RC24117 (W0611-164) November 28, 2006
Electrical Engineering

IBM Research Report

Experience of Developing and Implementing a Light-Weight Enterprise Grid

Colm Malone, Alex Zlatsin
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Nianjun Zhou
IBM Managed Business Process Services
294 Route 100
Somers, NY 10589

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

Experience of Developing and Implementing a Light-Weight Enterprise Grid

Colm Malone (*)
colm@us.ibm.com

Alex Zlatsin (*)
zlatsin@us.ibm.com

Nianjun Zhou (+)
jzhou@us.ibm.com

(*) IBM T. J. Watson Research Center
PO .Box 218
Yorktown Heights, NY 10598, US
(+) IBM Managed Business Process Services

Abstract

In this paper we present our light-weight grid solution which overcomes many of the more common inhibitors associated with "gridifying" enterprise-type applications. The motivation behind creating a light-weighted grid solution includes: simplified middleware components with a small foot-print and thus easier maintenance of individual resources.

Java-based middleware solutions like Globus®, can be quite heavyweight to install and even more cumbersome to maintain. With our goal of being a light-weight solution and also our desire to target non-dedicated type resources, our middleware layer needed to be as thin as possible.

Our grid solution is a prototype deployed over a geographic distributed world-wide grid system using z/Linux instances hosted under an S/390 mainframe, p-series AIX and x-series Linux workstations.

1. Introduction:

Grid computing is defined as flexible, secure, coordinated resource sharing among a dynamic collection of individuals, institutions, and resources. The sharing of these resources must be tightly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. Therefore, Grid computing presents unique authentication, authorization, resource access, resource discovery, and resource management challenges.

With complicated heterogeneous computing environments and heavy-weight grid middleware implementations, system architects are often frustrated by the deployment hurdles involved with adopting a grid solution. Certain grid middleware implementations require manual installation, or a high level of maintenance for each individual resource. Other grid middleware solutions require resources to be dedicated – which excludes non-dedicated type resources like user workstations.

Our goal is to target any potential resource as being a grid resource incurring as minimal a footprint as possible.

Enabling applications to run on individual grid resource can be a challenge too. Many solutions require administrator-level privileges on each resource. In keeping with our goal of minimal intrusion, we try to run/deploy applications without administrator-level privileges where possible.

The main inhibitors we encountered to deploying applications over the grid were:

1. Abstracting an application to run over the "grid". By this we mean, instead of targeting a specific resource for a specific task, the application should be able designed to run from any resource i.e. it should not care what resource it is running on. We implemented this by:
    a. Installing the software to a sharable network file-system that is accessible from all resources. This avoids installing software to every resource. By avoiding the use of an individual resource's local file

system, we increase the portability of the application to the grid while reducing the intrusion on each individual resource.

b. For synchronous type tasks like Web Services, we maintained the status of the application the grid using a global state file accessible by any resource. This allows the service to run from anywhere and for other resources to take over should the process fail on the current resource.

2. The second inhibitor is related to security. Isolation of application I/O files is needed to prevent accidental (or malicious) access of other users to confidential files. This problem is solved by:

   a. Using a project-specific network file account for storing I/O data files and the requisite software. Credentials delegation to the appropriate local IDs is provided by the Grid Scheduler on "as needed" basis to the relevant grid resources;

   b. Secure Shell (SSH) authentication is the mechanism used for remote job submission of grid applications. Public/private key pairs are used. Public keys being deployed to the appropriate project accounts on each resource. Private keys are secured for each project and are only accessible to the relevant project owner.

The third inhibitor is the virtualization of synchronous applications such as web services. Application URI addresses need to be abstracted from the actual hosting grid resource. With the use of a dynamic hostname, dynamic DNS can reassign the hostname that happens to be running the grid application at scheduling-time.

2. Grid infrastructure Overview

The following figure provides an overview of the existing infrastructure, breaking it down into four broadly grouped components that will be outlined in the next four sections. These four components are entirely custom and specific to light-weight enterprise grid infrastructure, but are built using commonly available software.

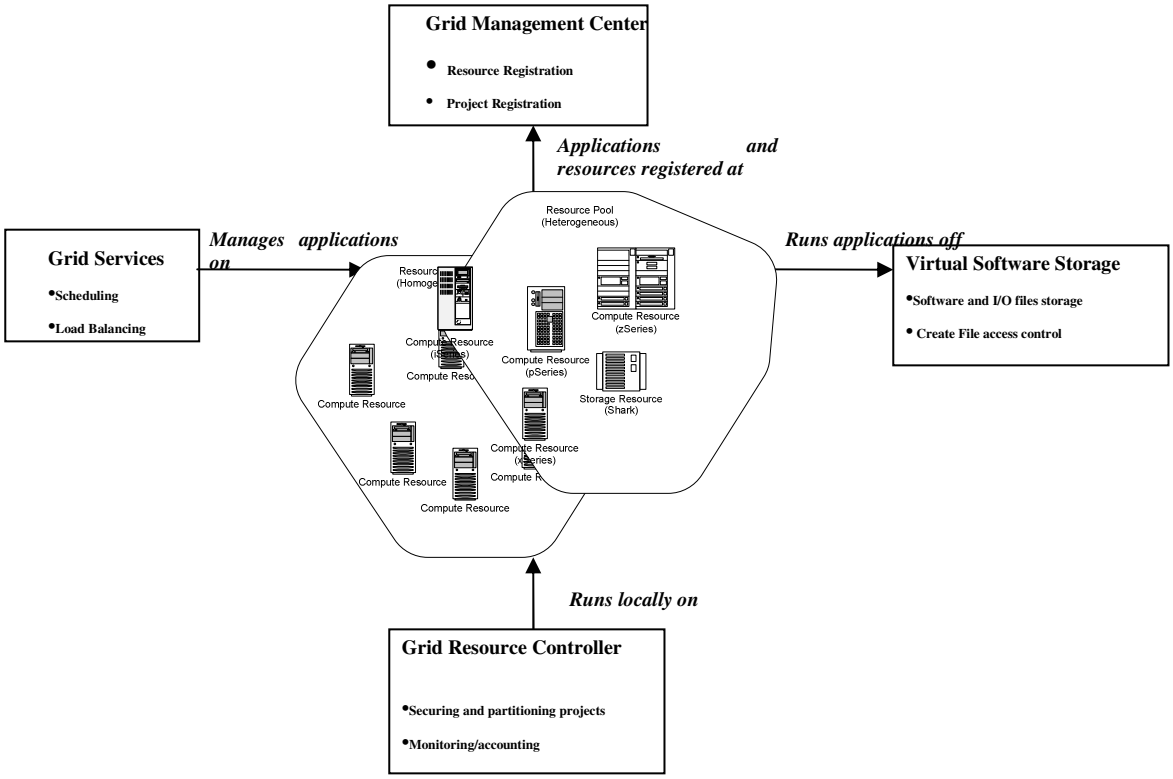

Figure 1. High Level Overview of the Light-Weight Grid Infrastructure

The function of the four components are:

- *Grid Management Center*: provides resource registration and application/project registration, to link project and resources, and to handle Job submission and link project, resource and data storage together.

- *Virtual File System*: is a network-file system to securely store grid application code and data. It allows for single installation of applications with the potential for network optimization via geographical resource pooling;

- *Grid Resource Controller*: secures and partitions projects. It also performs monitoring and accounting tasks;

- *Grid services* – provides scheduling and load balancing.

2.1 Grid Management Center

The *Grid Management Center* is at the center of Grid control. The user interface for the *Grid Management Center* is a Web application, currently running on WebSphere Application Server version 6. It comprises of an Enterprise Java Bean (EJB) to access the DB/2 database backend, Java Servlets and JSP to provide the GUI for the grid users.

The key concepts of the *Grid Management Center* are:

- Project: this consists of an application and a group of users that manage it on the administration site. It also constitutes a unique user ID on all resource machines, matching SSH keys, and a coordinated directory on the *Virtual File System*.

- Project Member - A project member has access to read-only information about the project, including: status, description, and SSH keys for resource access.
- Resource Pool - A resource pool is a logical group of resources that have been manually defined by some common relationship. For instance, resource pools may be created for a particular geographical region.

- Grid map file - This file is "borrowed" extension of the [4] Globus Toolkit *grid-mapfile*. The grid map file associates an application/project identity to a physical local user ID, for the purpose of job execution. The grid map file also includes Virtual file-system account and auditing information.

- User ID (UID) - for consistency across grid resources and an account code ID for tracking and/or billing purposes.

- Message broker: tracks changes to the grid map file and ensures these changes are propagated to each grid resource.

The following diagram represents the interactions of all these systems as newly created projects and their related user accounts propagate throughout the grid system.

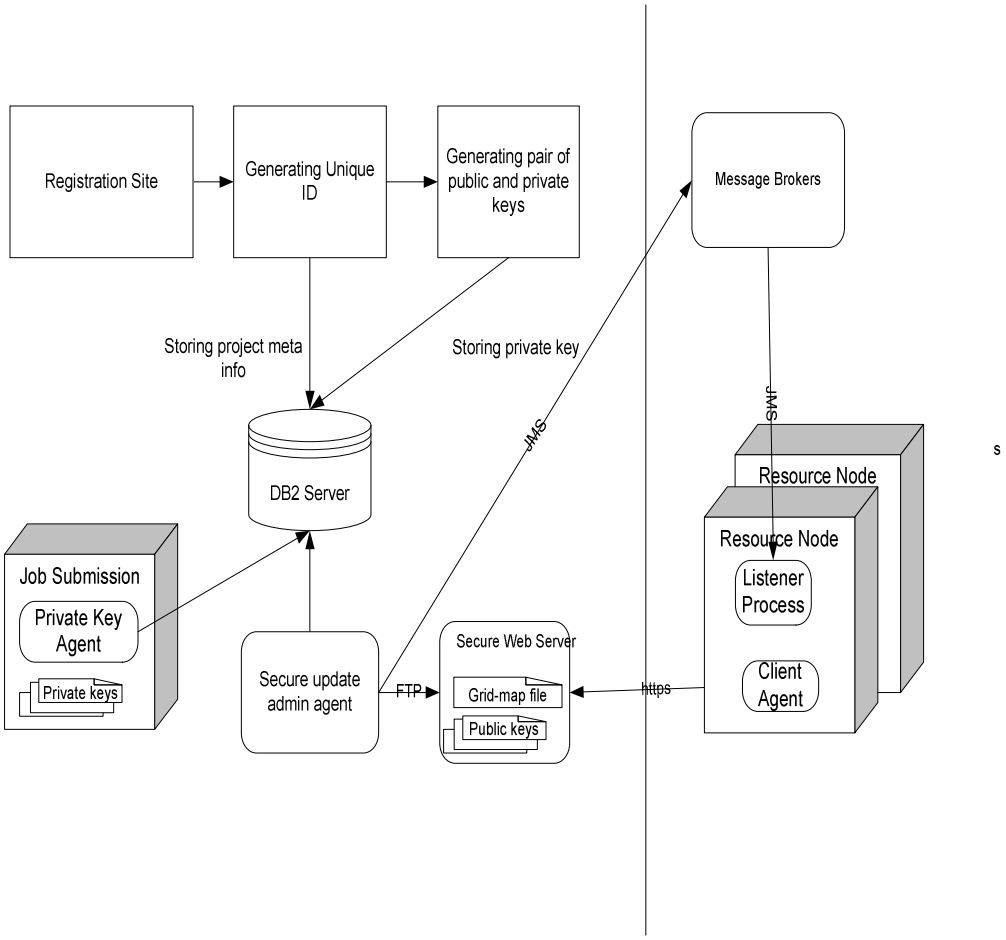

Figure 2 Process of Project Creation and Security Control of Projects with Resources

2.2 Virtual File System

In this light-weight grid implementation, the [2] IBM Global Storage Architecture (GSA) is used as the virtual file system. GSA maintains an LDAP (Lightweight Directory Access Protocol) directory of user IDs and passwords. Clients securely authenticate with their user ID and password directly to the GSA server. This in turn opens up NFS file sharing access to the client's current IP address and UID (numeric User ID) for a specified amount of time. With GSA's NFS authentication, the access controls are stored on the server, instead of storing a token locally on the client. More importantly, with GSA's authentication, the client can obtain credentials for not only its own IP address/UID but can also delegate credentials to a collection of other IP addresses/UID's without having to contact those resources directly. This allows the grid scheduler to delegate credential access to the requisite software and data before scheduling the job. IP address based authentication does have some security disadvantages however. IP spoofing—which is easy and prevalent on the internet—can lead to an unauthorized resource gaining access to private data. Within a managed enterprise network, however, switched-routing prevents problems like IP spoofing.

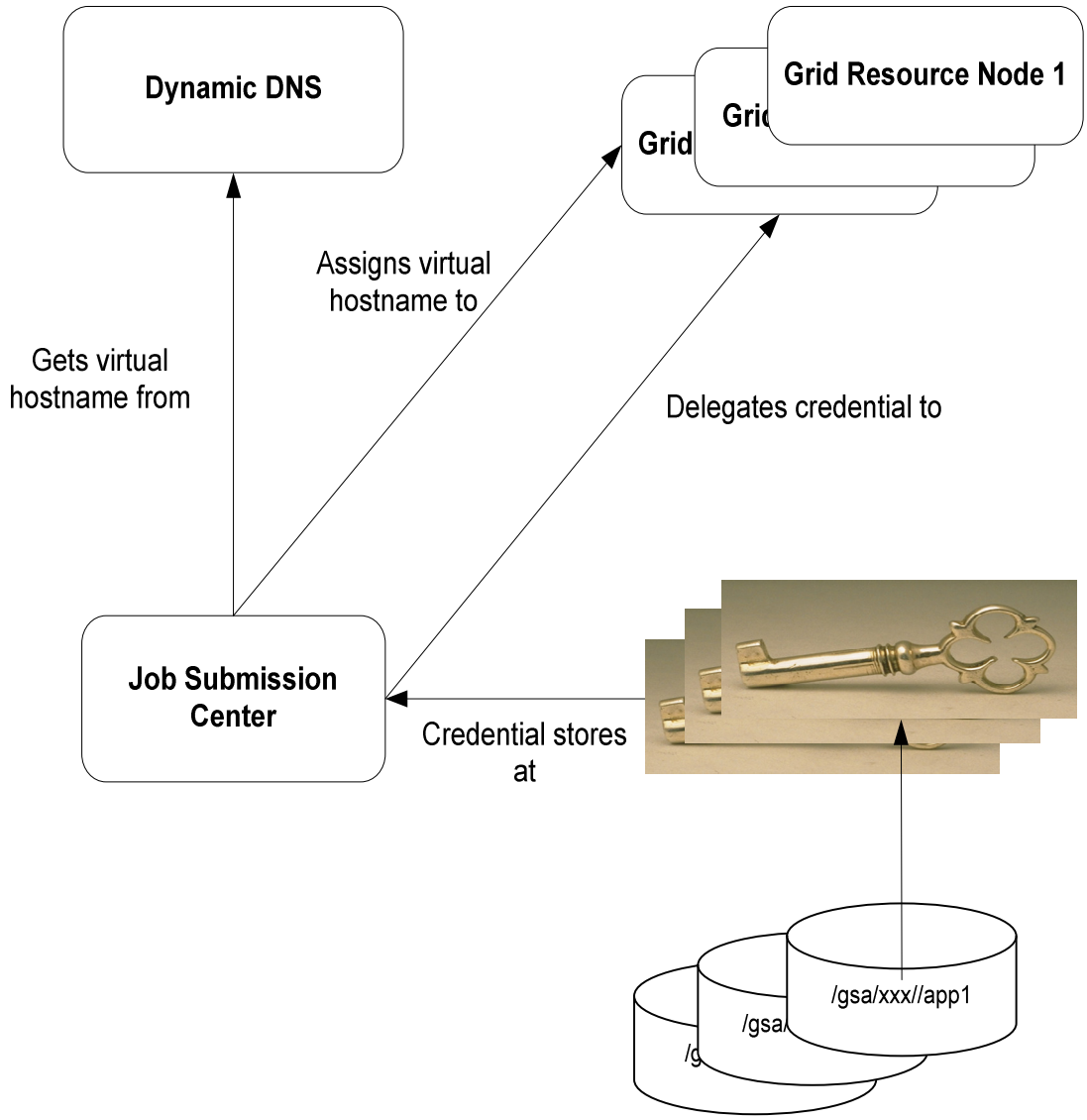

Figure 3 Process of running an application of a project in grid environment

2.3 Grid Resource Controller

The *Grid Resource Controller* consists of scripts and daemons to register, maintain and monitor each individual Grid Resource. The Grid Resource Controller consists of:

- A process to register a Grid Resource with the *Grid Management Center*
- Grid Project ID synchronization with the *Grid Management Center*
- Grid Resource meta-data monitoring & reporting to the *Grid Management Center*
- Secure Remote Access to Grid Resources
- Grid Services Daemon

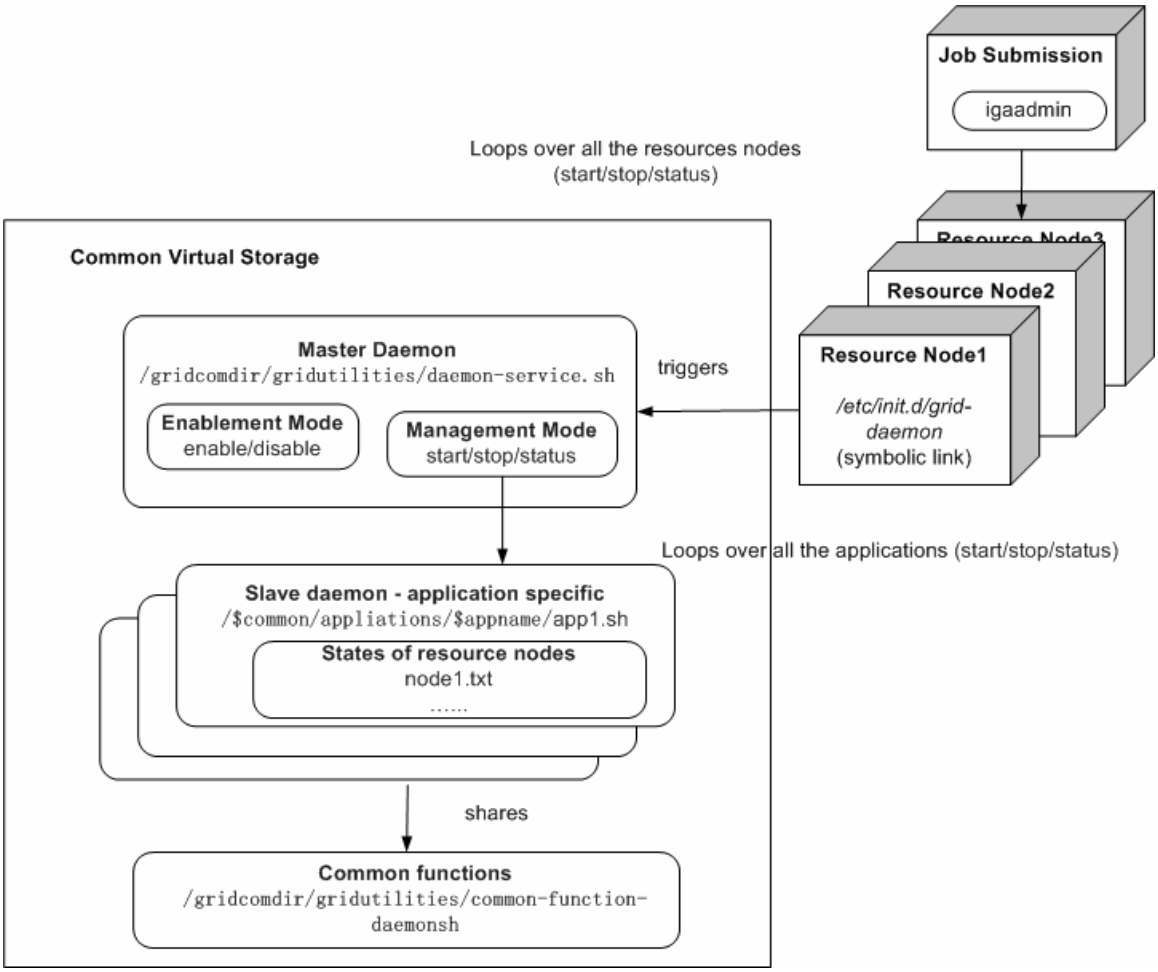

Figure 4. Grid services daemon control flow

2.4 Grid services

The grid services component is responsible for providing job scheduling and IP virtualization

2.4.1 Job Scheduling

The main purpose of Grid scheduler is to control an application. The application can be executed by a user multiple times. In some situations, you may want to run just one instance of an application to avoid data contention and to improve performance. In other situations, you may want to run multiple instances of an application when availability and reliability is important. Some application tasks run sequentially, some in parallel. In the case of multiple instances of an application, the number of concurrent application instances needs to be limited due to resource constraints.

A serial application can run asynchronously to achieve faster execution. In contrast parallel or bidirectional applications often require coordinated synchronization. Some applications can be restricted to particular geographical locations, while others can run anywhere.

We chose the [3] IBM LoadLeveler as our grid scheduler. Load-balancing is based on a resource's individual capabilities as well as pooling resources that are geographically proximate.

2.4.2 IP Virtualization

Hosting dedicated services on the grid should be transparent to the application owner. To achieve this, the individual resource(s) hosting the service needs to be abstracted from the actual service address (URI). This is a common problem with services in general. Dynamic DNS can be utilized to abstract the resource hostname from the service's URI address. The basic process is as follows:

- The scheduler selects a resource to host the service
- The grid scheduler sends a dynamic DNS update for the service address
- The scheduler dispatches the job to the selected resource
- Requests are automatically routed to the pertinent grid resource

3. Installing to *Virtual File System*

The following demonstrates the logistical pitfalls of installing a software application over a network file system.

We use Apache/Tomcat as an example of "gridifying" an application. The process is summarized as follows:

- Single installation of Tomcat installed to the virtual file-system.
- The Apache/Tomcat "container" requires a Java JDK. A JDK must be accessible for each type of resource regardless of hardware architecture. z/Linux resources must have access to a z/Linux JDK; AIX resources must have access to a p-series type JDK etc. To avoid local installations – these different JDK architectures are installed into the virtual file-system.
- Create an application container for each Tomcat cluster group
- Temporary, log, and configuration for each individual Tomcat "node". Each node maps to a specific grid resource
- Tomcat nodes are virtualized by using dynamic DNS so the underlying grid resources can be dynamically changed.

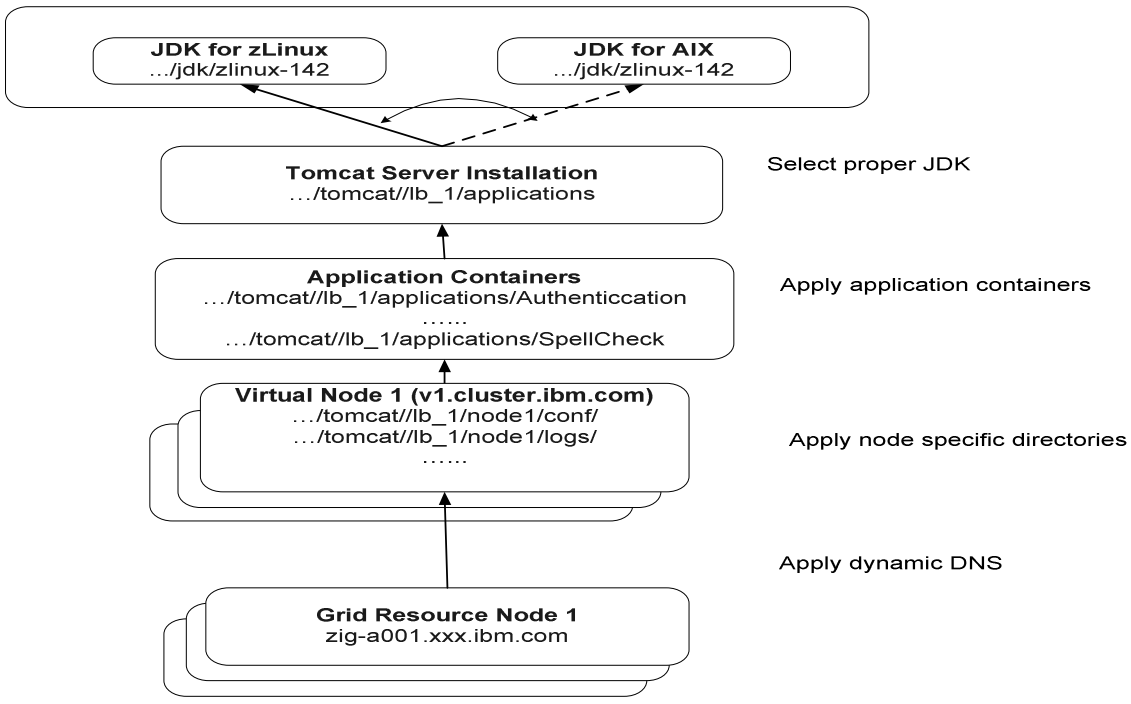

Figure 5. Grid application (Tomcat) in virtual file

4. Summary

This light-weight grid solution utilizes basic open source components to create a thin grid middleware layer. This simplifies the configuration and maintenance of individual grid resources. It also significantly reduces the footprint on each resource. Also, licensing issues on individual resources is not an issue with the use of open-source components.

WebSphere Application Server is used as the grid portal front-end for users.

A sharable network file-system allows grid applications to be deployed to all grid resources, regardless of architecture, all at the same time.

By abstracting where an application runs and thus allowing it to run from "anywhere" we free that application to be hosted anywhere on the grid. Dynamically DNS allows a service URI address to be static while the underlying hosting grid resource can change.

Project-specific network file accounts are used to securely segregate file access between projects. Credential delegation is used to deliver access to only the pertinent grid resources and their local accounts for the particular task at hand.

Secure Shell (SSH) is used for remote job submission. Authentication is via public/private key pairs – the public keys being deployed to each resource and the private keys securely accessible to only the appropriate application owners.

IBM LoadLeveler is used as the grid scheduler, profiling grid resources based on general capabilities and architecture, as well as pooling groups of resources into logical groups such as geographical location.

5. Acknowledgement

The authors would like to express their appreciation of support from the following people: Jean-Pierre Prost; Dikran S. Meliksetian; Soobaek Jang; Joshua M. Woods; and Amarjit Bahl.

6. References

[1] IBM Redbook, published 14 June 2006 A Virtualization Experience: IBM Worldwide

Grid Implementation, SG24-7229-00

[2] IBM Redbook: Implementing NFSv4 in the Enterprise: planning and migration strategies, SG24-6657

[3] IBM Tivoli Workload Scheduler LoadLeveler®

[4] The Globus® Toolkit

http://www.globus.org/toolkit/

[5] IBM RedBook: Workload Management with LoadLeveler®, SG24-6038-00