Jade for dynamic VOs

Nikos Parlavantzas

SARDES – INRIA Rhône-Alpes

August 8, 2007

Contents

1. Purpose

This document outlines the current version of the customised Jade framework for dynamic virtual organisations. Moreover, it provides a simple example of its use in deploying distributed applications.

2. Jade for dynamic VOs

Jade is an extensible framework for building autonomic management systems. Within Grid4All, Jade is being customised to address autonomic management of dynamic virtual organisations (VOs). This work involves integrating into Jade overlay services responsible for monitoring and controlling the main VO elements, namely, members, resources, and components [1]. Two overlay services have currently been integrated into Jade, namely, resource discovery, and component deployment. Integration of further services is on-going work. Based on these two services, the customised Jade framework supports deploying distributed applications on the overlay network taking into account requirements on underlying resources. The following sections outline the architecture of the current customised Jade version, and explain the process of deploying a simple application.

3. Architecture

At run-time, Jade comprises one *JadeBoot* and one or more *JadeNode* components. JadeNodes run on the managed physical machines and are responsible for hosting application components. The JadeBoot component bootstraps the system and exposes core services for managing the distributed applications (e.g., deployment service, and system representation service). The main functionality added in the customised Jade is deploying applications on an overlay network driven by resource requirements. Figure 1 shows the runtime structure of the customised Jade, focusing on components relevant to this new functionality. These components are outlined next.

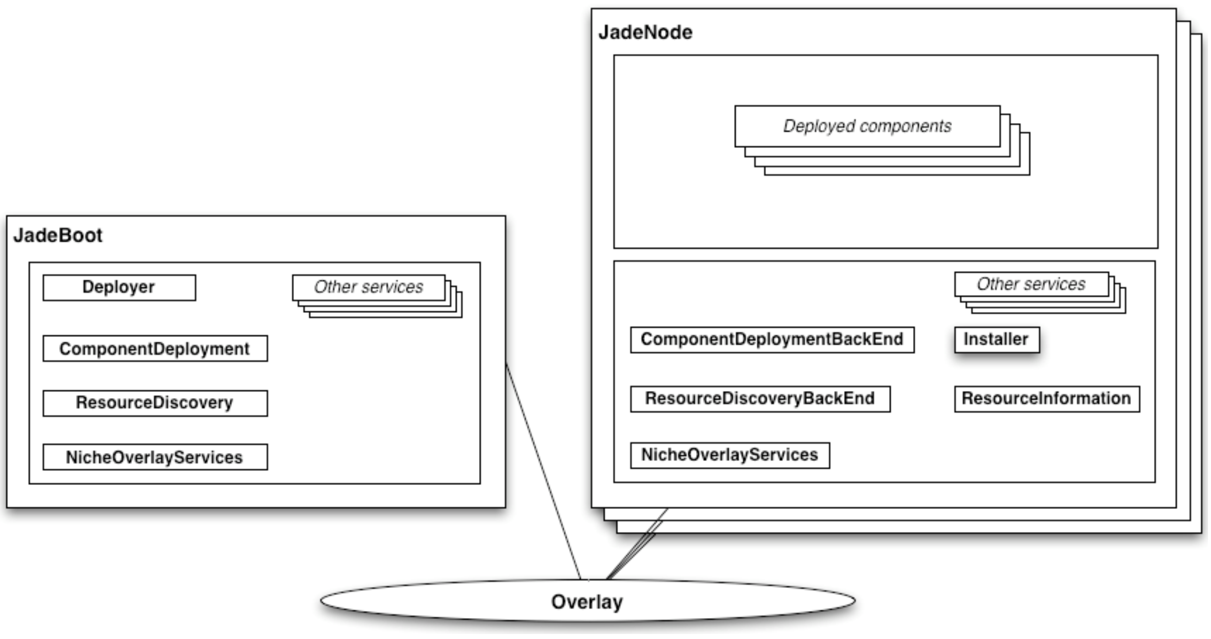


Figure 1. Architecture of customized Jade

The *NicheOverlayServices* component implements VO management services on top of a structured overlay network. Specifically, the present version implements discovering resources (currently, machines) and deploying components on those resources. The NicheOverlayServices API is decomposed and encapsulated by higher-level components that facilitate integration with existing Jade components. The *ResourceDiscovery*/ *ResourceDiscoveryBackEnd* components encapsulate the discovery service, and the *ComponentDeployment*/ *ComponentDeploymentBackEnd* encapsulate the deployment service. The front-end components capture interactions with managers (i.e., initiators of resource queries or of deployment requests), and the back-end components capture interactions with managed elements (i.e., resources and deployed components). Front-end components reside in JadeBoot, which is currently the single component with manager responsibilities.

The *Deployer* component is responsible for installing and configuring applications given a description of their software architecture in ADL. The supported ADL is an extended version of the standard Fractal ADL that allows specifying resource requirements for components (see example in next section). Deployer uses ResourceDiscovery to find machines that satisfy such requirements and ComponentDeployment to deploy components on these machines. At the managed-element side, ResourceDiscoveryBackEnd uses *ResourceInformation* to obtain machine properties in order to determine whether the machine satisfies a given set of resource requirements. ComponentDeploymentBackEnd uses *Installer* to install the required packages and instantiate a given application component.

4. Using Jade

4.1 Installing Jade

The source code for the Grid4All version of Jade can be downloaded from the SVN server at: https://proton.inrialpes.fr/svn/Grid4All

Jade supports a high degree of configurability; full details on configuring Jade are given in [2] and [3]. At a minimum, configuring Jade requires setting references to local directories in the following files: etc/oscar/bundle.properties, etc/oscar/bundle-jadeboot.properties, and etc/execute.properties. Configuring the underlying Niche communication library minimally requires setting the publishAddress parameter in etc/dsk/dks/dksParam.prop to a URL that maps to the webcache.php script. Full details on configuring Niche are given in [4].

4.2 Starting Jade

To start Jade, we first launch one JadeBoot and then any number of JadeNodes on one or more physical machines. To this end, we run the Ant tasks *jadeboot* and *jadenode* defined in the build.xml file located at the root directory of the source code. All the launched components join an overlay network managed by the Niche library.

4.3 Deploying applications on Jade

4.3.1 Architecture description

To deploy an application on a Jade system, we need a description of its architecture in ADL. As an example, consider a simple application consisting of three components (see Figure 2): the *Apache* component that wraps an Apache web server, the *Start* component that launches the previous component, and the *MyApache* composite component that encapsulates both previous components.

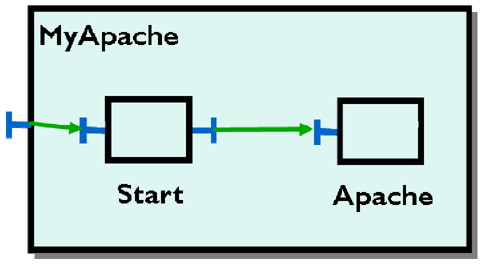


Figure 2. Example application

The architecture of this application expressed in ADL is shown in Figure 3.

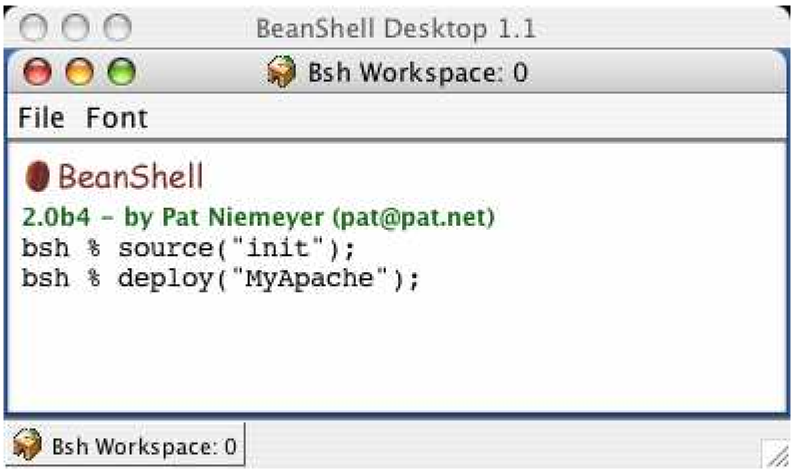
```
<definition name="MyApache">
            <interface name="service" role="server"
                    signature="org.objectweb.jasmine.jade.service.Service" />
            <component name="start"
                definition="org.objectweb.jasmine.jade.resource.start.StartType">
                    <virtual-node name="node2" resourceReqs="MemorySize>2GB"/>
            </component>
            <component name="apache"
                definition="fr.jade.resource.j2ee.apache.ApacheResourceType">
                    <attributes
                        signature="fr.jade.fractal.api.control.GenericAttributeController">
                              <attribute name="dirLocal" value="/tmp/j2ee" />
                              <attribute name="resourceName" value="apache" />
                              <attribute name="user" value="jlegrand" />
                              <attribute name="group" value="wheel" />
                              <attribute name="port" value="9090" />
                              <attribute name="serverAdmin" value="julien.legrand@inrialpes.fr" />
                    </attributes>
                    <virtual-node name="node2" />
                    <packages>
                              <package name="Apache v1.3.37 mac Wrapper" >
                                      <property name="local.dir" value="/tmp/j2ee"/>
                              </package>
                    </packages>
            </component>
            <binding client="this.service" server="start.service" />
            <binding client="start.rsrc_apache" server="apache.resource" />
            <virtual-node name="node1" resourceReqs="MemorySize>1GB"/>
</definition>
```

Figure 3. ADL description of example application

Resource requirements of components are expressed using the *virtual node* element. At deployment time, each virtual node is mapped to a concrete machine that conforms to the associated requirements. The example description states that Apache and Start are deployed on a machine with memory larger than 2GB, and MyApache is deployed on a different machine with memory larger than 1 GB. Note that the current prototype has no general support for specifying and interpreting complex resource requirements beyond this simple memory-based example.

4.3.2 Deployment

The most straightforward way to deploy applications on Jade is with *BeanShell*, a Java source interpreter with object scripting language facilities [5]. The BeanShell graphical console is launched with the Ant task *beanshell_console* in the build.xml file. After launching the console, we use the command source(“init”) to load a set of useful scripts for controlling Jade systems (see Figure 4). To deploy the example application, we then use the command deploy(“MyApache”) assuming that the file MyApache.fractal contains the description in Figure 3. Jade selects two JadeNodes and deploys the three components on them according to the ADL description. The console output of the selected JadeNodes confirms that the deployment has taken place.



```
bsh % source("init");
bsh % deploy("MyApache");
```

Figure 4. Beanshell console

References

[1] Brand, P., Höglund, J., Popov, K., De Palma, N., Boyer F., Parlavantzas, N., Vlassov, V., Al-Shishtawy, A., "The Role of Overlay Services in a Self-managing Framework for Dynamic Virtual Organizations", CoreGRID Workshop on Grid programming model, Grid and P2P systems architecture and Grid systems, tools and environments, Crete, Greece, June 2007.
[2] Jade Getting started, http://proton.inrialpes.fr/~jlegrand/jade/doc/getting_started.pdf
[3] Jade User guide, http://proton.inrialpes.fr/~jlegrand/jade/doc/user_guide.pdf
[4]Niche Getting started, http://korsakov.sics.se/svn/dks/trunk/Niche/Niche%20Getting%20Started%20Tutorial.pdf
[5] BeanShell, Lightweight Scripting for Java, http://www.beanshell.org/