Demonstration of Runtime Model Based Management of Diverse Cloud Resources

Xiaodong Zhang[1,2], Xing Chen[1,2], Ying Zhang[*1,2], Yihan Wu[1,2], Gang Huang[1,2], Qiang Lin[3]

1 Key Laboratory of High Confidence Software Technologies (Ministry of Education)
2 School of Electronics Engineering and Computer Science, Peking University, Beijing, China
3 Information Center of Guangdong Power Grid Corporation, China
{zhangxd10, chenxing08, zhangying06, wuyh10}@sei.pku.edu.cn,
hg@pku.edu.cn, linqiang@gdxx.csg.cn

Abstract. Due to the diversity of resources and different personalized management requirements, Cloud management is faced with great challenges in complexity and difficulty. For constructing a management system to satisfy a specific requirement, a redevelopment solution based on existing system is usually more practicable than developing the system from scratch. However, the difficulty and workload of redevelopment are also very high. This paper demonstrates Jade Cloud, which is a runtime model based Cloud management system. Based on the runtime models of OpenStack and Hyperic, Jade Cloud can manage both the infrastructure and software resources in a unified manner.
Demo link: https://www.youtube.com/watch?v=o2SiTGL689Q

Keywords: runtime model, Cloud management, diverse Cloud resources

1 Introduction

With the rapid development of Cloud computing, it brings unprecedented challenges to management of diverse Cloud resources, which mainly include two aspects. First, there are many different kinds of hardware and software resources in Cloud which have to be managed well. Second, there are different personalized management requirements consisting of specific scenarios and appropriate management styles.

To satisfy the personalized management requirements, Cloud administrators usually conduct redevelopment based on the existing management systems other than develop a new system from scratch [1]. However, the difficulty and workload of redevelopment are also very high, which have to understand the existing systems first and invoke and organize many kinds of heterogeneous management interfaces of Cloud resources and third-party services to satisfy a given management requirement.

To solve the above issues, we present a runtime model based approach to managing diverse Cloud resources and have published a paper in Models 2013 [1]. Through runtime model construction of Cloud resources, getting the composite model through model merge and model transformation from the composite model to customized

models, Cloud management tasks can be carried out through executing operating programs on the customized model.

To further demonstrate the above approach, in this paper, we present Jade Cloud - a runtime model based Cloud platform managing diverse Cloud resources. Jade Cloud is constructed based on OpenStack [2] and Hyperic [3]. OpenStack is an open source product which is aimed to manage Cloud infrastructure. Hyperic is a product which manages different kinds of software. However, to the best of our knowledge, there is currently no product which can manage both of the infrastructure and software resources in a unified manner. Based on the runtime model, Jade Cloud implements the unified management of both infrastructure and software resources.

2 Overview of Jade Cloud

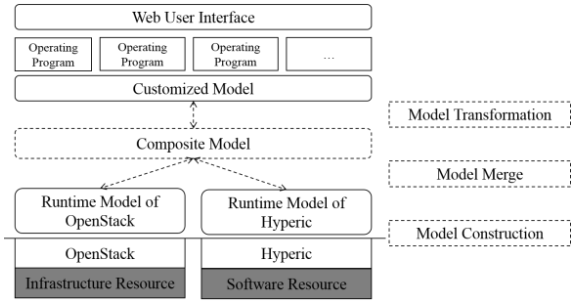

Fig. 1. Overview of Jade Cloud

Figure 1 shows an overview of Jade Cloud. Based on the runtime models of OpenStack and Hyperic, it implements the unified management of both resources through runtime model construction, model merge and model transformation.

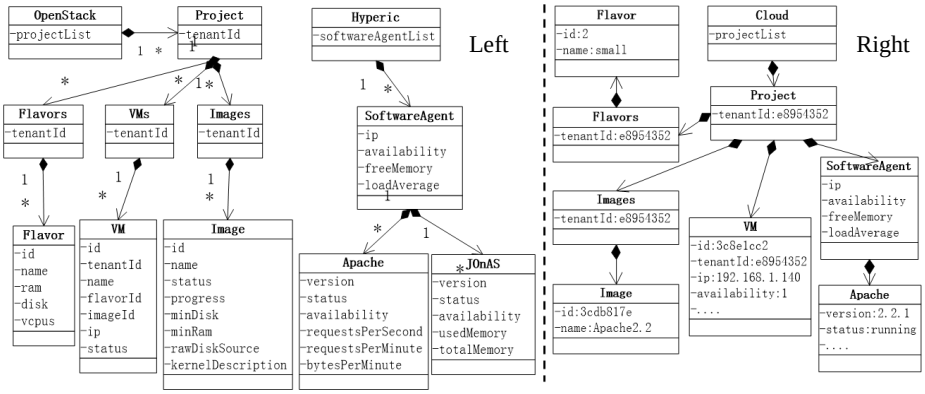

Fig. 2. Meta-models of OpenStack and Hyperic (left), a sample of the composite model (right)

We only define the architecture-based meta-model specifying the structure of the running system (OpenStack or Hyperic) and the access model specifying how to ma-

nipulate the system's elements, then the construction of runtime models of OpenStack and Hyperic are completed with the help of SM@RT [4]. Figure 2 shows the architecture-based meta-models of OpenStack and Hyperic. We then construct the composite model of these two models through model merge.The right part in Figure 2 shows a sample of the composite runtime model. We use the customized model to better meet the personalized management requirements, which is constructed through model transformation from the composite model. In Jade Cloud, The virtual machine and the software deployed on it are regarded as an appliance [5], which is the basic managed unit. Several appliances compose a project. The resources of the infrastructure are divided into many projects. Figure 3 shows the main elements of the customized model of Jade Cloud. All the details about the above approach can be found in [1].

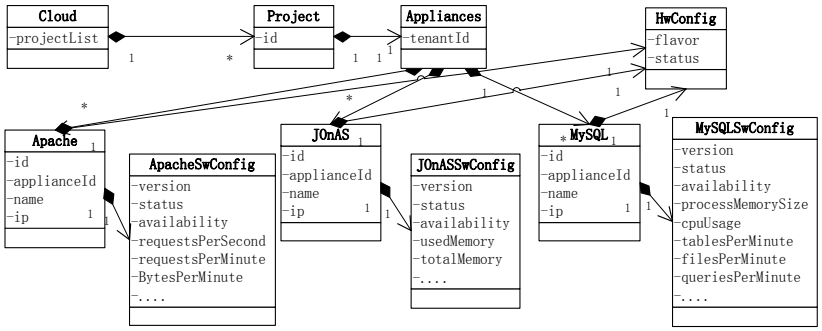

Fig. 3. The main elements of the customized model of Jade Cloud

3 Demonstrations

We deploy Jade Cloud on 11 physical servers and it can support about 150 appliances. Figure 5 shows a screenshot of the user interface of Jade Cloud. Administrators can manage the resources through updating the system deployment diagram on the web interface. In the diagram, the elements of the appliance stand for virtual machines with software products, which compose the runtime model of the Jade Cloud. Taking the appliance named "APACHE" as an example, we can get and set the hardware information (freeMemory, LoadAverage_5min etc.) and the software information (Bytes_Served_per_Minute, Requests_Served_per_Second etc.). Jade Cloud reuses and reorganizes the management interfaces of OpenStack Compute and Hyperic, instead of modifying the two systems. The runtime model based management system also benefits from many model-centric analyzing or planning methods and mechanisms. We have implemented three kinds of model-based Cloud services on Jade Cloud, which are Micro-reboot, Reliability Analyze and Load Analyze. The Micro-reboot service can reboot an appliance automatically when the appliance is shutdown unexpectedly. The Reliability Analyze service implements a SBRA-based algorithm to find out the key components which have crucial influence to the system. And the Load Analyze service can find the appliance with the highest workload.

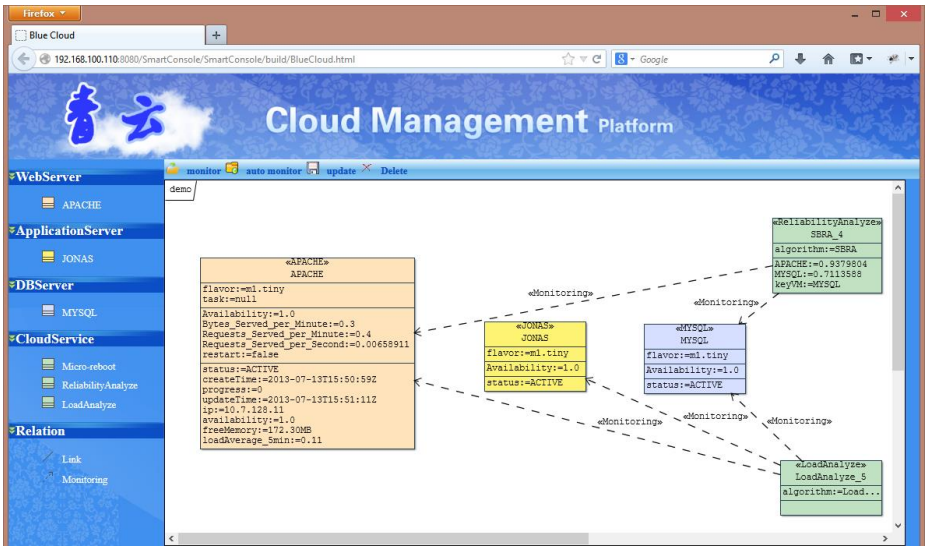

Fig. 4. User interface of Jade Cloud

With the help of SM@RT [1][4], the model construction, model merge, and model transformation are all one-off work, which only need to be done once. So the work load of the runtime model based approach is much smaller than the redevelopment. To evaluate the performance of Jade Cloud, we develop Java programs using JMX [6] management interfaces and QVT (Query/View/Transformation) [7] programs using runtime models. These two kinds of programs are used for executing five groups of management tasks respectively. The results are shown in Table 1.The execution time means the average time cost of each group of management tasks, and the data delay is the average delayed time to obtain the data.

Table 1. The performance test result of the unified managment

Management Tasks	Number of Appliances	Using Management Interfaces		Using Runtime Model	
		Execution Time (sec)	Data Delay(sec)	Execution Time(sec)	Data Delay (sec)
Create new appliances	1	22.9	-	24.1	-
	5	43.4	-	45.5	-
	10	58.7	-	62.1	-
Delete appliances	1	11.2	-	13.8	-
	5	29.7	-	30.1	-
	10	41.4	-	45.6	-
Get “usedMemory” attribute	5	1.2	0.6	0.6	30
	20	4.2	2.2	0.8	30
	100	18.6	11.4	1.6	30
Set “name” attribute	5	2.1	-	4.1	-
	20	8.3	-	11.7	-
	100	39.7	-	44.2	-
Restart Apache	5	9.1	-	12.3	-
	20	37.6	-	41.2	-
	100	192.4	-	207.3	-

For the “create”, “delete”, “set” and “restart” management tasks, the execution time of Java programs is less than the QVT ones. The main reason is that the two sets of programs are based on the same management APIs and there are some extra operations in runtime model based approach, which ensure the synchronization between the runtime model and real system. However, the difference is small and acceptable.

For the “get ‘used-memory’ attribute” management tasks, the execution time of Java programs is longer than the QVT ones, but the data delay of QVT programs is longer than the Java ones. The reasons are two folds. On one hand, the Java programs query the attributes of appliances through directly invoking the management APIs, so the execution time increases linearly with the number of the appliances and the delay is little. On the other hand, the runtime model is equivalent to the snapshot of system metrics and getting the attributes of appliances just needs a reading operation, so the execution time of the QVT programs is shorter. The runtime model is synchronized with the running system with traversing all the metrics of the running system, so the data delay increases linearly with the size of the model.

4 Conclusion and Future Work

In this paper, we demonstrated Jade Cloud, which is a runtime model based Cloud management system. Based on the runtime models of OpenStack and Hyperic, Jade Cloud can manage both the infrastructure and software resources in a unified manner without modifying existing systems. The work load and performance of Jade Cloud is also evaluated. For future work, we plan to add more model-based management functions and apply the approach in production environment.

ACKNOWLEDGMENT. This work is supported by the National High-Tech Research and Development Program of China (863) under Grant No. 2013AA01A208; the National Natural Science Foundation of China under Grant No. 61300002 and the China Postdoctoral Science Foundation under Grant No. 2013M530011

References

1. X. Zhang, X. Chen, Y. Zhang, Y. Wu, W. Yao, G. Huang, Q. Lin. Runtime model based management of Diverse Cloud Resources, accepted by MODELS 2013.
2. OpenStack. http://www.openstack.org/
3. Hyperic. http://www.hyperic.com/
4. H. Song, Y. Xiong, F. Chauvel, G. Huang and Z. Hu, et al. Generating Synchronization Engines between Running Systems and Their Model-Based Views. Models in Software Engineering, Pages 140-154, 2010.
5. Sapuntzakis, C. P., Brumley, D., Chandra, R., Zeldovich, N., Chow, J., Lam, M. S., & Rosenblum, M. (2003, October). Virtual Appliances for Deploying and Maintaining Software. In LISA (Vol. 3, pp. 181-194).
6. JMX. https://en.wikipedia.org/wiki/Java_Management_Extensions
7. Object Management Group. “Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT)”. Retrieved 9 May 2011.