Memory Usages of the remote VMs are stored in history table.

7) Steps 5 and 6 are carried out periodically for a time interval.

Hence the server will store metrics for all VMs over time in history table. This table is later used for graphical representation. In this context, sub-section 2.2 describes the UI components and its operation flow and the graphical representation of the metrics are described in section 3, the Experimental Results section.

B. UI Component –

UI component is integral part of monitoring server. This component is actually controlled by the monitoring server. This provides an interface to the user to interact with the server. A user can start, stop monitoring and view the graph of the metrics. This component is responsible for generating dashboard as per user requirement. This will show current status of monitoring parameters of all active VMs under the current user. The detail functions of this component are described below..

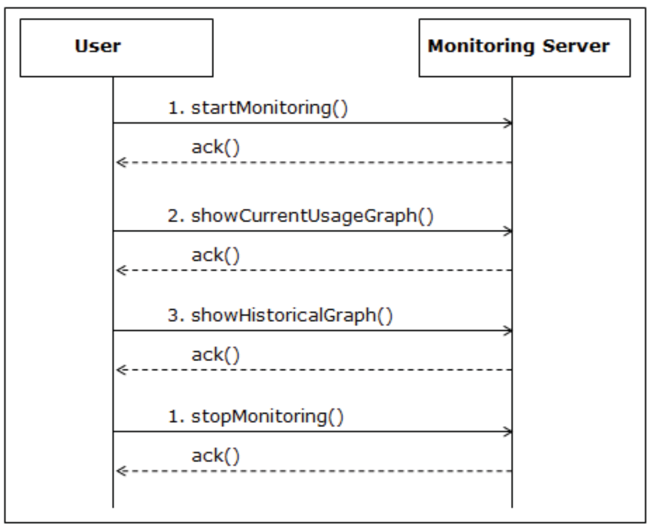

Figure 4. Operation Flow of UI Component

The functions of this component:

1) A user logs in to the system. If the server is already in monitoring mode, then the screen will show “Monitoring started” otherwise will show “Monitoring Stopped”.

2) The graph section of the screen will show the current status graph for CPU usage and Memory usage for all VMs in two separate graphs. In more details if 5 VMs are considered then one graph shows the latest CPU usage of 5 VMs in 5 line charts and another graph shows the Memory usage of 5 VMs in 5 line charts. User has the facility to change the time slot and refresh the graph.

3) User can generate line chart of any number of VMs and for any time period.

4) User can invoke monitoring start and stop command at any time to the server through UI. This will start or stop monitoring for all the VMs at a time.

5) There is no facility for start or stop monitoring for selective VMs. This is quite simple only needs some enhancement in the UI.

6) User can view historical graph of the metrics for selective VMs for a selective time period.

III. COST OPTIMIZED VM DEPLOYMENT TECHNIQUE

CPU is one of the primary resources in computer system. Monitoring CPU utilization is necessary for scheduling jobs and VM scale up and scale down. The service providers offer two types of plan for VM reservation, On-demand plan and Monthly plan. The minimum time of reservation for On-demand plan is 1 hour. Cost is calculated based on number of VMs reserved for how many hours. In this work cost optimization [7, 8] is done by minimizing the usage of number of VMs. This concept is applicable for on-demand plan only. VM deployment can be classified into two types…

- Static deployment
- Dynamic deployment

A. Static Deployment

Static deployment means based on volume of jobs and VM configuration a number of VMs are deployed for a time period based on prediction. Here degree of optimization depends on how perfectly the prediction is made. In this deployment VMs cannot be dynamically removed or added. VMs allocated for a set jobs, remain attached with that set jobs unless their reservation time expire.

B. Dynamic Deployment

In case of dynamic deployment, based on volume of jobs a minimum set of VMs are initially deployed. This minimum set is based on a prediction for the volume of jobs and VM configuration. Using a monitoring system VMs are continuously monitored for the CPU and Memory usage. This tracking information helps taking decision for deploying more VM(s) or removing VM(s). The present work focuses on this deployment which minimizes the total cost.

The monitoring system tracks the CPU usage and Memory usage of all VMs. If the usage of these metrics for all the VMs cross the upper boundary, then scale up is required. On the contrary, if the usage of these metrics goes below the lower boundary then scale down is required. The upper boundary and lower boundary are configurable items.

The present work uses a simple logic for scale up and scale down of VMs…

1) If the usage of all the VMs crosses the upper boundary, then go for adding one VM.
2) If usage of a VM goes below the lower boundary, then de-allocate that VM.
3) If the usage of all the VMs goes below the middle boundary, then de-allocate a VM which is nearest to its expiry time. Middle boundary is also configurable item.

Figure 5 and 6 shows how cost is optimized in case of dynamic resource deployment. Here a particular set of jobs are considered and five VMs are deployed for processing the jobs.

Time / VMs	End of 1st Hour	End of 2nd Hour	End of 3rd Hour	End of 4th Hour	End of 5th Hour
VM1	70%	70%	40%	40%	41%
VM2	72%	72%	42%	42%	42%
VM3	74%	74%	35%	30%	36%
VM4	71%	71%	41%	41%	42%
VM5	10%	10%	10%	10%	10%

Figure 5. CPU Usage of each VM at the end of each hour in Static Deployment

Cost for Static deployment → Cost_S = 5 * 5 * C = 25C C = cost per VM for 1 hour

Time / VMs	End of 1st Hour	End of 2nd Hour	End of 3rd Hour	End of 4th Hour	End of 5th Hour
VM1	70%	80%	50%	60%	65%
VM2	72%	75%	55%	65%	70%
VM3	74%	73%	53%	53%	72%
VM4	71%	79%	50%	De-allocated	
VM5	10%	De-allocated			

Figure 6. CPU Usage of each VM at the end of each hour in dynamic Deployment

Cost for dynamic deployment → Cost_D = ((5*1) + (4*1) +(4*1) +(3*1) +(3*1))*C= 19C

Obviously Cost_D < Cost_S, i.e. dynamic deployment removes those VMs that are idle and also if the load of all the VMs are moderate, then a VM is removed and jobs are distributed among the reaming VMs. In this work lower boundary for CPU usage is 10%, upper boundary is 90% and middle boundary is 60%, which are some static values.

IV. EXPERIMENT AND RESULT

In this work two VM instances of CentOS 6 image are created which are being monitored. The monitoring server is installed and running on the host where Cloud Controller is installed. This host is operating on CentOS 6. Monitoring server is using apache Derby DB 10.10.1.1 release and jdk 1.7. For development of monitoring server eclipse Keplar 4.3 is used. The UI for monitoring and the graph are show in below figure.

Figure 7. Screenshot of Monitoring Interface

The screen in figure 5 show the monitoring server management screen through which one can continuously monitor the VMs and see the graphs. It shows the active VMs, its current CPU usage and Memory usage in tabular form. The two graphs for CPU usage and Memory usage are shown lower part of the screen. Here two VMs of IP 172.31.252.1 and 172.31.252.2 are being monitored in 2 minutes interval. Start time of monitoring is 2014-02-05 10:00:05 (yyyy-mm-dd hh:mm:ss).

The line chart in figure 6 shows the CPU usage of these two VMs. From the graph it is clear that CPU usage of the two VMs over the time is average. There is no abrupt change in the chart. No VM is idle or overloaded. If any of the line goes near the x-axis over some time then it can concluded that VM is idle. On the contrary if any of the lines goes near 100% over some time, then it can be said that VM is overloaded.

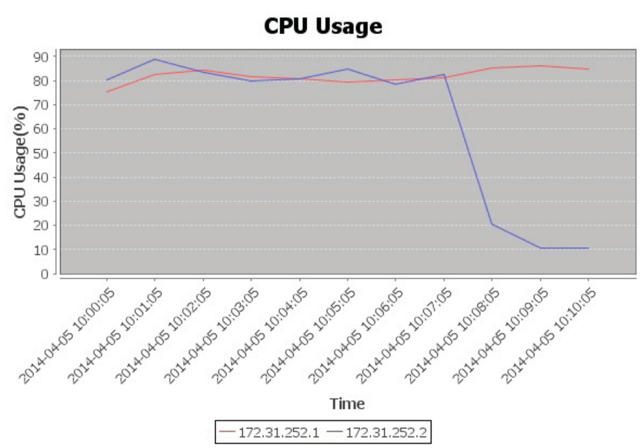

Figure 8. CPU Usage

Similarly the line chart in figure 7 shows the Memory usage of these two VMs. From the graph it is clear that Memory usage of the two VMs over the time is average. There is no abrupt change in the chart. No VM is idle or overloaded.

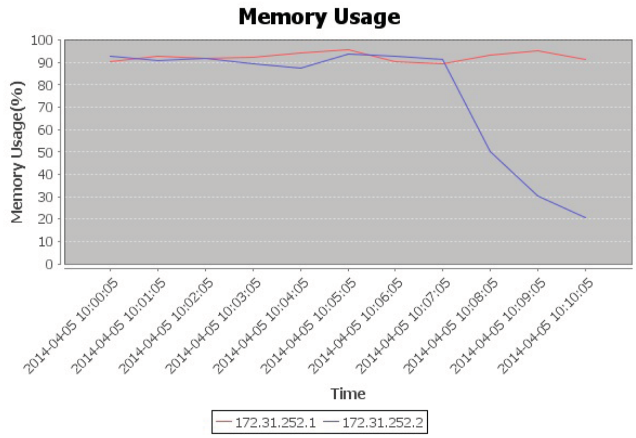

Figure 9. Memory Usage

V.CONCLUSION

Cloud computing is a rapidly accelerating revolution within IT and will become the default method of IT delivery. Now-a-days cloud services are widely used by large and small industry and even individuals. For cost optimization VM scale up and scale down is necessary and common activity. To take decision for scale up or scale down at right time the autonomous systems need to be monitored. Every time the health of each VM needs to be tracked. For monitoring the systems there are a number of tools freely available. The main drawbacks of those software tools is that the installation and configuration process is complex and need to install agent on VMs to be monitored, i.e. those VMs are aware of monitoring. But the main advantage of the proposed architecture is that it is very simple to install and configure. The VMs to be monitored are not aware of this monitoring. There is a scope of more enhancement and adding new feature in this architecture.

REFERENCES

[1] Chunxiao Li, Anand Raghunathan, Niraj K. Jha, "Secure Virtual Machine Execution under an Untrusted Management OS", Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, 978-0-7695-4130-3/10 IEEE, DOI 10.1109/CLOUD.2010.29.

[2] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D. The Eucalyptus Open-Source Cloud-Computing System. Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), Shanghai, China, May 18-21 May 2009.

[3] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", UCB Technical Report No. UCB/EECS-209-28, February 10 2009.

[4] D. Jiang, G. Pierre, and C.-H. Chi, "Autonomous resource provisioning for multi-service web applications," in Proceedings of the 19th international conference on World Wide Web, ser. WWW '10. New York, NY, USA: ACM, 2010.

[5] Kamalesh Karmakar, Tamal Mukherjee. "Virtual Machine Monitoring & Management in Cloud Environment". International Journal of Scientific & Engineering Research, Volume 5, Issue 1, January-2014.

[6] Anup H. Gade, "A Survey paper on Cloud Computing and its effective utilization with Virtualization", International Journal of Scientific & Engineering Research, Volume 4, Issue 12, December-2013.

[7] Sivadon Chaisiri, Bu-Sung Lee. "Optimization of Resource Provisioning Cost in Cloud Computing". IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 5, NO. 2, APRIL-JUNE 2012.

[8] A.Poobalan, V.Selvi. "Optimization of Cost in Cloud Computing Using OCRP Algorithm". International Journal of Engineering Trends and Technology (IJETT) – Volume 4 Issue 5- May 2013.