Autonomic Brokering Agent for Multi-Cloud Environment Management

Kodirov Nodir, Jaegi Lee, TaeSang Choi

Electronics and Telecommunications Research Institute (ETRI)

Yuseong-gu, Daejeon, 305-700, South Korea

{nodir, jkilee, choits} @etri.re.kr

Abstract

This paper introduces Autonomic Brokering Agent for the management of a heterogeneous cloud environment, which is composed of several cloud solutions. Although public cloud solutions are targeting to provide common API for their cloud resource management, there is still a huge gap between levels of manageability for each cloud solution. Autonomic brokering agent tries to resolve this issue from bottom-up approach, where cloud operational log-files are used as a primary source of information for building cloud management framework.

1. Introduction

Nowadays, there are many cloud solutions for providing infrastructure as a service. Although all cloud solutions rely on virtualization technique, each one has advantages and disadvantages over the other. End users are free to choose cloud solution based on their needs. Sometimes, user wants to use several cloud solutions together (also known as cloud federation, cloud bursting, or multi-cloud). This is the place where the need for multi-cloud resource management issue arises. Autonomic brokering agent (ABA) is being developed to provide fine monitoring and control over cloud resources composed of several heterogeneous cloud solutions.

2. Single API for different clouds

There are undergoing standardization efforts to create unified API for different cloud solutions. Cloud Data Management Interface (CDMI) by SNIA and Open Cloud Computing (OCCI) by Open Grid Forum.

There is another mainstream API approach for providing management of multi-cloud environment. They are Apache *Libcloud* and Apache *DeltaCloud*. Libcloud is Python library for accessing various cloud resources and managing them via single API. Deltacloud has slightly different approach to achieve the same (Figure 1).

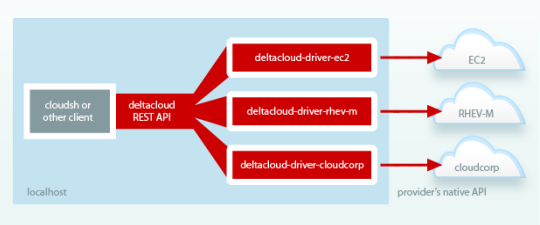

Figure 1. Deltacloud API illustration

An advantage of Deltacloud is on its independency from specific programming language. Deltacloud API can be called from any client application, which is able to make REST API calls. This makes Deltacloud very useful tool for cloud management from portable devices, such as smart phones and smart tablets. There are already few basic implementations are available in this area.

3. Cloud management platforms

Cloud includes cloud management utility for virtual machine (VM) provisioning, monitoring and cloud configuration. Typical example of small-scale private cloud management platform can be VMware vCenter Server. Figure 2 below illustrates centralized management capability of vCenter for VMs deployed on vSphere.

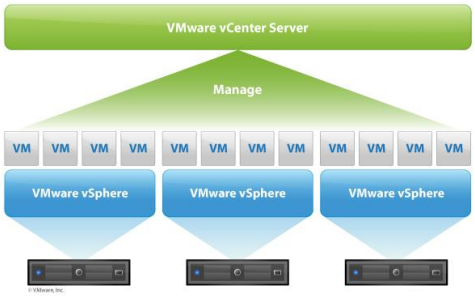

Figure 2. VMware vCenter Server

Larger scale cloud management is provided through Internet. User manages cloud by accessing web page. For example, Amazon EC2 customer uses Amazon Web Services to manage cloud environment. User logins into the EC2 portal and accesses VM management features for his environment. One of the leading solutions to provide multi-cloud management through single dashboard is done by RightScale [7].

RightScale provides single dashboard for underlying heterogeneous cloud environment. As it is depicted in the Figure 3, *Multi-Cloud Engine* is used to represent a heterogeneous cloud environment as a homogeneous management unit. It provides the same API for *Automation Engine* and other high level modules of the architecture.

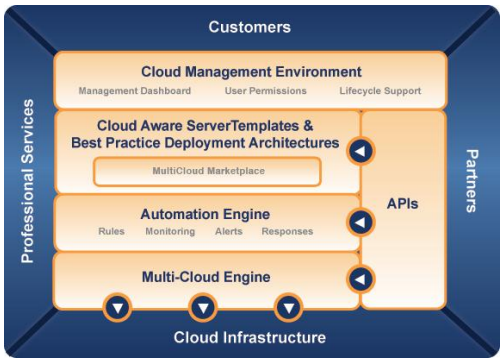

Figure 3. RightScale Cloud Management Platform

It does make a great sense to go on this direction to standardize cloud APIs and make applications manage cloud resources from the same API, but we think it to be a slow process and not going to happen sometime near future. Thus, we propose a shortcut solution with bottom-up approach for management of the resources deployed across different cloud environments. This would be the main focus of the next chapter.

4. Autonomic brokering agent

After installation, development and extensive maintenance of several cloud environments (such as Eucalyptus, OpenNebula and OpenStack) for research purposes, we examined such problems from the lower level and tried to provide bottom-up approach. Essentially, we looked at the log files generated by each cloud operating system during its normal operation. Gathering huge amount of log data, it was possible to see presence of essential information for the cloud management. It is not our intention to say that data gathered from log files are sufficient by itself for cloud management. We say data from log files can be highly valuable if it is used as supplementary to the data already provided by cloud management tool. Thus, it would be much easier to detect root causes of operational failures and monitor cloud environment in more detail. There are two main steps for brokering agent, Analyze and Normalize as illustrated in the figure 4.

Main purpose of the normalization step is to collect log data from different cloud sources and extract *useful* information. Log entries made by clouds are different in syntax and semantics. However, it was observed them to have common pattern. All log entries start with date/time information, indicating the time it was made. It is followed by the type of the log entry, which is usually specified as INFO, DEBUG or ERROR. Lastly, more detailed description is provided for particular type of log entry. Type value is core part of information which ABA is interested in. Logs having INFO or DEBUG type are categorized as monitoring information. ERROR logs are categorized as fault management data. The job of DoLogStandard module is output log entries on commonly agreed format, such as XML.

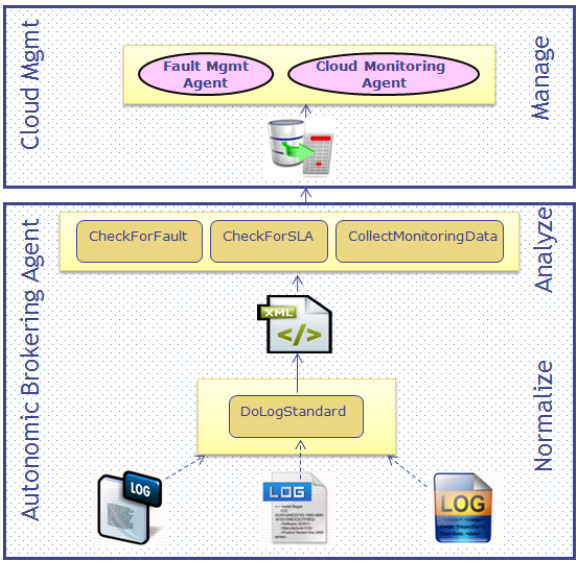

Figure 4. Autonomic Brokering Agent architecture

The second step is called analyze. This is the main logic execution part of the ABA. Reading XML content (which is now generic for all cloud environments), firstly it is checked for fault (ERROR type logs). If this is the case, it is directly reported to high level management instance, illustrated as Fault management agent (we assume it to be within cloud management framework). Secondly normalized data is checked for violation of any service level agreement (SLA). If none of above is the case, information is simply passed to cloud management framework as monitoring data (to cloud monitoring agent). Note that cloud management is not part of ABA. It is vice versa, ABA can be used as part of cloud management framework to provide additional features. As depicted in the figure 4, ABA output could be recorded into the DB as well, which makes output information useful for future use (e.g., root cause analysis).

5. Conclusion

This paper provided only introductory information for Autonomic Brokering Agent for multi-cloud environment. There is ongoing implementation works to build the full software stack and integrate it into the cloud management framework. Future works will include actual data gathered from running it on multi-cloud environment.

6. Acknowledgement

This research was supported in part by KCC (Korea Communications Commission), Korea, under the “Novel Study on Highly Manageable Network and Service Architecture for New Generation” support program supervised by the KCA (Korea Communications Agency) (KCA-2011-10921-05003).