Automatic Web Services Deployment

Ang Tan Fong, Ling Teck Chaw, Phang Keat Keong, Por Lip Yee
Department of Computer Systems and Technology
Faculty of Computer Science and Information Technology
University of Malaya, Malaysia
{angtf, tchaw, kkphang, porlip}@um.edu.my

Abstract

As web service has become the emerging paradigm, the area of web service research has received a lot of attention in recent years. Most of the web services are deployed by site administrators. As the number of request of web services increase tremendously, there is a need to replicate the services to multiple resources. Although manual deployment allow the services to be deployed safely, it is impossible to scale. To solve this problem, we propose in this paper a new architecture for automatic web services deployment. Through the experiment, we proved that automatic deployment strategy able to handle the increasing number of users' request in an effective manner.

1. Introduction

Web services technology is playing a major role in today's distributed computing. Web services can be advertised, located and used across the Internet using the standard protocol like SOAP, WSDL and UDDI. Due to its interoperability, Web services have been recognized as the next generation framework for building distributed applications over the Internet. As the number of services increase, an architecture called Service Oriented Architecture (SOA) has been proposed to organize and manage the services. SOA is a set of principles that define an architecture that is loosely coupled and comprised of service providers and service consumers that interact according to a negotiated contract or interface [1]. The primary goal of SOA is to expose application functions in a standardized way so that they can be leveraged across multiple domains.

Although SOA has enabled the merging of Grid and Web services, it provides poor support for resource matching. SOA provides a powerful framework for matching services functionalities but lack of QoS support. SOA selects the resources based on the web services functionalities without consider the non-functional properties, such as availability, reliability, accessibility, cost and accuracy. Another drawback of SOA is that, searching and mapping is done according to the registered services. If the number of requests is more than the primary resources, the request is send to the waiting queue. There is no solution to deploy the web services dynamically to the secondary resources. Further more, whenever users want to deploy a new service, they need to ask the site administrator to configure the web server and setup the web service. No auto deployment is available and it is impossible to scale.

In order to address the above problem, we propose in this paper a new architecture for automatic web services deployment. The architecture will first group all the relevant web services based on their functionalities. Then, it will select the most suitable resources according to the users' QoS constraints. If all the resources are fully utilized, the architecture will automatic deploy the services to the secondary resources. The adaptive mechanism is used for the service deployment. This strategy will reduce the number of tasks waiting in the queue and able to handle the increasing number of requests efficiently. Besides automatic deployment, the architecture allows the services to be stored in repository for future uses. Experimental results show that the automatic deployment strategy able to handle the increasing number of users' requests in an effective manner.

The remainder of this paper is organized as follows. Section 2 provides an overview of the related work. In Section 3, we discuss the deployment architecture. Section 4 details the prototype implementation and describes the experiments we have conducted. Finally, we conclude our work in Section 5.

2. Related Works

978-0-7695-3507-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CSIE.2009.417

Recently, QoS issues of Web services have obtained great interest in the research community. In [2], method to support QoS of web service in the distributed domain is proposed. The paper describes various supporting QoS attributes for Web services. By using agent based approach, the QoS information is gathered. Then, the service is selected dynamically based on the consumers' preferences and providers' capabilities. In [3], the author defines a new discovery process that combines the functionality and QoS of services. The author again uses agent to handle the QoS measurement, evaluation and QoS information exchange.

In [4], the issue of QoS driven service selection in dynamic Web service composition is investigated. The author extends OWL-S with a lightweight QoS model to better facilitate service selection process. The results show that the algorithm is time efficient, while still achieving satisfying optimal rate. Jingya Zhou et al. [5] propose an efficient algorithm in selecting the best resource according to users' performance requirements. The algorithm can select the least cost composite service while satisfying end-to-end QoS requirements and has greater success rate.

Besides QoS issues, various dynamic service deployment mechanism have been explored in the area of distributed systems. Daniel et al. [6] propose architecture for a system which allows the deployment of services in a group of computers, connected in a peer-to-peer fashion. The architecture uses publish and subscribe approach to detect events and perform self actions. Another similar approach, Snap [7] deploys web services over a Distributed Hash Table and creates replicas of a service on demand. When the demand reduces, the replicas are deleted. However, the research assumes all compute nodes are equal and able to execute any services.

In [8], the authors present architecture to enable on demand resource provisioning. The project provides a Universal Factory Service (UFS) that provides a dynamic Grid service deployment mechanism. The results show that dynamic deployment can use resource more efficiently compared to static partitioning. Other similar approaches are stated in SODA [9], OGSI.NET [10] and DistAnt [11].

Gabor Kecskemeti et al. [12] propose automatic service deployment using virtualization. The paper describes an extension to the Globus Workspace Service [13]. Service deployment with virtualization can support both the on-demand deployment and the self-healing services. The author defines an infrastructure on which an automated service deployment solution can build on. However, the virtualization middleware introduces a visible overhead in the performance of services. Other similar works in virtualization include VMPlants [14] and Virtualized Clustering project [15].

Vanish et al. [16] quantitatively compare manual, script-based, language-based, and model-based deployment solutions as a function of scale, complexity, and susceptibility to change. The research concludes that if the number of deployed systems is small or the systems' configurations rarely change, a manual solution is the most reasonable approach. However, in larger environments, the script-based solutions are well matched for large scale deployments, language-based for services of large complexity, and model-based for dynamic changes to the design.

From the findings, we propose to use agent based approach for QoS collector. The resources will be ranked based on QoS attributes and the resources are dynamically selected based on users' constraint and requirements. When the number of requests more than the registered resources, automatic service deployment will be executed based on modify version of Snap [7] that consider resources as heterogeneous nodes. As pilot phase, the research will mainly focus on script-based solutions. Most of the running task will be the parametric services. Our work primarily differs from previous researches in that we will focus on adaptive mechanism on service selection and service deployment.

3. Automatic Web Services Deployment Architecture

We propose automatic web service deployment architecture to handle the increasing number of users' request. Figure 1 depicts the architecture of the automatic web services deployment. As shown in the figure, the user first logon to the User Interface (UI). The user can submit a new service or request an existing service. If it is a new service, UI will register the service to UDDI server. After that, the user's constraints and preferences are submitted to the UDDI server as XML file according to the OASIS Standard (WS-BPEL 2.0) [17].

Apart from handling user request, the UI will monitor the completion of the service. When the service is completed, the output is stored in the UI server and the user is notified. UI support both the synchronous and asynchronous submission from user. Another feature provided by UI is the security module. The UI provides an interface for authorization and authentication. Users need to provide their identity in order to use the services.

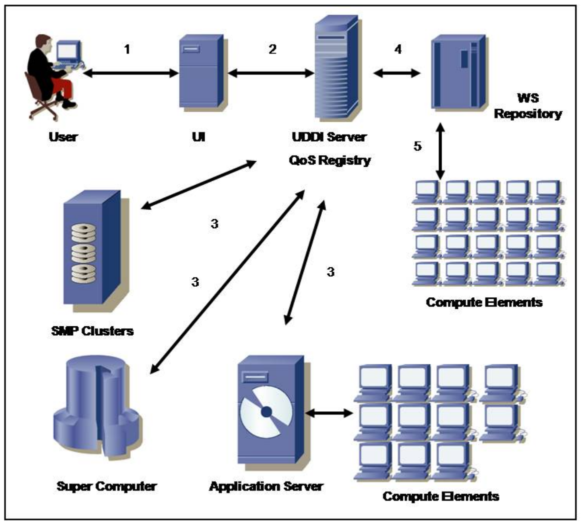

Figure 1. Automatic Web Services Deployment Architecture

The main functionality of UDDI server is to enable the publishing and sharing of information about web services. We modify the UDDI server to select the most suitable services based on not only the services functionalities but also the users' constraints and preferences. A QoS registry is developed to integrate with the UDDI server. The QoS registry contains the QoS attributes for the services and resources. The service properties will include memory and storage requirements, subtask dependency in a service, completion time and communication volumes. When a new service is registered to the UDDI server, service profiling module is executed from time to time to retrieve the service properties. Besides, analogical benchmarking is performed to measure the service performances on dynamic resources. The behavior of the services such as static, linear and exponential can be analyzed from the benchmark.

Agent is installed in all the compute resources that available in the architecture. The main feature of agent is to collect the resources performance information such as processing speed, number of core and number of concurrent request, throughput and latency, availability, reliability, accessibility, accuracy. The agent collects the information periodically from the resources, summarize it and store it in the UDDI server.

By using the information above, the UDDI server can perform the selection operation. The operation consists of three phases namely matching, numbering and ranking. During the matching process, all the services that satisfy the functionalities and users constraints and preferences are selected. Then, the value of QoS is normalized to calculate the numerical result. Finally, the ranking process will rank the resources using different weight for different QoS properties. UDDI will select the resource with the best ranking.

The Web services repository is the main component that facilitates the automatic deployment. When the waiting queue of UDDI server reaches certain threshold, the UDDI server will instruct the web service repository to perform automatic services deployment. This is happening when the web server, cluster and super computer unable to handle the increasing number of request. Without the automatic deployment, the administrator need to setup and register the web service manually. The manual process can not scale and is time consuming. By using the automatic deployment, the Web services repository able to select the requesting service and deploy it to the secondary resources. The deployment is using the manager worker model and no web server is required at each compute elements.

Apart from its main feature, Web services repository can be considered as the secondary server of the architecture. It has the latest copy of web services from UDDI. It can perform analogical benchmarking to identify the behavior of the service when the secondary compute elements are idle.

4. Implementation and Experiments

The architecture is currently implemented on .NET and J2EE environment. Therefore, it can be used for any web service technologies. Since Web services manage to solve the interoperability problem, we can easily deploy the web services on different platforms. Besides the basic library of .NET and J2EE, the UDDI .NET SDK version 2 is used to allow the integration with UDDI server. In addition, we develop a QoS registry which contains the QoS attributes for the services and the resources on top of the UDDI server. Our implementation links the QoS information with the services functionalities together.

For the automatic deployment module, no web server is required to install on the compute elements. Http.sys architecture is used as the low-level HTTP protocol stacks for the dynamic deployment on compute elements. It is a kernel-mode component that offers HTTP services to all applications on the machine. When http.sys receives a request, it can forward it directly to the correct worker process. Http.sys is capable of caching responses directly within the kernel. This improves the overall throughput and performance. Figure 2 shows the Http.sys architecture.

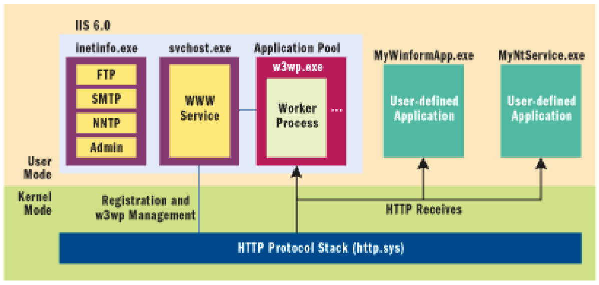

Figure 2. Http.sys Architecture

After the implementation, the architecture is tested with the parametric service. We have conducted a set of experiment using the static parametric service. Table 1 show the resources use for the experiments. The first three set of resources are the primary resources while the last two set are the secondary resources. The average execution time of each service on different resources is shown in Table 2.

Table 1. Resources

	Resource	Core	Unit	Type
1	Server with CPU 2.33GHz	8	5	Primary resources
2	Server with CPU 1.86GHz	8	5	Primary resources
3	Server with CPU 1.86GhHz	4	5	Primary resources
4	PC with CPU 2.33GHz	2	20	Secondary resources
5	PC with CPU 2.0GHz	2	20	Secondary resources

Table 2. Average Execution Time

	Resource	Core	Average Execution Time (s)
1	Server with CPU 2.33GHz	8	4.6
2	Server with CPU 1.86GHz	8	6.2
3	Server with CPU 1.86GHz	4	14.6
4	PC with CPU 2.33GHz	2	19.1
5	PC with CPU 2.0GHz	2	22.2

Figure 3 illustrate the distribution of the total execution time of different number of request. M1 is the standard architecture without the automatic deployment while M2 is our propose architecture that dynamically deploy the service to the secondary resources. When the number of request is small, M1 outperform our architecture since no services waiting in the queue. However, when the number of request increase, our propose method is better. Our method can utilize the secondary resources efficiently and at the same time reduce the total execution time.

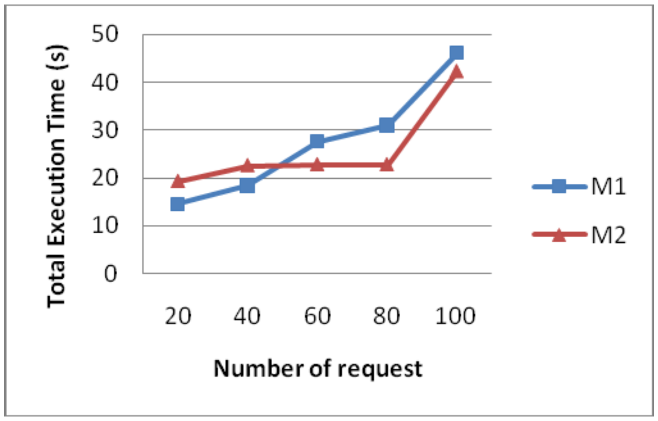

Figure 3. Total Execution Time using two methods

5. Conclusion

Web services are suitable for implementing in the distributed and heterogeneous platform. However, more research is required to discover their potential benefit. This paper presents an automatic web service deployment that able to handle the increasing number of users' request in an effective manner. The major contribution of this paper is the efficient usage of secondary resources. Since most of the research concentrate on primary resources, our research focus on the secondary resources. The secondary resources include those computers in the students' lab that are underutilized. In addition, most of these computers nowadays consist of multiple cores. A partial of these compute resources can be shared without affect the resources performance. Experimental results show that our architecture reduce the number of tasks waiting in the queue and reduce the overall execution time.

Our future work consists of providing intelligent scheduling, interface for self healing, and more advanced security features. In addition, we will test the architecture using the linear and exponential parametric services. Besides, we may consider working on language based and model based solutions deployment.

6. Acknowledgement

This work is partly financially supported by the university research grant under no PJP (FS251/2008B).

7. References

[1] Manoj Mansukhani, Service Oriented Architecture White Paper, HP Inc, June 2005.

[2] Seung-Hyun Lee, Dong-Ryeol Shin, Web Service QoS in Multi-Domain, In 10th IEEE International Conference on Advanced Communication Technology (ICACT), 2008.

[3] Mossab A. Al Hunaity, Towards an Efficient Quality Based Web Service Discovery Framework, IEEE Congress on Services, 2008.

[4] Yan Yang, et al., An Approach to QoS-aware Service Selection in Dynamic Web Service Composition, In IEEE Third International Conference on Networking and Services (ICNS'07), 2007.

[5] Jingya Zhou, et al., QoS Adaption aware Algorithm for Grid Service Selection, In 12th IEEE International Conference Computer Supported Cooperative Work in Design, 2008.

[6] Daniel Lazaro, et al., An Architecture for Decentralized Service Deployment, In IEEE International Conference on Complex, Intelligent and Software Intensive Systems, 2008.

[7] Pairot, C., et al., Deploying Wide-Area Applications Is a Snap. IEEE Internet Computing, 11 (2) 72-79, 2007.

[8] Eun-Kyu Byun, et al., A Dynamic Grid Services Deployment Mechanism for On-Demand Resource Provisioning, In IEEE International Symposium on Cluster Computing and the Grid, 2005.

[9] X. Jiang and D. Xu, SODA: a Service-On-Demand Architecture for Application Service Hosting Utility Platforms, In Proceedings of the IEEE International Symposium on High Performance Distributed Computing, 2003.

[10] G. Wasson, et al., OGSI.NET: OGSI-compliance on the .NET Framework, In Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, 2004.

[11] W. Goscinski and D. Abramson, Distributed Ant: A System to Support Application Deployment in the Grid, IEEE/ACM International Workshop on Grid Computing, 2004.

[12] Gabor Kecskemeti, et al., Automatic Service Deployment Using Virtualisation, In IEEE 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2008.

[13] K. Keahey, et al., Virtual workspaces in the grid. ANL/MCS-P1231-0205, 2005.

[14] I. Krsul, et al., Vmplants: Providing and managing virtual machine execution environments for grid computing. In International Conference on High Performance Computing, Networking and Storage (SC04), 2004.

[15] Javier Alonso, et al., High-Available Grid Services through the use of Virtualized Clustering, In 8th IEEE Grid Computing Conference, 2007.

[16] Vanish Talwar, et al., Comparison of Approaches to Service Deployment, In Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICSCS'05), 2005.

[17] OASIS Standard, Web Services Business Process Execution Language Version 2.0, 11 April 2007. http://www.oasis-open.org/committees/download.php /23964/wsbpel-v2.0-primer.htm.

[18], Aaron Skonnard, Run ASMX Without IIS, MSDN Magazine, December 2004.