RC 22470 (W0205-219) May 31, 2002

IBM Research Report

A Web-Services-Based Deployment Framework in Grid Computing Environment

Zongwei Luo, Shyh-Kwei Chen, Santhosh Kumaran, Liang-Jie Zhang, Jen-Yao Chung, Henry Chang
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

A Web-Services-Based Deployment Framework In Grid Computing Environment

Zongwei Luo, Shyh-Kwei Chen, Santhosh Kumaran, Liang-Jie Zhang, Jen-Yao Chung, Henry Chang
IBM TJ Watson Research Center
PO Box 218
Yorktown, NY 10598, USA
Email: zongwei@us.ibm.com

ABSTRACT

Grid computing offers great opportunities for companies to tap new streams of revenues by taking advantages of the wired computing powers based on the grid service architectures. Resource allocation is one of the key concerns in such a computing environment. In this paper, we present a deployment framework for grid computing. The framework enables open grid services to dynamically deploy (deploy, update, and remove) computing powers including services, and services supporting runtime, etc. in the grids. One of the key components in this framework is the service deployment gateway that offers grid deployment services. The gateway is built upon an integration platform that captures the deployment logics. Implementation details are also provided in this paper to demonstrate this framework.

Keywords: Deployment Framework, Grid Computing, OGSA, Web Services, Web Services Gateway.

1. INTRODUCTION

In the software development life-cycle, deployment is the step to deliver the software package over actual runtime systems. Our experience told us that a large amount of time was spent on the software deployment. A lot of work is mostly mechanical and highly repeatable but still requires specific skills. Software deployment is thus a perfect re-engineering target.

In the Internet computing environment, various cooperates form virtual computing coalitions from disparate resources. To meet the demands of dynamic business operations, mechanisms are needed to realize dynamic deployment services. Such dynamic service deployment systems also make it possible for the companies to better utilize the wired computing power.

As the virtual organizations become more common, it is increasingly required to deploy services remotely. Existing solutions like Installshields (www.installshields.com) help users to package the software into an installable file. The file is then distributed to deploy the packaged software.

In web services enabled systems, service client can invoke a web service by using a known URL or searching the Universal Description, Discovery and Integration (UDDI) registries for service invocation information. The UDDI-based search and discovery method enables web services as the building blocks for dynamic e-business. Since resources are allocated across different service locations, it is critical to deploy services, removing services, updating services, and re-deploy services to another locations without disrupting the currently running services.

In this paper, we propose a web services deployment framework in the grid computing environment based on the Open Grid Service Architecture (OGSA) [1]. Our deployment framework does not replace common deployment software products like Installshield, but provides a higher layer to utilize such deployment software in the grid environment. We envision the deployment service as one of the basic services in the grid environment. Organization of this paper is as follows. We first present the deployment

framework in grid environment in Section 2. We develop a deployment language to describe the deployment task in grid environment in Section 3. Then we present the deployment service gateway in Section 4. The services in the framework are described in Section 5. The service implementations are described in Section 6. Section 7 concludes this paper with discussion.

2. DEPLOYMENT FRAMEWORK

The deployment framework is based on open grid architecture. The deployment services are described in web services description languages. Interfaces specified in Open Grid Architecture are also implemented. The required interface for OGSA is GridService [1]:

- FindServiceData: Query a variety of information about the grid service instance, including basic introspection information (handle, reference, primary key, home handleMap), richer per-instance information, and service-specific information (e.g., service instances known to a registry). Extensible support for various query languages.
- SetTerminationTime: set (and get) termination time for Grid service instance.
- Destroy: Terminate Grid service instance.

The deployment framework deploys applications in runtimes that are heterogeneous and distributed in nature. Each runtime has its own deployment syntax. Instead of creating another syntax that requires each runtime to adapt to, the deployment framework takes a mediation approach. The deployment service gateway mediates the deployment tasks. To ease the deployment integration, a deployment mediation language is used. The mediation language describes the deployment tasks. The deployment service gateway accepts deployment requests specified through the deployment language via deployment connector and invokes the deployment implementation. The deployment adapter implements the deployment.

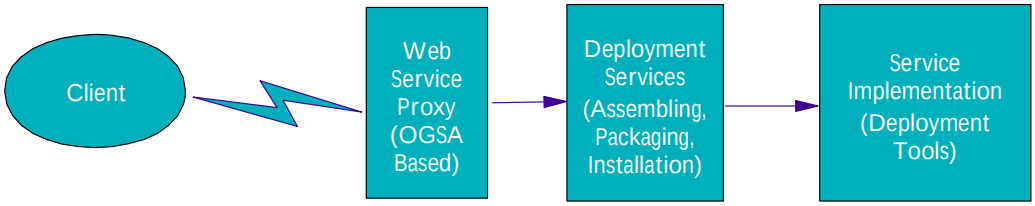

Figure 1. OGSA based deployment service framework

A possible functional architecture for the deployment framework is shown in Figure 1.

- Client: a client can request deployment services remotely via web services calls.
- Deployment service gateway: it provides deployment services compatible with OGSA. It allows clients to discover available services provided. It also provides advanced deployment services such as solution assembling, solution packaging, in addition to solution installation services.
- Deployment services language: it is the language used to describe the deployment tasks that may be the payload in the web services requests by clients. It is platform independent, and compatible with the Web Services Description Language (WSDL) [3] and the Web Services Flow Language (WSFL) [4].

The framework is layered conceptually as follows:

- Deployment description layer: it is the layer for the deployment service language.
- Deployment interpretation layer: this is the layer that interprets the deployment request described in deployment services language.
- Deployment request brokering layer: this is the layer that routes the deployment

services request to appropriate deployment service implementations.
- Deployment adapter layer: this layer contains various deployment service modules, including deployment assembling, deployment packaging, and deployment installation modules.
- Deployment implementation layer: this is the layer that deploys solutions into target systems.

3. DEPLOYMENT SERVICE LANGUAGE

In the service deployment domain, we propose a deployment language to describe the service deployment task. The deployment task described in the language is then mapped to a specific runtime system.

Deployment language includes the following:
- Solution package description: it describes the solution packages using concepts from deployment taxonomy.
- Solution package provider profile: it describes the solution package provider.
- Solution deployment model: it describes the solution deployment implementation.
- Service description: it describes the open grid web service provided.
- Service model: it describes how the open grid web services are implemented.
- Service provider model: it describes the open grid service provider profile.
- Service connection profile: it describes the open grid service connection profile.
- Service quality profile: it describes the quality of the open grid services provided.

The deployment service language follows and extends the WSDL [3] and the WSFL [4] syntax to include software deployment taxonomy. Service descriptions are accessible freely by service clients and WSDL and WSFL compatible.

The software deployment taxonomy includes the following concepts. They are consistent with concepts used in common software modeling languages such as Universal Modeling Language (UML) [5].

- Package
- Node
- Node instance
- Component
- Component instance
- Interface
- Object
- Composition
- Communicates
- Dependency
- Constraint
- Comment

4. DEPLOYMENT SERVICE GATEWAY

The key component to interpret the deployment language is the deployment service gateway. The gateway reads in the deployment language and then takes corresponding actions to deploy the software packages. The gateway could be implemented in any platform. In our implementation, the gateway is based on an application server platform. The interface to incoming request is the web service proxy. The service proxy accepts web service requests wrapped in an acceptable transport protocol. The web service proxy is built to intercept HTTP/SOAP (Simple Object Access Protocol [6]) requests or service requests through other bindings. The adapter provides capabilities to deploy service implementation into a runtime system.

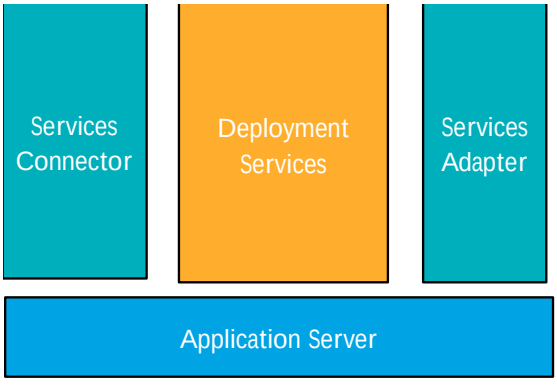

Figure 2. Deployment service gateway component view

The component architecture for service gateway is shown in Figure 2. Current implementation for this architecture is as follows:

- The web services proxy is built as a Deployment Connector that is responsible for making the connection using the http transport and the SOAP protocol for exchanging messages. This connector is agnostic to the type of the message. Such a connector in the future will typically present a Java Connector Architecture (JCA) interface.
- The web services gateway also consists of one or more Deployment Adapters that use the Deployment Connector. The Adapter is specific to the type of message (e.g. Purchase Order Application Adapter) and essentially consists of the business logic that is used to process the message.
- The Deployment Services implemented by the Deployment Adapters are exposed as Web Services. These Web Deployment Services connections are described using WSDL. The service connection descriptions are made OGSA consistent as well.

Figure 3 shows the Web Deployment Service interaction pattern when the Deployment Gateway component is engaged as in the deployment interaction.

- A business partner client (i.e. application) makes a request, e.g., buyer invoking a web business service provided by the deployment gateway.

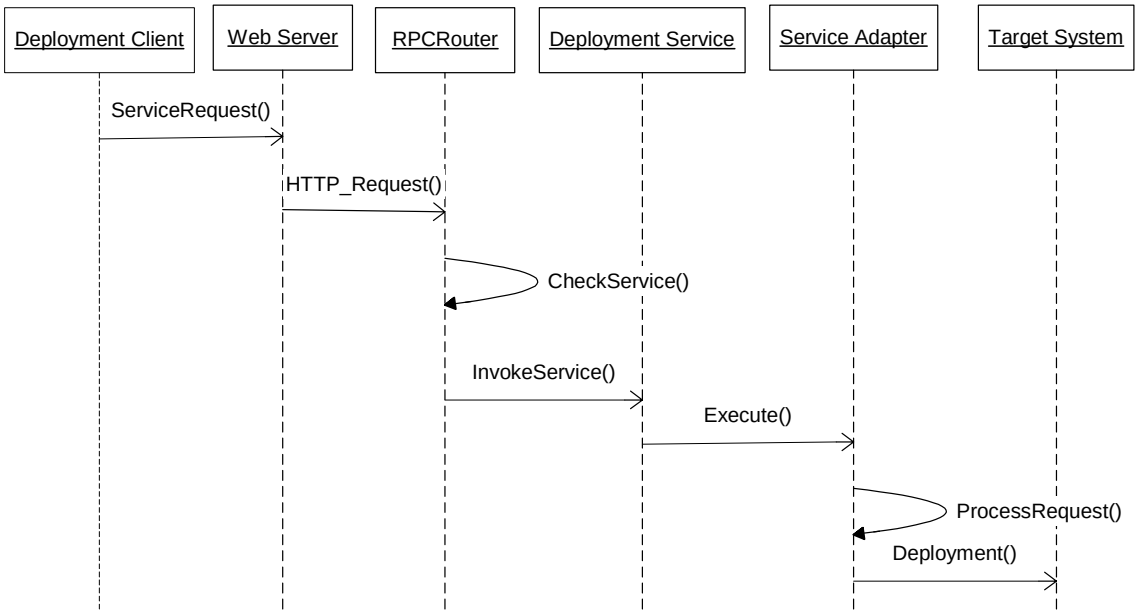

Figure 3. Interaction diagram in deployment service gateway

- The web server in the Deployment Connector receives the request, a SOAP message, and routes it to the SOAP Servlet for handling.
- The SOAP Servlet (also know as the rpcrouter) in the Deployment Connector checks against the registry of deployed services.
- The SOAP Servlet invokes the pluggable service provider, i.e., the Deployment Service.
- The Deployment Service in turn invokes the Deployment Adapter that provides the implementation for the Web Deployment Service.

- The Deployment Adapter processes the incoming SOAP message.
- The deployment tasks eventually are fulfilled upon the target systems.

5. DEPLOYMENT SERVICES

The deployment services gateway provides various services for deploying solutions in the grid computing environment. It offers OGSA-based web services connections. It provides assembling services to assemble the solutions according the clients' requests. It provides packaging services to package different solution parts for their deployment platforms. It can invoke various deployment tools to install solutions in the target systems. It can also record and verify the package dependences.

- Connection service: This is the service fulfilled by the web service proxy. The actual communication binding is the web services bindings such as SOAP[6].
- Assembling service: Currently the solution assembling is based on a state-machine-based process language. A partial schema described in XML Schema [7] is shown in Figure 4. Assembling based on WSFL is being implemented. The process or flow language links different parts in the deployable solutions. The actual deployment is made persistent, i.e., the deployment configuration information is stored in a persistent storage.
- Packaging service: It includes the solution package services. Different platforms will accept different package formats. The packages for WebSphere Application Server (WAS) [8] platform, e.g., will be like EAR, WAR, and PAR etc. formats. One package tool used in WAS platform is ejbdeploy tool.
- Installation service: It includes the services for solution installation, update, and remove, etc.

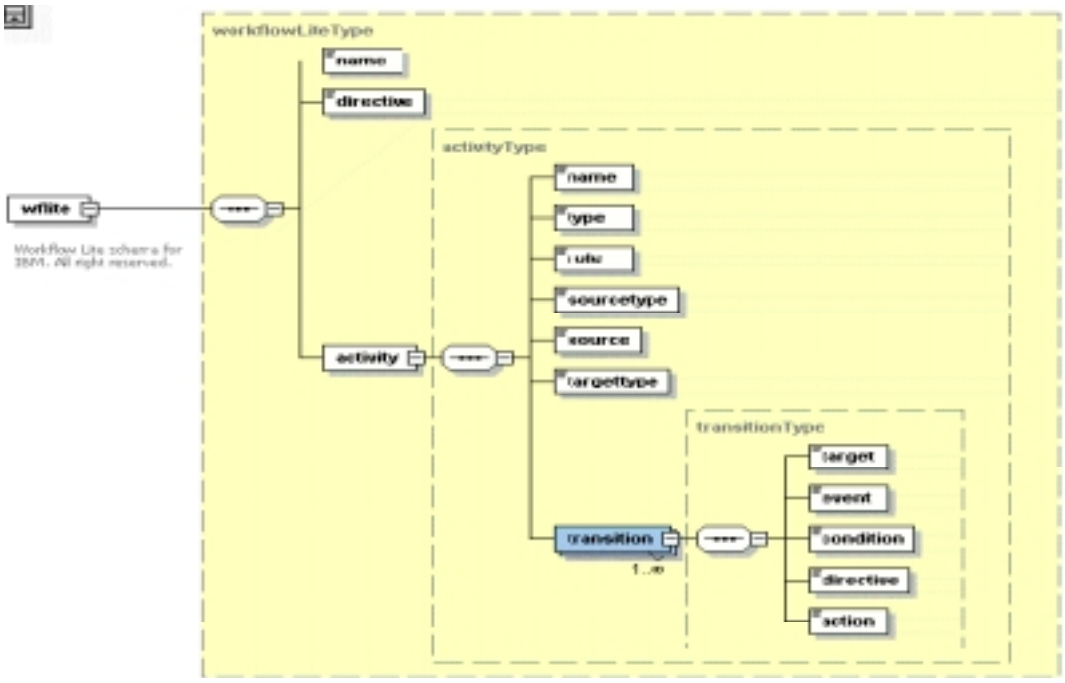

Figure 4. Schema for a state-machine-based process flow language

6. SERVICES IMPLEMENTATION

In this section, the deployment service implementations will be described using a concrete example platform – an application sever platform.

WAS application deployment adapter is one of the deployment services adapters. The adapter

reads WAS XML configuration information [8], then loads in corresponding EAR files and configures the web application server.

Before the XML configuration files are read, different parts in the solutions may need to be packaged. The current version of the packaging tool reads in the package XML description that is part of the solution deployment request in deployment service language. The XML description contains the solution definition, from which, information about the available platforms is retrieved. In the case of EAR packaging, the ejbdeploy is invoked to create necessary EAR files when they do not exist in the packaged solution.

During the installation, user interaction is possible. When this mode is activated, the installation tools will let the user select a packaged solution and provide information for where every package of components is to be transferred. Once all components are transferred, all configurable components are shown so the user can select them and load the appropriate application to configure them.

In summary, the EAR solution deployment steps in WAS platform are as follows:

- Deployment gateway receives the deployment request and retrieves the solution deployment XML files.
- Deployment gateway develops the necessary resources for the solution. The code generator in the gateway could generate certain resources such as EJB and JSP. Deployment gateway publishes the resources in a deployment repository. A notification is generated saying the solution is ready for deployment.
- Solution deployment gateway retrieves the solution package when it receives the notification and reads the notification.
- Solution deployment gateway launches the XMLConfig application to deploy the solution.

7. DISCUSSION

In this paper, we have presented a deployment framework in grid environment. The key component in the framework – deployment services gateway presents web services interfaces for the provided deployment services such as assembling service, packaging service, and installation services, etc. A deployment service language is developed to describe the deployment services.

Grid computing offers great opportunities for companies to tap new streams of revenues by taking advantages of the wired computing powers based on the grid service architectures. Resource allocation is one of the key concerns in such a computing environment. The framework presented enables open grid services to dynamically deploy (deploy, update, and remove) computing powers including services, and services supporting runtime, etc. in the grids.

8. REFERENCE

[1] UDDI, http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[2] I. Foster, etc. The Physiology of the Grid, An Open Grid Services Architecture for Distributed Systems Integration, http://www.globus.org/research/papers/ogsa.pdf

[3] WSDL, http://www.w3.org/TR/wsdl

[4] WSFL, http://www-4.ibm.com/software/solutions/webservices/pdf/wsfl.pdf

[5] M. Fowler, K. Scott, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 2nd Edition, August 1999

[6] SOAP, http://www.w3.org/TR/2001/WD-soap12-part0-20011217/

[7] XML Schema, http://www.w3.org/XML/Schema

[8] WebSphere Application Server, http://www.ibm.com/software/webservers/appserv