Early Experience of Building a Cloud Platform for Service Oriented Software Development

Hailong Sun, Xu Wang, Chao Zhou, Zicheng Huang, Xudong Liu
School of Computer Science and Engineering
Beihang University
Beijing, China
{sunhl, wangxu, zhouchao, huangzc, liuxd} @act.buaa.edu.cn

***Abstract*—Cloud computing is commonly characterized as a three-layer architecture including IaaS, PaaS and SaaS, while service oriented approach is widely considered as a promising software development method. In this paper, we report our early experience of moving traditional service oriented software development to the cloud computing environment. Our primary goal is to provide instant development, instant deployment and instant running services for service oriented software developers. Corresponding to the three layers of cloud computing, our work includes software appliance management, app engine and online development environment. We elaborate on the design and preliminary implementation experience.**

Keywords-service oriented computing; cloud computing; software development;

I. INTRODUCTION

Traditional software development lifecycle involves a series of tasks including requirement specification, design, coding, test, deployment, etc. To fulfill these tasks, developers often need the support of various software tools that are usually purchased and installed locally. Beyond that, software testers must set up an infrastructure environment to deploy their software. In other words, software developers whose main task should be focused on building software logic have to deal with the software development environments. Therefore, it will be very meaningful if software developers can obtain a development environment on demand, which will greatly facilitate development work and improve the software productivity.

In addition, service oriented software is known to be the next generation software development approach[1], which supports building on-demand new software applications through integrating existing services. However, service oriented software development is a complex process that involves atomic service development, business process modeling, service orchestration, verification & testing, application deployment, etc. And various supporting tools and middleware are needed to build a software application using service oriented approach. As we mention early, software developers have to deal with software development environment issues themselves. The objective of this work is to relieve the burden of setting up and maintaining a development environment for service oriented software developers.

In recent years, cloud computing [2] has been widely considered by both industry and academic communities as a promising computing paradigm. The essence of cloud computing is to transparently deliver services over the Internet. These services are broadly classified into three categories including IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service). With the support of cloud services, various types of users do not need to take into account the technical issues behind these services. In this work, we are concerned with how to leverage cloud computing technologies to simplify the software development. Specifically, we are working on building a cloud platform for service oriented software development. In fact, we also have noticed that there is similar work in this direction. For instance, IBM launches its Smart Business Development and Test Cloud [3] to implement a flexible and cost-efficient, cloud-based development and testing environment. Hajjat et al. [4] present their work on the challenges of migrating traditional enterprise application to cloud computing environment.

We have already developed a suite of tools and middleware for service oriented software development, which is named SOARWare[5]. In this paper, we present our early experience of moving SOARWare to clouding environment so as to build a cloud platform for service oriented software development. With this work, we hope to achieve the goal of instant development, instant deployment and instant running of service oriented applications.

The remainder of this paper is organized as follows. Section II provides a brief introduction to our previous work on service oriented software development, which is the working basis of this work. Then we analyze several motivational problems. In Section III, we describe the architecture design and early implementation experience. Finally, Section IV concludes this work.

II. PRELIMINARY WORK AND PROBLEM ANALYSIS

A. SOARWare-A Service Oriented Software Production and Running Environment

SOARWare is our previous work on service oriented software development based on Web service technologies. It aims at providing an environment that can support the full lifecycle of software development through service oriented computing. To achieve this goal, four important issues are identified as follows.

978-1-4244-8396-9/10/$26.00 ©2010 IEEE.

(1) Service resource management. As more and more services are developed and published over both Internet and Intranet, there must be some mechanism to manage this resource so as to provide support for efficient service discovery and consuming. Service resource management mainly includes service crawling, organization, mining and searching.
(2) Design and development. Business process is the core concept in service oriented software development. The business process centered development involved a series of activities including modeling, service orchestration, verification and testing, which requires various people to participate in. Thus how to provide support for the efficient corporation is an important issue.
(3) Runtime management. After the application is developed, it will be deployed to runtime environment. The service oriented application is constructed through composing various services. The service container and composite service engine are designed to support the running of atomic and composite services respectively. Additionally, other issues including the management of service containers and composite service engines, message routing and adaptive service selective are all the necessary functionality of runtime management.
(4) Unified access platform. Similar to clouding computing, it is very important to provide a unified access platform for service consumers, service providers and service developers.

SOARWare consists of three major components including SOARBase, SOSPL (Service Oriented Software Production Line) and SOARBus (Service Running Bus).

First, SOARBase is a fundamental component to address the service manageability issue. The role of SOARBase in SOARWare is similar to the role of DBMS in traditional software systems. Not only service providers can publish their services to SOARBase, but also SOARBase itself can search existing online service registries or make use of web search engine to actively collect Web services over the Internet. Furthermore, SOARBase is responsible for managing the collected service information using data mining technologies. In all, the objective of SOARBase is to provide support for discovering needed services as efficiently as possible.

Second, since SOARBase provides the source of services for producing software, there still needs an effective mechanism to manage the software production process. This mechanism is provided by SOSPL (Service Oriented Software Production Line). The business process is the kernel concept in software production with SOSPL. Software production is composed of a series activities centering on business process, including business process modeling, service orchestration, verification, testing and deployment. On the one hand, SOSPL provides tools to support various development activities; on the other hand, SOSPL must support the automatic control of the development process.

Third, after service oriented software is produced by SOSPL, it is deployed onto runtime environment that consists of SOARBus (our implementation of ESB), service containers, composite service engines and service evolution management modules.

B. Problem Analysis

Although SOARWare provides a nearly complete solution to service oriented software development, there exist several problems in practice.

First, developers need to obtain (e.g. downloading) and install various tools in their development machines. Since service oriented software development involves many activities depending on the complexity of application requirements, the maintaining of software tools is not a trivial task. Especially, in the case of tool update, there can be incompatible issues between different tools.

Second, to test the developed software applications, developers must set up a runtime environment. As service oriented applications are highly distributed and dynamic, setting up an effective test environment is a great challenge. The developers not only need to find appropriate infrastructure resources, but also need to consider the deployment policies of both atomic services and composite services.

For the first problem, we need an online development environment. With such an environment, developers no longer need to obtain, install and maintain development tools themselves. All they need to do is to start Web browser and develop application logic with corresponding tools. The second problem fits well with cloud computing paradigm. We will elaborate our solution in next section.

III. SYSTEM ARCHITECTURE AND KEY TECHNOLOGIES

A. Service Oriented Software Development Cloud

According to the analysis in Section II, the primary goal is to provide an online development environment and dynamic hosting environment for service oriented software. The benefits that developers will obtain can be concluded as follows:

- *Instant development*. Service oriented software developers do not need to obtain development tools and assemble a development environment themselves. Instead, all the tools are provided online and can be used in a Web browser. Thus development can be started instantly.
- *Instant deployment*. After a software application is developed using online development environment, developers can start the deployment process instantly without knowing where and how to deploy.
- *Instant running*. Once an application is deployed successfully, developers can start running and monitoring the application using online tools for testing purpose.

To provide the aforementioned instant services for developers, as illustrated in Figure 1, our service oriented software development cloud adopts a three-layer system architecture corresponding to the three types of cloud services.

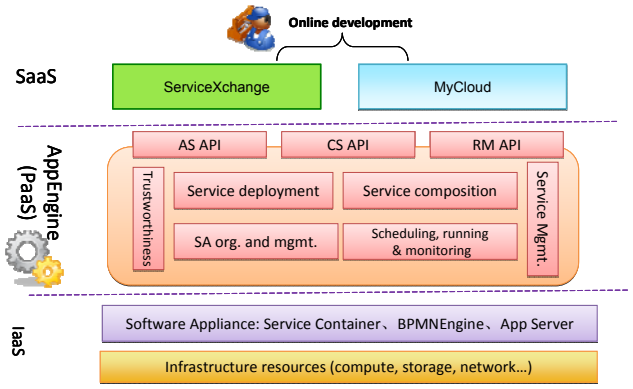

Figure 1. Architecture of Service Oriented Software Development Cloud

The IaaS layer is mainly responsible for providing infrastructure resources like computers, storage and networks. These underlying resources are fundamental resources to establish runtime environment for service oriented applications. There have been several IaaS providers, e.g. Amazon EC2 [6]. On the basis of the underlying resources, we also need the support of middleware including service container, composite service engine, service bus and application server. VM (virtual machine) technologies are leveraged to meet the dynamic resource requirements. Therefore, VM images for runtime middleware that are called *software appliance* should be generated beforehand.

The PaaS layer is the kernel layer of the service oriented software development cloud. This layer provides an App Engine for hosting service oriented software applications. The role of this layer is similar to Google App Engine [7]. However, GAE only provides hosting environment for Java and Python programs, and it does not support service oriented applications. Basically, after a software application is developed, the developer can send a deployment requirement to App Engine. The latter will dynamically set up a runtime environment and deploy the received software application. Then the developer can test, run and monitor the deployed application. In the whole process, the developers do not need to consider the technical details.

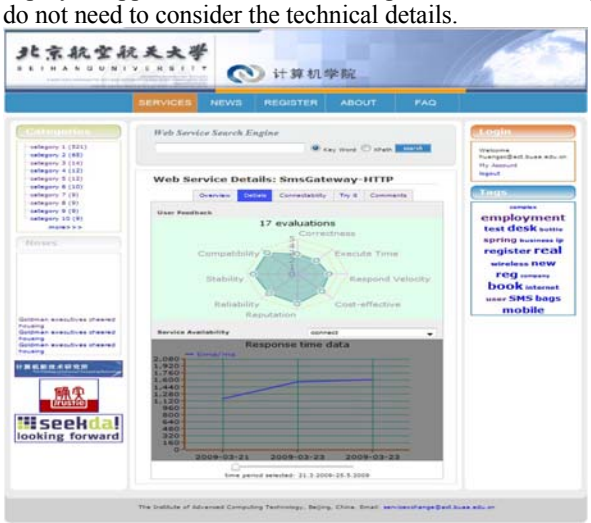

Figure 2. Snapshot of ServiceXchange

The SaaS layer aims at providing an online development environment for developers so as to support developers to focus on application logic development. Once a software application is developed, the developer can ask the App Engine for a hosting environment to test and run the produced application.

B. Online Service Oriented Software Development Environment

The central point of service oriented software development is that application can be built by composing existing services. Following the technical approach adopted by SOARWare, Web services are chosen to be the building blocks of service oriented software applications. We design ServiceXchange [8] to support the sharing of Web services. ServiceXchange is developed on the basis of SOARBase. The service management functionalities on the backend are moved to App Engine, while ServiceXchange is focusing on providing an online interface for users. With ServiceXchange, users can publish, find, subscribe a service, and they can also submit their feedbacks of certain services in terms of functional and non-functional attributes.

MyCloud is a personal development environment for an individual developer. Each registered developer will obtain a virtual workspace in which various development tasks can be performed. First, a developer can manage the subscription of atomic Web services. For example, a subscribed Web service can be tested or unsubscribed. Second, online development tools are provided for each developer. Currently, we have a prototype implementation of BPIDE_Lite (shown in Figure 3), a Web version of BPIDE (Business Process Integrated Development Environment). BPIDE_lite provides tool support for business process modeling using BPMN (Business Process Modeling Notation) specification and Web service choreography. BPIDE_Lite is implemented using Adobe Flex framework, so it can be accessed with most of Web browsers that have Adobe flash plug-in installed. Third, each developer is allocated a virtual service container and a virtual composite service engine respectively. The former provides a view of a developer's atomic Web services, while the latter is a view of composite Web services. The reason why we use "virtual" is that a developer has no knowledge where an atomic or composite service is located physically.

C. App Engine: Dynamic Hosting Environment for Service Oriented Software

Unlike traditional software, service oriented software runs in distributed network environment and cannot be tested on local computers. Therefore developers need a runtime environment to test the developed software applications. As a whole, App Engine provides a dynamic hosting environment for service oriented software, which is the key to implement *instant deployment and running* of service oriented software. Basically, App Engine is responsible for managing the runtime environment on behalf of developers so that developers can focus on application logic development.

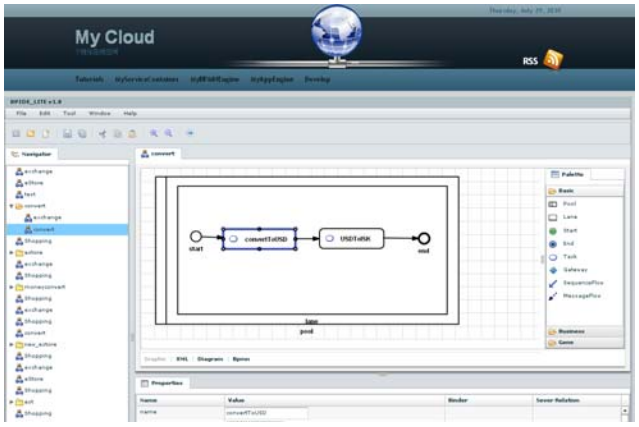

Figure 3. Snapshot of BPIDE_Lite

First, App Engine is responsible for various backend management tasks including service management, trustworthiness management and SA (software appliance) management. Service management is the backend of ServiceXchange, which involves with automatic Web service crawling, clustering, classifying, service relation mining and tagging. Because service implementation is transparent to users and the runtime environment is often over the Internet, trustworthiness guarantee is an important challenge for service oriented software. We try to deal with this issue from two aspects. On the one hand, we use the feedbacks submitted by users from ServiceXchange to compute the trustworthiness of Web services, which provides a criterion to select candidate component services; on the other hand, we are working to adopt sand-box mechanism in service container and composite service engine to prevent the malicious behaviors of distrusted services. Finally, SA management is responsible for the organization and locating of various software appliances.

Second, App Engine accepts software in the form of either atomic or composite Web services and deploys the received services to service container or composite service engine dynamically. The deployment module first finds an available software appliance from SA management module, then invokes the deployment interface of the obtained SA to perform deployment task.

Third, to deal with the concurrent requests for running the same application, App Engine needs to provide certain scheduling mechanisms. The mature scheduling strategies like FCFS can be applied accordingly.

For most of situation, developers invoke App Engine through online development environment, even without knowing the existence of AppEnigne. In addition, other developers using SOARWare's SOSPL can also invoke App Engine through APIs including AS API (Atomic Service API), CS API (Composite Service) and RM API (Running and Monitoring API) to deploy and run their applications.

D. Dynamic Software Appliance Provisioning

One of the important characteristics of cloud computing is multi-tenant access. Therefore, resource competition must be considered in building the cloud for service oriented software development because there may be multi developers who are using the platform service concurrently. We assume the underlying infrastructure resources can be obtained by other cloud services like Amazon. Thus we are only concerned with dynamic provisioning of software appliance.

When App Engine asks for a software appliance, a VM image containing the requested middleware image will be initiated; instead, when App Engine requests for shutting down an idle software appliance, the corresponding VM image will be stopped and the underlying resources will be released at the same time.

IV. CONCLUSION

Service oriented software is believed to be a promising software development approach. However, current developers whose main work should be application logic development take a lot of effort to maintain development and runtime environment. In this paper, we present an ongoing work on building a cloud platform for service oriented software development. By leveraging cloud computing paradigm, we aim at providing an online development environment and a dynamic application hosting environment for service oriented software developers. This work is on the basis of SOARWare-our previous work on service oriented software production and running environment.

In this paper, we introduce the motivation, architecture design and early implementation experience. In future, we are continuing the design and implementation of this work.

ACKNOWLEDGMENT

This work was partly supported by China 863 program under Grant No. 2007AA010301. We thank other colleagues who make contribution to this work.

REFERENCES

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," in IEEE Computer. vol. 40 (11), 2007, pp. 64-71.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. "Above the clouds: A Berkeley view of cloud computing". Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[3] IBM Smart Business Development and Test Cloud. http://www-935.ibm.com/services/us/index.wss/offering/midware/a1030965.

[4] M. Hajjat, X. Sun, Y. E. Sung,D. Maltz, S. Rao, K. Sripanidkulchai and M. Tawarmalani, "Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud". Accepted by SIGCOMM 2010.

[5] H. Sun, X. Liu, X. Li, J. Zeng, Z. Huang, "SOARWare: A Service Oriented Software Production and Running Environment," Proceedings of the 12th International Asia-Pacific Web Conference (APWeb 2010).

[6] Amazon Elastic Compute Cloud (EC2). http://aws.amazon.com/ec2/.

[7] Google App Engine, http://code.google.com/appengine/

[8] Service Exchange, http://www.servicexchange.cn.