Contemporary Engineering Sciences, Vol. 7, 2014, no. 14, 713 - 718
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2014.4679

Design and Implementation of AMGA Metadata Catalog System for Interoperability in European Middleware Initiative

Geunchul Park, Soonwook Hwang, Jik-Soo Kim, Jae-Hyuck Kwak and Taesang Huh*

KISTI, Korea Institute of Science and Technology Information
245 Daehak-ro, Yuseong-gu, Daejeon, Korea 305-806
*Corresponding author

Abstract

AMGA (ARDA Metadata Grid Application) is a grid metadata catalog system, that is part of the gLite middleware, developed for the EGEE (Enabling Grids for E-sciencE) project. Since the end of EGEE, gLite has participated in the European Middleware Initiative (EMI) with other major middleware systems such as ARC, dCache and UNICORE. Through AMGA's three-year EMI collaboration, it has been actively developed, tested and deployed. Previously, AMGA had made progress by focusing on development for the purposes of function implementation and improvement as a metadata catalogue system. After participating in the EMI, it has designed and implemented middleware-interoperability functions according to EMI policies and strategies.

Keywords: AMGA, Grid, Metadata, EMI, Interoperability

1 Introduction

AMGA (ARDA Metadata Grid Application), a grid metadata catalogue service, has been developing, through the EGEE project, functions needed by research communities in various fields. AMGA was developed based on the requirements of the HEP (High Energy Physics) group and the Biomed group, its

two key community groups. HEP was seeking to efficiently manage millions of files and the related metadata; Biomed, using fewer files and metadata than HEP, was instead interested in higher-level security and ACL (Access Control List). In meeting these dual requirements, AMGA realized metadata interfaces that support a flexible, dynamic and hierarchical schema. In order to save metadata, it uses a variety of RDMS such as Oracle, PostgreSQL, MySQL, and SQLite. It also offers security functions including X.509-based certificates, VOMS support, Grid-proxy certificates, and detailed ACL. [1][2][4]

The European Middleware Initiative (EMI), which is the result of the collaboration of four major middleware systems in Europe, aims to expand and improve interoperability and deployment over a massive distributed infrastructure, and, at the same time, to construct a model for the future maintenance and evolution of middleware components. In order to register in the EMI repository, a candidate must comply with EMI development policy. It also should comply with a strict quality assurance process and respond to the EMI's common strategy. The core focus of the EMI strategy is the improvement of usability, compatibility and manageability, and the securing of long-term sustainability. Another central EMI priority is the enhancement of the interoperability between EMI-affiliated products. [2][3]

After participating in the EMI, AMGA began to develop functions to facilitate co-utilization with other EMI-affiliated products. In the development process, a common library was established for distribution of software via the EMI Repository. For these purposes, various operating systems recommended by EMI were provided. ETICS was used for the purposes of software development and testing. Our software, AMGA, was released in accordance with the integrated Packaging policy. In addition, a function for the purposes of service application and monitoring was developed, and an integrated security function was achieved. This paper summarizes the main issues and results related to development of improved interoperability in the EMI as a result of its collaboration with AMGA. [2][5]

2 EMI development system

There are a few requirements for software co-utilization and interoperability. Each software program must support an equivalent operating system and an equivalent type of package, and there must be a common library in use. In addition, the test must be carried out in an identical environment. In order to satisfy these conditions, all EMI-affiliated software has been developed through ETICS (eInfrastructure for Testing, Integration, and Configuration of Software). [2][3][5]

ETICS is an on-line collaborative project-development management service that oversees the setup of configurations, enforces quality standards, and builds packages and tests them in environments as close as possible to real-world infra-

structures. AMGA, for example, was developed using ETICS. Through ETICS, EMI products and libraries including the dependency needed for building and executing AMGA were used, so that AMGA could be compatible with other EMI products. By using ETICS moreover, all collaborating EMI-affiliated products are able to maintain equal dependency. Maintaining equal dependency is extremely important, as EMI is distributed through a repository. Also absolutely necessary is an ETICS-integrated development environment developed by cross-reference between EMI products. ETICS is also able to provide a function for generating various development-related reports. Successfully built software is stored in a temporary repository so that it is ready for immediate use in testing. [2][5]

It has been suggested that in the third phase, where EMI comes to an end, each software program should adopt an independent packaging policy. The shift in building method will be from the use of source code with ETICS to the use of source RPM, which is designed to generate the package without ETICS. It also improved the completeness of the package by fixing the RPMLint error for EPEL. For Redhat LINUX, AMGA also used MOCK in packaging, and for Debain, it used pBuilder in packaging. [2]

For software co-utilization and interoperability, all of the aforementioned methods are vital. However, as a tool such as ETICS requires much in the way of system resources as well as continuous management by the administrator, it is difficult to apply to small-scale projects. In these cases, identical Virtual Machine Setting is used in testing, and compatibility can be secured through methods including mutual exchange of Packaging information. In any case however, the Library Dependencies must be maintained, simply to meet the minimum requirements for interoperability.

3 Information index service and service monitoring

One of the necessities governing the application of Middleware such as AMGA is to understand the potential uses of the applicable software or resource. Information sharing is necessary in regard to where software desired for use is to be run, what the installed version is, and whether it is currently available. The administrator responsible for managing the service, moreover, must be able to monitor whether the applicable service is carrying out normal operations. Accordingly, AMGA has implemented a function that publishes service information on BDII and EMIR. For such monitoring purposes, the Nagios Probe has been developed. [2][3]

In EMI-2, publishing version information, introduced to facilitate and enhance resource co-utilization, refers to the posting of information regarding the status and resources of EMI-affiliated services. Each of the services provides related data (according to the product version and EMI distribution version) to BDII (Berkeley Database Information Index) through the GLUE2 schema. In order to accomplish this, a script for the purposes of generating related data as schema is

necessary. In the gLite group, to which AMGA belongs, these information-generating scripts are managed as a whole as a glite-info-provider-service package. AMGA also uses this package to generate the GLUE2 schema. Using the mdservice script provided by AMGA allows for easy and simple publishing of AMGA service information to BDII. Executing the mdservice script installs the glite-info-provider-service package using yum, and using a file from the package, it creates a script file. This script file communicates with the AMGA Server on a regular basis, and transmits status information in the form of LDIF to BDII. [2]

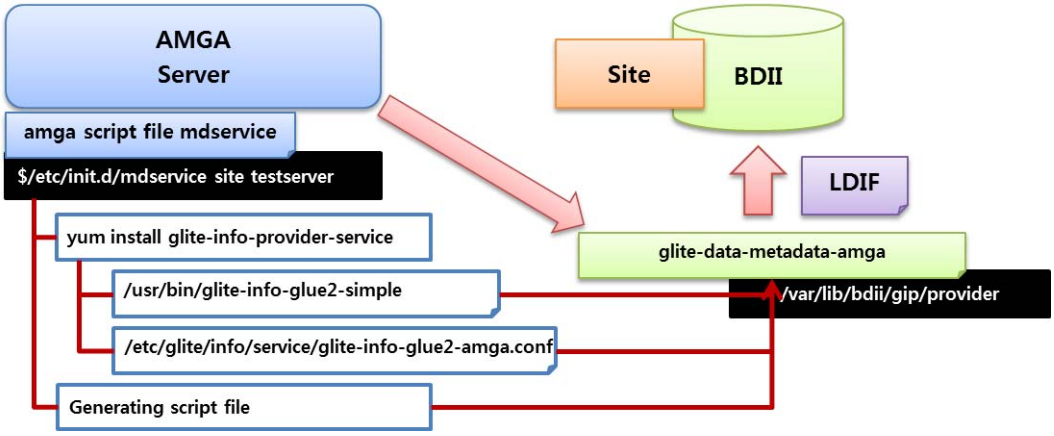

Fig. 1 Publishing service information of AMGA to BDII

EMI-3 incorporates a function for registering AMGA service information to the EMIR, EMI's common service registry. This is similar to EMI-2's method of providing data to BDII, except that in the case of the refined EMI-3, the contents of the provided data are redesigned for EMI service. In order to provide AMGA data to the EMIR, AMGA service-related data based on GLUE2.0 was designed in the form of a function that provides this to EMIR DSR (Domain Service Registry) through EMIR-SERP (Service Publisher). In order to provide information through EMR-SERP, a function for creation of a Jason file was installed. [2][3]

AMGA developed a probe for Nagios, the monitoring tool widely used for enhancement of usability. In accordance with the request from EGI, the EMI recommended that each product should provide a plug-in for this probe, which many people use. Accordingly, AMGA also manufactured and produced the related plug-in. For ease of utilization and modification, the AMGA Nagios plug-in was written using Perl, by which the status of the AMGA server (e.g. availability) can be monitored. Fundamentally, the Nagios plug-in and the AMGA server use the TCP/IP socket to communicate. When the Nagios plug-in accesses the AMGA server and transmits 'statistics\n\n', the AMGA server returns brief information on the running state of the server to the plug-in. The Nagios plug-in

analyzes this information and, in accordance with the plug-in rule, sends the corresponding status information to Nagio, which displays the status information on the monitoring page. [2][3]

4 Security

EMI security policy is carried out to serve two main purposes. One is to satisfy the needed security element through interlocking of software within the EMI, and the other is to utilize the caNl (Common Authentication Library), which is the common certification library provided by EMI. [2][3]

For improvement of security, AMGA proceeded with integration to EMI's security product Argus. The Argus authorization service, an attribute-based authorization system, is designed to answer questions such as "Can user X perform action Y with resource Z at this time?" The integration of AMGA with Argus was planned for the purposes of using the Banning List of Argus to access AMGA's permission function related to the Resource. When a user accesses AMGA through AMGA Client the AMGA server checks for the possible existence of the user on the banning list by accessing the Argus PEP server through PEP client API. If the user exists on the banning List, he is blocked from access. [2][3]

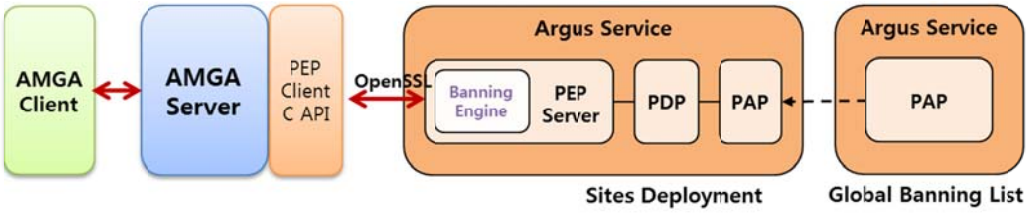

Fig. 2 Designing of integration with Argus

The EMI, for the purposes of integrated authorization between EMI products in the 3rd year and, thereby, increasing their integrated utilization, provides the caNl library. In order to support most of the EMI products, this is provided as a Java, C and C++ library, and was designed to be applied to AMGA during socket generation. The caN1 library currently is being implemented. [2][3]

5 Conclusion

This paper summarizes the main issues and results relevant to the development of increased and enhanced interoperability in the EMI, which is a middleware co-utilization project of AMGA. A different approach from past ones, which entailed only simple functionalization, was needed in order to increase interoperability between software. Much effort was required in securing both intercompatibility and software co-utilization. We carefully considered the necessary relevant functions, and created those that we considered to be vital. This

improved manageability through integration with BDII and EMIR and implementation of the Nagios probe. All of this also improved compatibility, through support of various platforms, the caNl library application, and integration with other middleware.

Three years of development experience at the EMI represents a major turning point in the development of AMGA. Through the EMI, it was possible to effectively respond to the requests and satisfy the requirements of large communities such as EGI and WLCG. Through efforts at creating connections with EMI's other products, it was possible to improve compatibility. Through this kind of EMI collaboration, know-how about strict quality assurance systems and systematic development processes can be gained. The most significant benefit has been the experience of defining and creating the vital functions that are essential for middleware interoperability. Undoubtedly, this experience will greatly facilitate the development of new middleware in the future.

References

[1] S. Ahn, N. Kim, S. Lee, D. Nam, S. Hwang, B. Koblitz, V. Breton and S. Han, Performance analysis and optimization of AMGA for the large-scale virtual screening, *Software: Practice and Experience*, John Wiley & Sons, Inc., **39** (2009), 1055-1072.

[2] C. Aiftimiei, A. Aimar, A. Ceccanti, M. Cecchi, A. Di Meglio, F. Estrella, P. Fuhrmam, E. Giorgio, B. Konya, L. Field, J.K. Nilsen, M. Riedel and J. White, Towards next generations of software for distributed infrastructures: the European Middleware Initiative, E-Science (e-Science), 2012 IEEE 8th International Conference on. IEEE, (2012) October 8-12; Chicago, USA

[3] T. Huh, J.H. Kwak, S. Hwang and G. Park, Collaborative EMI Development Experience with AMGA, *International Journal of Software Engineering and Its Applications*, **8** (2014), 157-170.

[4] B. Koblitz, N. Santos, and V. Pose, The AMGA Metadata Service, *Journal of Grid Computing*, **6** (2008), 61–76.

[5] A. D. Meglio, M-E. Begin, P. Couvares, E. Ronchieri and E. Takacs, ETICS: the international software engineering service for the grid, *Journal of Physics: Conference Series*, **119**(4) (2008).

Received: May 1, 2014