Implementation of AMGA Web Application using AMGA Manager based-on RCP

Taesang Huh¹, Geunchul Park ¹, Jae-Hyuck Kwak¹, Soonwook Hwang¹, Sunil Ahn¹

¹Department of Supercomputing Center, Korea Institute of Science and Technology Information, Korea
{tshuh, gcpark, jhkwa, hwang, siahn}@kisti.re.kr

Abstract. As the size and complexity of the project increases, multiple users in a distributed environment jointly analyze the large-scale data. Metadata service is considered to be one of integral parts of Data Grids, where it is common to see millions of files spread all over the geographically distributed environments. ARDA Metadata Catalog Grid Application (AMGA), one of the most popular and widely used metadata catalogs in recent e-Science environments. Consequently, we describe the implementation of the AMGA web application focusing on the transformation of AMGA Manager using Eclipse Rich Client Platform (RCP) to a Rich Ajax Platform (RAP) based web application in order to allow users to access AMGA service with common web browser.

Keywords: AMGA, Metadata catalogue, AMGA Manager, Eclipse, RCP, RAP, Grid computing

1. Introduction

This paper focuses on the development of a general-purpose AMGA web application that provides a powerful Eclipse framework allowing engineers and programmers to quickly and easily embed data and metadata features inside their own applications, using a standard design-pattern-based approach. AMGA has powerful functionalities to ensure good performance and scalability, along with a multi-threaded multi-process based on Database (DB) connection pooling, a hierarchical collection structure, replication, and a federation mechanism for an efficient distributed environment. There are many AMGA user communities worldwide: Belle II, INDICATE, DECIDE, DKRZ, EUMEDGRID. [1-4] In the past, the interface is clumsy, and has limited login authentication for uploading of the proxy file generated by a pre-procedure of another User Interface (UI) machine. Moreover, there has been no professional Client/Server (C/S) software offered for efficient management of all AMGA contents. [5-6]
In response to these problems, we determined the development and implementation procedure for a general-purpose and intuitive AMGA GUI toolkit from a C/S program based on transformation of RCP, which normally runs as a desktop application on a personal computer, to a RAP-based web application that can be deployed in other runtime environments with limited manpower. The result was AMGA GUI client (AMGA Manager), an easy-to-use and general-purpose GUI toolkit for AMGA. [2] On top of this framework, we built a web application that allows users to manage the metadata catalog and administrators to control AMGA services (setting the configurations in amgad file, and start/stop/restart service). Also, we developed an AMGA web application for access to and management of the metadata catalog from any platform. Users need only a web browser and their valid authentications: Virtual Organization Membership Service (VOMS), Grid Security Infrastructure (GSI), Identifier/Password (ID/PW) and certificates. [7]

3. Design and Implementation

3.1 Design

The design AMGA web application shares the AMGA Manager design, because this development is to be transformed mainly in consideration of multi-users and RAP plug-ins, according to the concept of reuse packages developed by Eclipse RCP (according to the RAP project, 70% - 90% reuse is possible). [8-9] But, all features of AMGA Manager are imported into the web application along with these added administrator functionalities: controlling the AMGA configuration and the behavior of AMGA services. In Figure 1, multi-users access AMGA web service based on Grid environment as well as existing environment with their valid authentications using only web browser.
3.2 Architecture

The architecture of AMGA web application is shown in Figure 2. AMGA web application interacts with AMGA server through a set of AMGA Java API and AMGA configuration and service control API after authentication/authorization in the connection module. The AMGA configuration and service control API allows to control the service and to change the service parameters at AMGA server. The RCP source files of collection Management, metadata management, properties management and SQL editor modules are reused and service management is made of new RAP coding. Then all RCP components are changed to a web service at transformation layer: RAP-based AMGA web application.

3.3 Development Environment

Development environment of AMGA web application consists of following three categories:

(1) Server environment:
Sever environment has main role to provide web service relating to AMGA service.

(2) Client development environment:
Client development environment allows developers to transform RCP, AMGA Manager to RAP, develop the components of web service and test the function level in the AMGA web application.

(3) Client testing environment:
Client testing environment is used only for the purpose to test web service running.

Fig. 1. AMGA web service
The development environment, including the test and debug environments, contains an SLC5 OS-based tomcat servlet container interacting with AMGA (ODBC: psqlodbc-08.03.0200, Backend DB: Postgresql). We used Eclipse for RCP and RAP developers (a development toolkit known as Integrated Development Environment (IDE)) in the coding, debugging and testing phases in order to accelerate the overall development process. As a prerequisite, JDK 1.7 was installed on the server side as well as client side. On the client side, the web application was tested in several web browsers (Opera, Chrome, Internet Explorer, and Firefox) working on various OS platforms (Mac, Debian, and Windows).

3.4 Implementation

We first changed all of the RCP dependencies to RAP versions and performed debugging so as to convert the RCP source files to RAP ones since all RCP APIs were not available in RAP and some RCP APIs, such as Graphic Context (GC), StyledText, FileDialog and MouseMove Events, caused many errors in case of applying RCP APIs to RAP. For example, Text was applied to RAP instead of StyleText, which isn’t supported by RAP, and needed additional code modification. Also, we considered UI libraries and changed to the org.eclipse.ui plug-in for RAP because RCP and RAP needed different platforms. Session-based singleton was used for a web application to provide multi-users simultaneous access. We also developed add-on interfaces to handle the AMGA configurations and services and show the list of AMGA sites.

Fig. 2. After login, main web page
As illustrated in Figure 3, after a successful login with proper authentication, users will be able to browse AMGA collections hierarchically according to the specific metadata schema with its entries and attributes. At this time, the most of the code is being executed on the server side, whereas the thin-client side, users’ PC running the web browser, is only updated when needed.

4. Conclusions

In this paper, we have discussed implementation of a general-purpose AMGA web application through transformation of the RCP-based AMGA Manager to a RAP-based web service. A powerful RAP enables to quickly and easily embed data and metadata features inside their own web service. The AMGA web application provides the basic manipulation functionalities of AMGA Manager as well as administrator functionality for manipulation of AMGA configurations and handling AMGA services. After a successful login, users are able to browse the hierarchy of AMGA collections, to inquire about their schema, permissions and entry list. Users also have the ability to manipulate collections, their metadata schema, entries, access control, user/group information, federation, plain table, export/import metadata files, service configuration, service behavior and list of AMGA sites; all of this is accomplished via a user-friendly web interface that removes complexities to enable easier accessing Grid services, encapsulates AMGA syntaxes, and provides SQL editor for automatic query composition, various wizards, and specialized viewers.

Acknowledgements

This work has been partially funded by the European Commission as part of the EMI(Grant Agreement INFSO-RI-261611)

References

9. B. Muskalla, Patterns for Single-Sourcing RCP and RAP applications, EclipseSource version 1.1.4