Chapter 10
Distributed Libraries Management for Remote Compilation and Execution on Grid Platforms with JaDiMa

Jesús De Oliveira
Universidad Simón Bolívar, Venezuela

Yudith Cardinale
Universidad Simón Bolívar, Venezuela

Eduardo Blanco
Universidad Simón Bolívar, Venezuela

Carlos Figueira
Universidad Simón Bolívar, Venezuela

ABSTRACT

In distributed environments (e.g., grid platform) it is common to find pieces of reusable code distributed among multiple sites. The possibilities of compilation and execution with remote libraries have a great potential to facilitate the integration of pieces of software developed among different organizations. This chapter describes JaDiMa (Java Distributed Machine), a collaborative framework to construct Java applications on grid platforms. JaDiMa automatically manages library repositories to allow users to compile and execute applications which use distributed libraries, without keeping these libraries locally. JaDiMa services are implemented as Web Services following the SOA approach; library repositories are modeled as a JXTA P2P network; and semantic annotations of libraries assist developers on the tasks of discovering libraries. We describe an implementation of JaDiMa as part of SUMA/G, a Globus-based grid environment. We show experiences and an empirical evaluation of JaDiMa execution and compilation processes for an application which uses remote libraries for managing graph and network data.

DOI: 10.4018/978-1-61520-973-6.ch010
INTRODUCTION

Grid platforms increase the capabilities of environments in which multiple users, geographically distant, may share data, pieces of software, computation resources, and even specialized devices (Berman, Fox, & Hey, 2003; Abbas, 2004). In this direction, it is very common that programmers use several library components developed by third parties to achieve the global goal required by an application. Following the principle of reusability, it is much more efficient for developers to delegate specific functionalities to already available software, extensively proven and developed exclusively for such functions, and concentrate themselves on the resolution of their specific problem. Code reuse is a very good way to increase developer productivity and application maintainability.

However, when pieces of reusable code are distributed (e.g. on a grid platform), developers have to find and obtain the suitable libraries, then reference them in their applications such that the compilation process can be accomplished. In addition, end users must have the same libraries in order to be able to execute the application. In general, these applications are distributed along with libraries which they depend on. Hence, the reused pieces of software must remain local to the compilation and execution environment. In a distributed environment, this approach presents serious disadvantages: i) waste of disk space, when a library is used by several applications, or when only a small portion of this library is used; ii) difficulty in handling and updating versions of the library, by leaving in the hands of the developers and end users, the responsibility of updating its local versions with newer ones; and iii) in the case of development of scientific applications for grids, libraries are only required for local compilation, since the application is not going to be executed locally but in some of the remote execution nodes of the grid platform. Downloading the libraries locally only for compilation represents a waste of space and time for developers.

The idea of exploiting resources belonging to different companies and research institutes to solve problems in science and commerce domains is promoting new business models for service providers that lease computing utilities (Vassiliadis et al., 2004). This requires an integration of market-based economic theories in distributed resource management systems to create a trusted and profit environment for competitive customer (Menascé & Casalicchio, 2004). The possibilities of compilation with remote libraries and remote execution have a great potential to facilitate the integration of applications and pieces of software (libraries) developed among different organizations.

A Service-Oriented Architecture (SOA) (Erl, 2005) is essentially a collection of loosely coupled services that inter-operate according to a formal specification, independent of the platform and the programming language used to implement each service. SOA relies on a set of policies, practices, and frameworks by means of which it ensures that right services are provided and consumed, enhancing services inter-connectivity. Flexible SOA connections between services, as well as software components that deliver simple data among other services or that coordinate simple activities, are seen as services which can be combined with other services to achieve specific goals. Thus, SOA provides a scalable and robust framework to integrate heterogeneous software agents and enhance reliability of isolated software components.

Motivated by these considerations, we focused on developing a Java-based framework, called JaDiMa (Java Distributed Machine, http://www.sourceforge.net/projects/jadima), a collaborative platform to construct high performance Java applications on grid platforms. JaDiMa is a system that automatically manages remote library repositories used in a Java application. JaDiMa allows users to compile and execute Java applications using those distributed libraries, without the need of keeping them in the developer and user hosts. It leverages