Virtual Machine Monitoring & Management in Cloud Environment

[1]Kamalesh Karmakar, [2]Tamal Mukherjee

Abstract— Cloud Computing is becoming a popular paradigm. In recent trend lot of new services are based on Cloud Environments, and a lot of people are using Cloud networks. Since different types of service providers coexist and their billing strategies are different, a VM monitoring system is being introduced to monitor VMs and to migrate VM in different Cloud. To automate VM migration a broker module has been introduced based on VM monitored data. In this paper a Broker Architecture has been designed to reduce effort of cost calculation and VM deployment.

Index Terms— Broker Interface, Cloud Service Provider, Hybrid Cloud Environment, Policy Comparator, Service Level Agreement, VM Migration, VM Monitoring.

1 Introduction

THIS research work is based on Monitoring and Management of Virtual Machines (VMs) [1] in Cloud Environment [2]. To provide automated dynamic resource allocation, deployed systems are monitored and decisions are taken to scale virtual resources. VM migration [3, 4] allows applications to dynamically change bindings between programs and physical hosts to manage fail over recovery, to maximize resource utilization and to reduce cost.

The popularity of large-scale cluster deployment [5, 6] in Cloud Environment increases day by day. To minimize the cost and maximize resource utilization, rented resources should be scaled dynamically. When some of these VMs are under-utilized it can be scaled down and vice versa. In this work, feasibility has been surveyed for the migration of VM. At the time of VM migration, downtime is being considered as negligible value.

Migration provides the ability to move a running VM between physical hosts with no interruption to service. Migration is transparent to the end user. VMs are migrated from one Cloud Environment to another if sufficient resources are not available in the hosted Cloud Environment or sufficient Private Cloud Resources are available to provide the service. Using migration, VM(s) can be seamlessly migrated without considerable downtime and without requiring any special support for application replication. It provides desirable features in modern computing like server consolidation, performance isolation and ease of management.

In this research Xen [7] is being used for Virtualization. There are two types of migration schemes, Live migration and Cold migration. In Live migration whole memory page tables are copied from current physical system to the target physical with transparency to the user without any service interruption. Using Live Migration, administrators can reduce planned downtime and improve their operational efficiency. In Cold migration VMs are halted before copying all its memory pages to the destination host and new VMs are restarted after successful copy of all memory pages. In Cold migration downtime is high compared to the Live migration technique for this reason in this research work Live migration has been considered.

In the present scenario VMs can be deployed and run in public Cloud Service Provider's (CSP) infrastructure directly by provided client interface. But deployment process is manual in nature. Service users should decide what to do if scale up or scale down is required. Even a less experienced service user may be confused to choose the best fit service provider according to their need.

In recent days many Cloud Service Provider (CSP) are available. To select the best provider and to manage and automate the VM deployment process, one VM management and monitoring module (Broker) module has been introduced.

In this context section 2 describe the proposed Architecture of Broker in details followed by different operational flow in section 3.

2 Proposed Architecture of Cloud Broker

Cloud Computing allows user to easily rent access to fully featured applications, computing infrastructure and data storage. A high level of compatibility can be maintained between legacy application and workloads in an IaaS Cloud, because IaaS Clouds allow users to install and run operating systems according to their need.

This paper proposes a model for automated monitoring and management of VM(s) using Broker Interface in gradually growing CSP's. Figure 1 depicts the proposed Architecture of Cloud Broker. Proposed Broker Architecture allows user to easily monitor VM resource usages to scale Virtual Resources according to the number of service requests.

- *Kamalesh Karmakar is currently working as Assistant Professor in the Department of Computer Science & Engineering and Information Technology in Meghnad Saha Institute of Technology, Kolkata, India, Ph-+91-9832777321. E-mail: k.karmakar.ju@gmail.com*
- *Tamal Mukherjee is currently doing some research work on Cloud Computing and working as Visiting Faculty in the Department of Computer Science & Engineering and Information Technology in Meghnad Saha Institute of Technology, Kolkata, India, Ph-+91-9476497855. E-mail: msit.tamal@gmail.com*

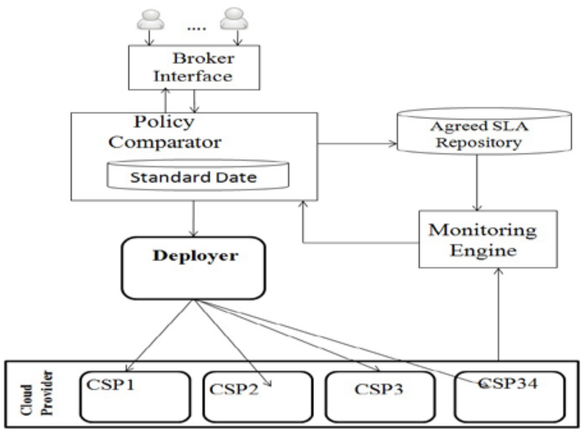


Fig. 1. Architecture of Proposed Cloud Broker.

User interacts to Broker Interface and Broker module dispatches the request to suitable CSP's Infrastructure. It aggregates the infrastructure demands from the user and evaluates them against the available services published by CSPs. Based on submitted user requirements; a Requirement Descriptor (RD) file is generated.

Policy Comparator (PC) finds out suitable CSP based on defined parameters of RD. If available resources are sufficient to fulfill user's need, PC generates Acknowledgement Message (AckMsg) to user and selects the CSP to reduce the cost and better performance; otherwise it will notify user about unavailability of the required resources'. After completing the resource comparison operation, the PC unit generates Deployment Descriptor (DD) containing the deployment details; and forwards it to the Deployer module to reserves the resources in CSP.

Monitoring Engine (ME) fetches the metrics data according to mentioned parameter in RD through Agent and compares fetched matrices data stored in Agreed Service Level Agreement (SLA) Repository. This comparison mechanism runs repeatedly after certain time gap mentioned by the Broker. The ME unit instructs PC unit whether the resources' are underutilized or overloaded.

In this context the following subsections describe different modules of proposed Architecture. Subsection 2.1 describes the functionality of Broker Interface, followed by working principle of Policy Comparator in subsection 2.2. Subsection 2.3 describes the working principle of Deployer module, followed by different monitoring activities by Proposed Monitoring Engine in subsection 2.4.

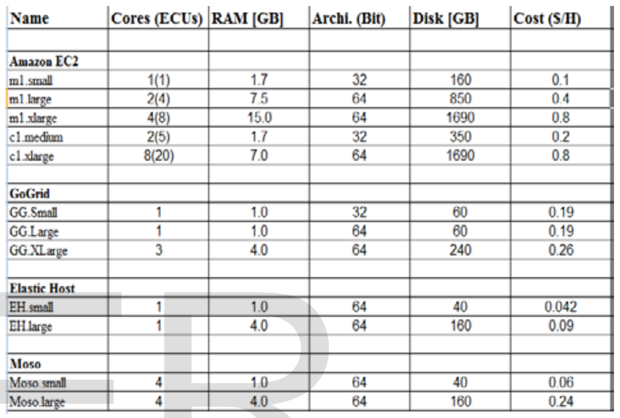
2.1 Broker Interface (BI)

BI enables users to place requirement via this interface. This is an interface of proposed Cloud Service Broker (CSB) to provide brokering functions between different CSP(s) and Cloud Service users. The CSB Interface provides the ability to access and control an appropriate Cloud resource to satisfy specific criteria. Another purpose of BI is to handle login or registration of user and generate a unique ID for user

2.2 Policy Comparator (PC)

Policy comparator is notified in two different ways. Firstly it gets some new request submitted from client interface (Broker Interface) and for deployed VMs, monitoring engine collects data and decide whether scaling is required and notifies PC unit.

The work of PC has two fold. First, PC unit has the information about the CSP's resource characteristics for the instance types offered by the registered CSP. This thesis paper only considers, IaaS resources offered by the four CSPs like, Amazon EC2, GoGrid, Elastic Host and Moso, the resource characteristics are described in fig 2. The implemented PC functionality of the broker is extensible enough to permit the easy integration of custom resource matching policies.



Name	Cores (ECUs)	RAM [GB]	Archi. (Bit)	Disk [GB]	Cost (S/H)
Amazon EC2					
m1.small	1(1)	1.7	32	160	0.1
m1.large	2(4)	7.5	64	850	0.4
m1.xlarge	4(8)	15.0	64	1690	0.8
c1.medium	2(5)	1.7	32	350	0.2
c1.xlarge	8(20)	7.0	64	1690	0.8
GoGrid					
GG.Small	1	1.0	32	60	0.19
GG.Large	1	1.0	64	60	0.19
GG.XLarge	3	4.0	64	240	0.26
Elastic Host					
EH.small	1	1.0	64	40	0.042
EH.large	1	4.0	64	160	0.09
Moso					
Moso.small	4	1.0	64	40	0.06
Moso.large	4	4.0	64	160	0.24

Fig. 2. Resource Characteristics.

Finding the best suitable Provider the Policy Comparator unit instructs Deployer to deploy the job on that particular CSP. PC specifies deployment using XML document. XML document describes a deployment consisting of several nodes, which correspond to VMs.

In second case ME unit informs the PC unit about the action need to perform to maintain the SLA after comparing the metrics data saved in SLA Repository and data fetched from the presently running VMs. If ME unit sends AckMsg to the PC unit then it instruct the Deployer not to engage in any operation, but if the ME sends any Action to the PC then it will engage in operation according to the nature of the action. Operation can be broadly classified in two types one is scale up and another is scale down. If the action is regarding the overload of resource then the ME instructs the PC to select new VM to maintain the SLA. So the PC unit checks the available best suitable VM from the Database and compares the requirements according to need and informs Deployer to deploy that particular VM. The resource characteristics of the VMs are stored in database as shown in fig 2. After deploying the VM, it migrate the overloaded job to the newly deployed VM. If the action is regarding the under utilization of resource and multiple VMs are running then need to scale down to reduce the cost. So PC instructs to the Deployer to copy the job from underutilized VMs to another running VMs.

2.3 Deployer

The tasks of Deployer are deploying the VM(s) in specified Cloud Environment, installing Monitoring Agent in deployed VM(s) and migration of VM(s) if needed.

Deployer receives the DD document regarding Provider information from the PC unit and it deploys the VM(s). After deploying the VMs it installed Agents on that deployed VM(s).

Agents have more capabilities and power than Agent less monitoring solutions and allow IT administrator to have more control of their monitoring configuration. In this proposed model Agent is installed in VMs to monitor VM's resource usage. The Agent's monitoring services include operating system metrics, service state, process state, file system usage etc.

Another task of Deployer module is to migrate and scale VM to fulfill user's change requirement. If a VM is running in one Cloud Environment and it can be migrated to another Cloud Environment; Deployer performs the necessary steps.

2.4 Monitoring Engine

This component is solely responsible for interacting with the Monitoring Agent installed in the deployed VMs to monitor continuously and manages the resource to closer look over the SLA metrics data by comparing collected data and the stored SLA Repository data. The work of ME has two folds.

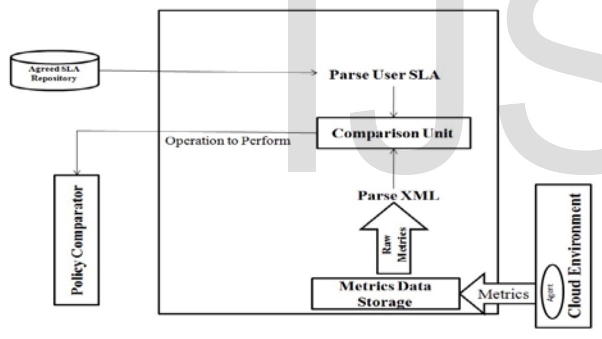


Fig. 3. Comparison Procedure.

First, ME Updates the resource status metrics including utilization, memory, availability, and other metrics mentioned in the SLA based on feedback given by the Agent component.

Secondly, it is responsible for comparing the SLA metrics based on the fetched monitoring data from the Agent. The comparison method is shown in the figure 3.

The working mechanism of SLAC is to compare the SLA metrics based on the fetched monitoring data from the Monitoring Engine. After agreeing on SLA terms the SLA will store on the Agreed SLA Repository. If any changes in SLA metrics are required then it will be negotiated by the SLAN and updated SLA will be stored on the Agreed SLA Repository. The internal working of the SLA Coordinator is shown in the fig 2.Once the ME got the updated resource status metrics by the Agents; it loads the service SLA from the agreed SLA repository. IaaS resource metrics from Agents and the metrics mentioned in SLA are parsed and send to the comparison unit. The Comparisons Unit's knowledge about the managed system is represented as a number of rules, such as:

- Facts: SLO violation occurred, resource X below threshold.
- Action: Request increase of resource X

This action will be forwarded to the PC unit whose work is to deploy the job and to manage the VM resources.

3 Operational Flow

The Operation Flow addresses the way of communication and interaction among different module to ensure effectiveness and efficiency step by step.

The following subsections describe Operational flow among different modules. Subsection 3.1 describes the Basic Operational flow between the user, Broker and the CSP, followed by VM Deployment flow in subsection 3.2. Subsection 3.3 describes the VM Monitoring flow.

3.1 Basic Operational Flow

This operational flow shows the basic operation performed between the user, Broker and CSP as shown in Fig 4.

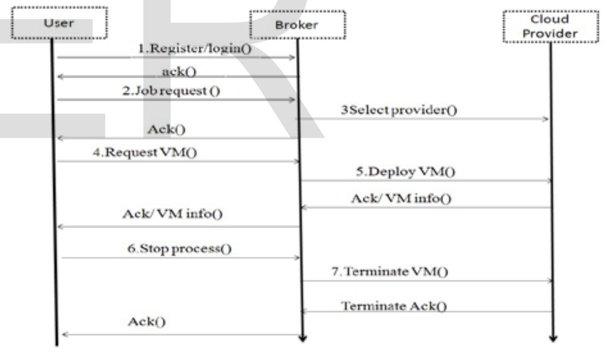


Fig. 4. Basic Operational Flow.

The basic flow functions are being described below.

1. A registered user logs in to the Brokers portal and after successful login the user gets access to the integrated CSP by Broker.
2. The user submits the job request to the Broker with some predefine SLA parameters like location awareness, memory, cost, CPU speed, availability etc.
3. According to the mentioned SLA parameter or metrics the broker compare the available service based on some policy. Acknowledgement report is sent to the user after selecting the Provider or user is notified by unavailability of proper services.
4. The user request to Broker interface to deploy the se-

lected VMs and the job requested by him.

5. The Broke deploys the selected VM(s) and job(s), requested by the user. The Provider sends back the AckMsg message to the Broker after VM deployment and the same message is forwarded to the user by the Broker.
6. User sends a stop message to stop the VMs after completion of job to the Broker.
7. The Broker sends VM termination message to the corresponding CSP. CSP returns VM termination acknowledgement to the Broker. Broker sends acknowledgement to the user.

3.2 VM Deployment Flow

Fig 5 describes VM deployment procedure.

1. The user requests for SLA template from Policy Comparator.
2. Policy Comparator forward this query to the CSP for SLA template, provide by the various CSPs and sends it back to the user.

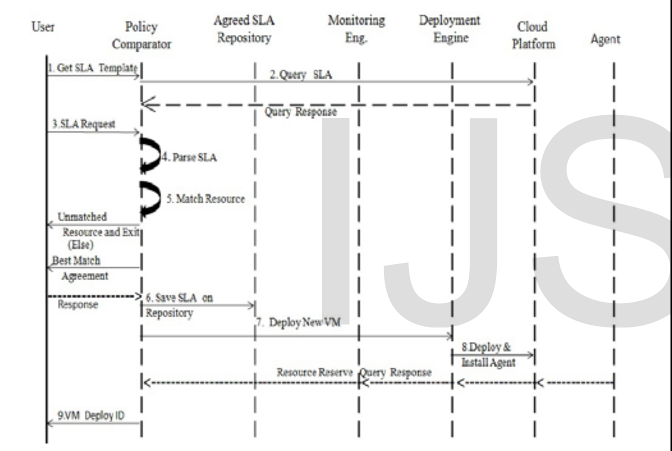


Fig. 5. VM Deployment Flow.

3. User submits RD to the Policy Comparator.
4. The Policy Comparator parses the SLA definition.
5. The Policy Comparator finds the best suitable CSP by matching the CSP's resource properties with the user's service requirements. The user gets a response with respect to the request from the Policy Comparator with the respective matching results. If none of the providers can be matched, the mentioned steps may be repeated for renegotiation or may exit else an agreement can be established with the matched provider.
6. Policy Comparator stores the negotiated RD which is termed as SLA and it is stored on repository for future purpose.
7. If negotiation is established then PC unit instruct the Deployment Engine to deploy the new VM(s).
8. Then Deployment Engine install Agents to get monitoring data of that VMs. Agent installment response sends to the Deployment Engine, Monitoring Engine, and Policy Comparator.
9. VM deployment ID is send back to the user.

3.3 VM Monitoring Flow

Fig 6 describes the VM monitoring flow.

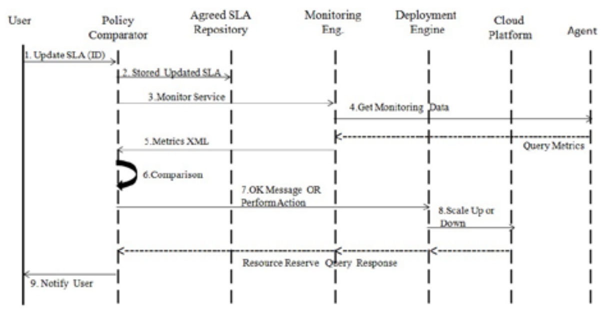


Fig. 6. VM Monitoring Flow.

1. At the time of job execution if user wants to update in SLA metrics then it inform to the Policy Comparator.
2. The modified SLA is stored in SLA Repository.
3. Policy Comparator request to Monitoring Engine for monitoring the VMs.
4. Monitoring Engine request the Agent to get the mentoring metric data from the VMs. The Agent returns the metrics data.
5. The Policy Comparator prepares an xml document of gathered metrics data and stores that data.
6. Policy Comparator has a comparison unit which compares the monitor metrics data with the updated SLA metrics data.
7. If the monitoring resource does not violate the SLA and it maintains the QoS then it sends an AckMsg to the Deployment Engine and if the monitoring metrics does not match with the SLA metrics then it request Deployment Engine to deploy new VMs to scale up or scale down.
8. Deployment Engine informs Cloud Provider to scale up or scale down.
9. After scale up or scale down the Policy Comparator informs to the user about the action.

4 CONCLUSION

The Cloud service Broker has been designed to manage and monitor Virtual Machine(s) in Cloud Environment. The user needs to choose a template according to need. User should provide the required / mandatory information to fill up the template before submitting the request. In this proposed model the resource management and monitoring is the task of Broker. The broker automates system deployment and decides when resource should be migrated or scaled.

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] 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.

[3] Christopher Clark, Keir Fraser, Steven Hand, Jakob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt and Andrew Warfield , "Live Migration of Virtual Machines" , Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI '05) Volume 2, Pages 273-286, May 2005, Boston, MA, NSDI'05.

[4] Erik Elmroth, Lars Larsson, "Interfaces for Placement, Migration, and Monitoring of Virtual Machines in Federated Clouds", Grid and Cooperative Computing,. GCC '09. Eighth International Conference. Page(s): 253 – 260, ISBN: 978-0-7695-3766-5, DOI: 10.1109/GCC.2009.36.

[5] Rafael Moreno-Vozmediano, Ruben S. Montero, Ignacio M. Llorente, "Multi-Cloud Deployment of Computing Clusters for Loosely-Coupled MTC Applications", Draft for IEEE TPDS (Special Issue on Many-Task Computing), July 2010.

[6] K.Senthil, "Performance Analysis of multi-cloud Deployment in Many task Applications", IJERT ISSN: 2278-0181, Vol. 1 Issue 5, July - 2012.

[7] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauery, Ian Pratt, Andrew Wareld, "Xen and the Art of Virtualization". In Proceedings of the nineteenth ACM symposium on Operating System Principle (SOSP 19), pages 164-177, ACM press, 2003.