Cross-Domain Middlewares Interoperability for Distributed Aircraft Design Optimization

Yongjian Wang[1], D'Ippolito Roberto[2], Mike Boniface[4], Depei Qian[1], Degang Cui[3], Jiyun Jiang[2]
[1] Sino-German Joint Software Institute, Beihang University, Beijing, China
[2] NOESIS Solutions N.V., Interleuvenlaan 68, B-3001, Leuven, Belgium
[3] Science Technical Committee, Aviation Industries of China, Beijing, China
[4] IT Innovation Centre, University of Southampton
yongjian.wang@jsi.buaa.edu.cn

Abstract---The Bridge project is an EU FP6 project funded by the European Commission to support the EU-China joint effort on secure and distributed cooperation between European and Chinese industrial communities such as the distributed aircraft design optimization etc. The interoperability between SIMDAT GRIA and CNGrid GOS, which enables a joint grid platform between SIMDAT and CNGRID infrastructures, is fundamental to achieve this goal.
This paper presents the interoperability solution adopted by the Bridge project, including job management, data sharing, and authentication & authorization. A prototype of distributed aircraft design optimization based on the interoperability between GRIA and GOS has been implemented, which can effectively aggregate and integrate different analysis services that are provided by different geographical distributed partners. The strategies adopted in implementing the prototype are discussed and preliminary results are presented.

Keywords: Interoperability, Distributed Design Optimization

I. INTRODUCTION

Analysis services, such as mesh generation tools, structural prediction tools and Meta-Modeling services are the basis of virtual product development in many different industries. Such analysis services are, to an increasing extent, being integrated into complex problem solving environments to allow engineers to easily drive the whole development process in an integrated environment. To meet the challenges of geographically and logically distributed development processes, analysis services have to be Grid-enabled and integrated into Grid-enabled problem solving environments.

In general, Grid technologies [1] [2] have evolved to fulfill two major aspects. First, distributed resources are to be virtualized to provide a single consistent view to the end-user and allow him/her to use these resources without having to worry about infrastructure or business aspect details. Second, Grid technologies are based on open standards. Both aspects are essential analysis services requirements for distributed companies and their suppliers that are working in multiple product development disciplines and want to share analysis services based on open standards. In this view, Grid technologies are the only viable alternative to fulfill these requirements.

In order to address these challenges, the European Research project "BRIDGE" [3] has been set up. This project extends the interoperability of underlying Grid infrastructures (SIMDAT [4] and CNGrid [5]) through the development of enhanced specification-based services and *gateway* technology to support secure and distributed cooperation between European and Chinese industrial communities.

Based on the experiences from SIMDAT and CNGrid, the Bridge project has a special focus on industrial applications. In particular, this paper focuses on the aerospace application developed during the Bridge project and demonstrates the use of analysis services based on GRIA and GOS, workflow definition and execution based on GRIA as applied to the optimization of the topology of an airplane wing structure. To achieve this goal, cross domain grid infrastructures interoperability between European and China is therefore needed.

The paper is organized as follows: section 2 describes distributed design optimization; section 3 introduces system architecture of aerospace application; section 4 explains cross-domain analysis services coordination based on the middlewares interoperability between GRIA and GOS; section 5 gives a preliminary result of our work; finally, in section 6, the conclusions and future works are described.

II. DISTRIBUTED DESIGN OPTIMIZATION

In this paper, the flap position for an aircraft wing is optimized to reduce noise during landing without losing lift. The wing and body geometry models are

978-0-7695-3535-7/08 $25.00 © 2008 IEEE
DOI 10.1109/eScience.2008.176

created by using *Dassault CATIAv5*. The wing and body models have been taken from a civil transportation aircraft; these models have been used to create aerodynamic, acoustic and aeroelastic models for analysis.

The generated geometry has been modified according to the optimization needs. Therefore, an automatic model creation tool has been implemented to generate the parameterized model according to geometry modifications. The design parameters are the *x*, *y* and *theta* position of each flap, as shown in Figure 1.

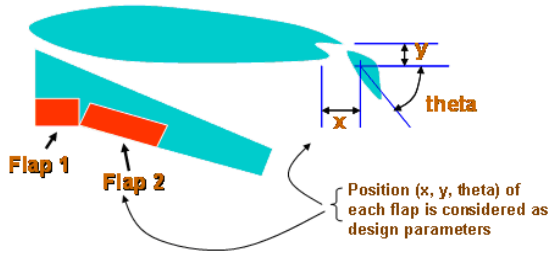

Figure 1: Design Parameters

In order to achieve the noise reduction objective, a complex simulation process has been set up, including a number of different analyses, each provided by a partner of the Bridge project consortium. The simulation process includes a geometry model generator, an aerodynamic simulation service, an aero-elastic simulation service (provided by AVIC-II, China), a Meta-Modeling service (provided by FhG-SCAI, German) and an acoustic simulation service (provided by EADS, France). An outline of the logical integration of these disciplines is shown in Figure 2.

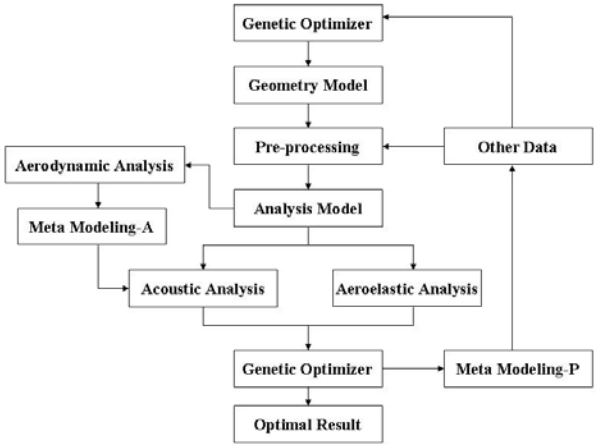

Figure 2: Aerospace Optimization Scenario

A dedicated genetic algorithm (provided by AVIC-II, China) has been applied to carry out the optimization process. Meta-Modeling technology has been used to reduce the number of needed evaluations in order to accelerate the optimization process. The workflow service (provided by LMS, Belgium) captured the above simulation process as well as the optimizer with a grid enabled workflow and completed the optimization loop.

The simulation structure described above is a representative multi-disciplinary analysis case for the aeronautic industry, and the normal analysis procedures and optimization processes cannot be used to achieve an optimal result in a reasonable timeframe.

III. System Architecture

The aerospace application has been developed by AVIC-II, Beihang University and LMS International. The application development can be classified into three different parts:

1. Development of the automatic model creation tools.
2. Creation of GRIA and GOS based analysis services;
3. Development of the aerospace application analysis workflow (called AGrid hereafter);

In Figure 3, the system architecture of the aerospace application is outlined.

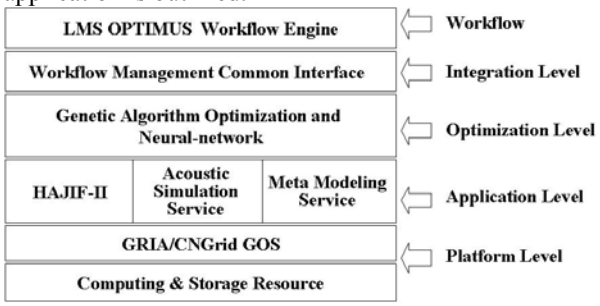

Figure 3: System Architecture

It can be divided into five separate layers:

- Platform Level: it's the basic level and consists of underlying computing & storage resources and GRIA/ GOS;
- Application Level: it consists of analysis services used in aerospace application, including HAJIF-II, acoustic simulation service and Meta-Modeling services etc;
- Optimization Level: it's the core of aerospace application, consists of Genetic Algorithm Optimization (GAO for short hereafter) and Neural-network;
- Integration Level: it provides unified interface to underlying applications and resources, we support both GRIA and GOS;
- Workflow: it in charge of integrating distributed analysis services to form a functional aerospace application.

GAO is the core of AGrid; OPTIMUS is used to integrate and drive the simulation process, which is defined as a multi-level workflow. The analysis

services in the optimization process for aerospace application are assigned to geographically distributed clusters. OPTIMUS can integrate grid related services by executing batch file, which submits jobs to underlying grid middleware. The analysis applications in the process of GAO can be launched from a batch file, which submits jobs to underlying grid middlewares.

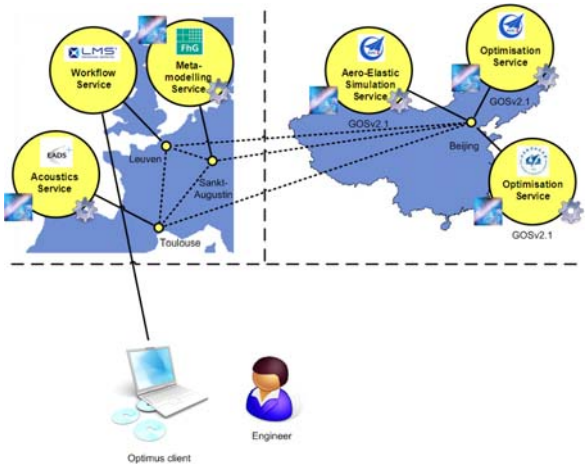

Figure 4: Aerospace Application Deployment

Figure 4 shows the deployment of analysis services required in the aerospace application across China and Europe. In this deployment, LMS OPTIMUS workflow drives the scenario, orchestrating acoustic and Meta-Modeling services in Europe, as well as aeroelastic and optimization services in China. The relationships among the different modules in the runtime are shown in Figure 5.

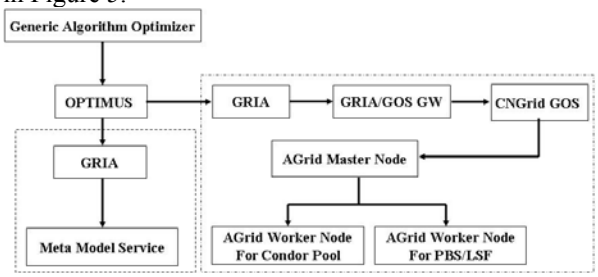

Figure 5: Relationships among Different Modules

As Figure 5 shown, analysis services deployed in Europe are integrated into the main workflow directly through GRIA, but the services deployed in CNGrid are integrated through the GRIA/GOS gateway which will be introduce in section 4.2.

IV. Cross-Domain Interoperability

To implement distributed design optimization, we need to resolve cross-domain service coordination. Many things need to be considered during this process. In this section, we will introduce the work done.

A. *Multiple Level Distributed Aircraft Design Optimization*

The simulation services participate collaboratively in the overall aerospace design process. The big challenge in this scenario lies in that we need to deal with multiple level optimization design that is driven by the genetic algorithm, which is more complex than traditional optimization design.

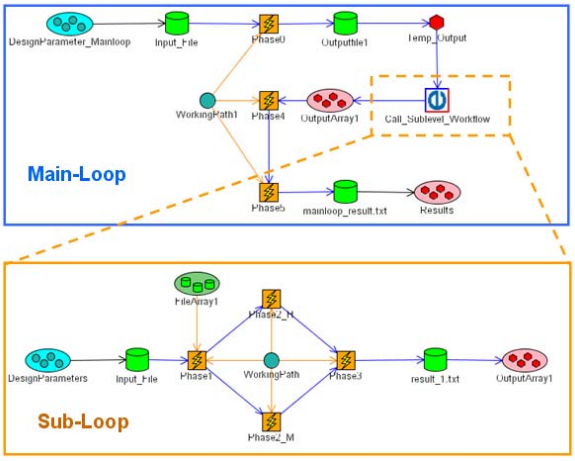

Figure 6: Multiple Level OPTIMUS Workflow

As Figure 6 show, OPTIMUS provides the main mechanisms for the execution of the grid-enabled workflow in the optimization loops. OPTIMUS is the workflow engine we used in the Bridge project. It is a generic workflow definition, capture and federation tool, allowing the creation of "simulation" based workflows. OPTIMUS executes the workflow in a distributed manner on the individual grid infrastructure of China and Europe. It integrates the aerodynamic service, aeroelastic service, acoustic service, Meta-Modelling service and genetic algorithm optimiser to form the complete optimization application.

B. *GRIA/GOS Interoperability*

It's important to implement interoperability in order to support effectively coordination among the analysis services. Achieving interoperability among heterogeneous grids is not a trivial task [6] [7] [8]. In order to fulfill the interoperability requirements of Bridge project, three problems must be addressed:

1. Authentication & Authorization;
2. Cross-Domain Job Management;
3. Efficient Cross-Domain Data Share.

1) *Authentication & Authorization*

Authentication & authorization are two basic pillars of the security mechanism. Authentication is used to determine the identity of a security principal, whereas authorization is used to determine whether a known

security principal is permitted to perform a requested action on a resource.

The security mechanisms of GRIA and GOS are compared in Table 1

Table 1 Difference in Security Mechanisms

Middleware	Authentication	Authorization
GOS	WS-Security	SAML Token
GRIA	WS-Security	PBAC

WS-Security is supported by both GRIA and GOS, so the authentication problem can be resolved through using X.509 certificates and the difficulties lie in authorization.

GOS adopts SAML token as the authorization solution which utilizes assertions and per-define rules to decide whether an operation is allowed or not. While GRIA provides a dynamic access control mechanism called PBAC that enforces access control policies in respect to business processes for interacting with stateful resources. Figure 7 shows a typical PBAC system.

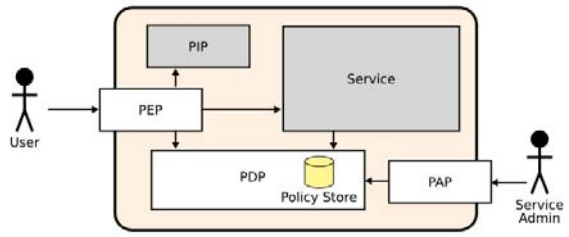

Figure 7: PBAC Architecture

There are many differences between SAML token and PBAC, a simple and unified mechanism is needed to express the access control requirements and make the decisions. In the Bridge project, GOS was extended to support the policy management operations defined in the security profile of NextGRID project [9]. Support for this profile was added to GRIA in the NextGRID project itself. The core of the policy management operations is the PolicyRule type, which is used to carry the information. It consists of three parts:

1. MatchPattern: A match pattern can be applied to a set of credentials presented by a subject to determine whether the subject matches the pattern or not;
2. Roles: Each type of resource has a set of roles which a subject may have. Each policy rule affects membership of only a single role, given by the role element. Unlike the tokens matched by the match patterns, which are chosen by and meaningful only to the client, roles must be meaningful to both client and service. The set of available roles for a resource is defined by the instance.
3. Type: Three types of rule are defined including NECESSARY，SUFFICIENT，DENY

Two examples of PolicyRules follow the first matching an X.509 identity and the second matching a SAML assertion:

```
<rule>
  <matchPattern>
      <subjectDN>CN=Beihang</ subjectDN>
      <issuerCertificate>q8qqhf</issuerCertificate>
  </matchPattern>
  <type>SUFFICIENT</type>
  <role>reader</role>
</rule>
<rule>
  <matchPattern>
      <issuerCertificate>q8qqhf</issuerCertificate>
      <attributeName>project</attributeName>
      <attributeValue>Bridge</attributeValue>
  </matchPattern>
  <type>SUFFICIENT</type>
  <role>writer</role>
</rule>
```

PolicyRule contains all the necessary information for access control operation. GRIA and GOS can make their own decisions based on them.

2) Gateway-Based Job Management

In the current gateway-based solution, GRIA acts as a front interface for GOS and it transparently forwards received jobs to CNGrid for processing.

The GRIA job service is used to manage jobs, but it does not access underlying resource mangers directly to submit and check jobs. Instead, it introduces an extra layer of underlying resource manager dependant platform scripts to submit and check jobs respectively. For each resource manager, GRIA requires a separate suite of platform scripts to handle jobs such as:

- Start job script to submit jobs;
- Check job script to check the status of a job;
- Kill job script to terminate a job.

The job service then can be configured to use different platform scripts suitable for corresponding underlying resource manager. These platform scripts know how to handle (start, check, kill) jobs for that particular resource manager, and can be instructed to run a particular application via its application wrapper. At present, three different kinds of platform scripts are shipped with GRIA by default to support Condor, TorquePBS and Local Execution.

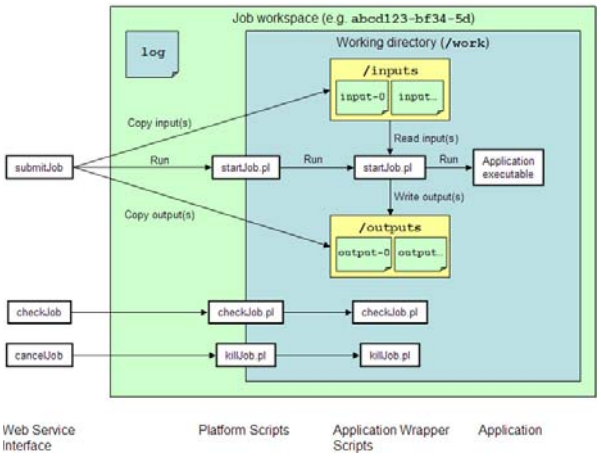

Figure 8: GRIA Job Service Interfaces and Scripts

Figure 8 illustrates how the platform script layer sits between the job service and application wrappers hiding resource manager details.

Table 2 compares the job description languages and submission manners adopted by GRIA and GOS.

Table 2: Job Management Mechanisms

Middleware	Job Description Language	Submission Manner
GOS	JSDL	Portal, Web Service
GRIA	JSDL, created using Application metadata	Web Service, Client Toolkit

Because of the differences between job management mechanisms, two issues must be handled during cross-domain job management:

1. Job Description Language;
2. Job Submission Manner.

Job description language is used to define job requirements, its major elements includes: command definition, resource requirement definition and data staging definition. Different syntax is used to describe job requirements in different job description languages, a conversion must be performed between the JSDL [10] dialects used by GOS and GRIA by the gateway during job processing.

Table 3: Job Description Languages Mapping

Job Description Language	GOS	GRIA
Application	JobDefinition/Application/ POSIXApplication/ Executable	JobDefinition/Application/ApplicationName
Argument	JobDefinition/Application/ POSIXApplication/ Arguments	JobDefinition/Application/ POSIXApplication/ Arguments
Data Transfer	JobDefinition/DataStaging	JobDefinition/DataStaging EPR

Table 3 shows a mapping of major elements between GOS and GRIA JSDL dialects.

We implement a GOS client toolkit which encapsulates all the necessary conversions including JSDL creation, job submission, and job status checking etc. The job submission and status checking operations are done through the Web Services interface provided by GOS which complies with OGSA-BES. The GOS client toolkit is packaged as the application wrapper which will be launched by the Local Execution platform scripts shipped with GRIA. We also extend the GOS data transfer mechanism to support direct interaction with GRIA Data Service using HTTPs protocol.

The whole process of gateway-based interoperability between GRIA and GOS is illustrated in Figure 9 and the steps can be summarized as:

1. The end user firstly uploads necessary input data to the GRIA Data Service using the HTTPs protocol, then invoke the target application, which is published through the GRIA Job Service;
2. The GRIA Job Service invokes GRIA/GOS wrapper through Local Execution platform scripts;
3. The GRIA/GOS wrapper modifies the access control rules of data uploaded in step 1 to make sure GOS can access it, then invokes GOS using the GOS client toolkit;
4. GOS downloads data from the GRIA Data service using the HTTPs protocol, processes the job request and uploads the result files to the GRIA Data Service;
5. The end user downloads result files from the GRIA Data Service and prepares the next service invocation.

In step 3, access permissions are modified based on the pre-configured GOS BES certificate. If users are allowed to access the GOS service directly, not just via GRIA, then it's possible that one user can access the data that does not belong to him/her because GOS may on behalf of multiple end users. To avoid this situation, we propose a dynamic security token based permission mechanism which will be introduced in section 4.2.3.

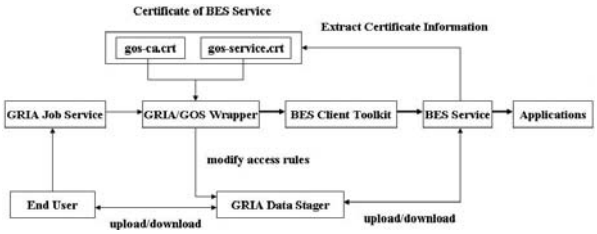

Figure 9: Gateway Based Job Management

3) Cross-Domain Data Management

In many situations, the output of one service will become the input of another service. And considerable amounts of data are generated by distributed design optimization such as CFD computation.

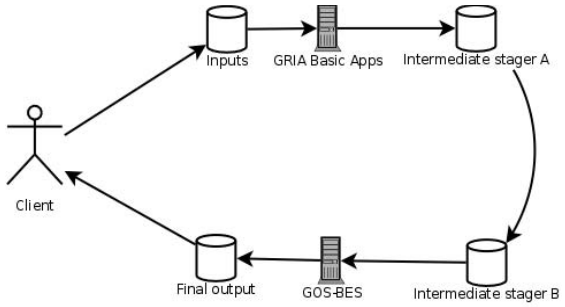

Figure 10: Cross-Domain Data Management

For efficiency reasons, data, especially large volume data, should be transferred directly from the output of one service to the input of another, rather than having the client download it from one service and upload it to the other. Here, the output of a GRIA job is used as input for a GOS-BES job as shown in Figure 10.

Table 4 compares GOS and GRIA in data management mechanism.

Table 2: Difference in Data Management

Middleware	Protocol	Authentication	Authorization
GOS	FTP	User/Pass	SAML Token
GRIA	HTTPs	TLS/SSL	PBAC

TLS/SSL [11] is the authentication mechanism of HTTPs. During handshakes phase of TLS/SSL, all the participating parties should know each other's certificates in advance. Multiple GOS installations may access the same GRIA simultaneously, as GRIA stores the required certificates in the access control rules of each individual data resource. To support flexible administration, GRIA utilizes *AnyCertProvider* provided by Shibboleth [12] which is a plug-in of Java security framework. The *AnyCertProvider* disables the simple default trust check performed by Tomcat so that authorization is enforced through PBAC instead.

Because service will perform data transfer operations for multiple users and it is important that one user cannot use the service to access another user's data without permission. Let's take the sample scenario shown in Figure 10 as an example. When a user requests that data stager B should fetch data from data stager A, stager B first invokes an extra operation on A to ensure that the client has read access to the data. If this succeeds, B invokes a second operation to actually fetch the data. But using two calls introduces a small race condition and is somewhat inefficient. For example, imagine that some user has a public data stager which anyone can read. This user now decides to upload some sensitive information into it and change the access control rules to remove public access, and then uploads the sensitive data. If a user tried to transfer the data at the same time, it is possible that the first test call would pass (while the data was still public) but that the actual data received would be the private information.

Therefore, it is desirable to perform the check in the same invocation as the transfer. There are two reasonable methods for doing this:

1. Include the identity of the client somewhere in the fetch message sent from B to A. Stager A must check if the specified client has the read access right, in additional to its usual check on the sender (B) of the message. A new custom header must be defined to contain this information.
2. Stager B creates a new identity (including a new private key) representing the service acting on behalf of the client. To stager A, this appears as a new identity (to which the client has granted read access). Stager A therefore only performs one check and no custom header is required. However, stager B must communicate with the client with the new identity.

A variation on method 2 is to have the client sign stager B's new identity. Stager A could then automatically allow B access to the data, even without an explicit policy rule. The authorization is in the new certificate signed by the client. In the Bridge project, solution 2 has been adopted, as it does not require modifications to the transfer software.

Figure 11 depicts the data access sequence diagram for the GOS service to access the output of a GRIA service directly. Security token used during data access need to uniquely identify the end user and a dynamically signed GSI proxy [13] is suitable for this purpose.

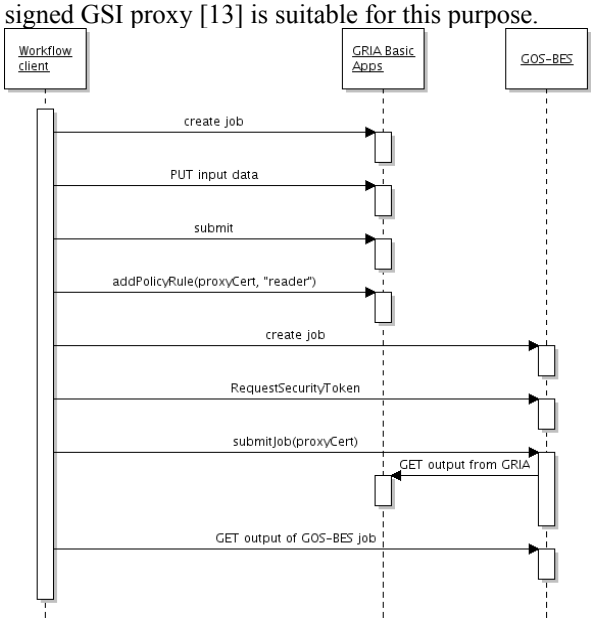

Figure 11: Data Access Sequence Diagram

V. PRELIMINARY RESULTS

A testbed installation has been deployed across China and European administrative domains as shown in Figure 12.

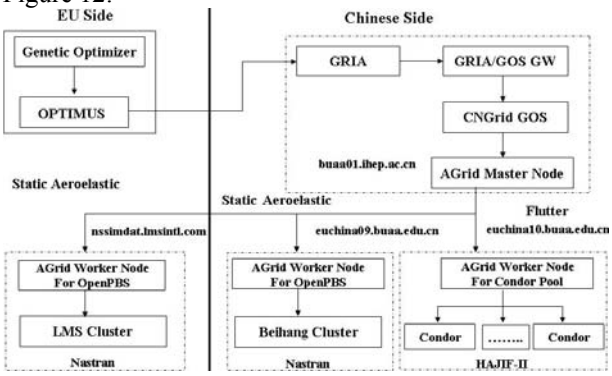

Figure 12: Testbed for Aerospace Application

It consists of the following components:

- *Optimization Client*: includes two different components:
  - *Genetic Optimizer*: the GAO optimization module of aerospace application;
  - *OPTIMUS*: workflow engine used to integrate analysis services to form a optimization process;
- *GRIA/CNGrid gateway*: includes a wrapper used to forward invocations from GRIA to GOS:
  - *GRIA*: acting as the frontend of GOS, supporting integration with OPTIMUS;
  - *GRIA/GOS GW*: acting as the wrapper of the GOS;
  - *GOS*: exposing AGrid as GOS service;
- *AGrid Environment*
  - *AGrid Master Node:* deployed as the GOS service and acts as the central control node of AGrid;
  - *AGrid Worker Node*: used to interact with underlying cluster resource managers, supporting both Condor Pool and OpenPBS;

The testbed has being running stably for more than half a year and has been successfully used for a live demo performed during the first project review meeting held in Leuven, Belgium on March 14th, 2008.

An application showcase has been executed with the testbed and the results can be observed in Figure 13. Here, average is the average fitness value of all individuals. The bestvalue can be calculated with weight sum of x1 and x2 values. The x1 and x2 represent aeroelastic and acoustic objective fitness respectively. The x1 and x2 converge to some certain value with the optimization step, which shows that the testbed can be used for the optimization in aeroelastic and acoustic problems and proved to be an effective method for solving the multi-objective multi-discipline optimization in aircraft design.

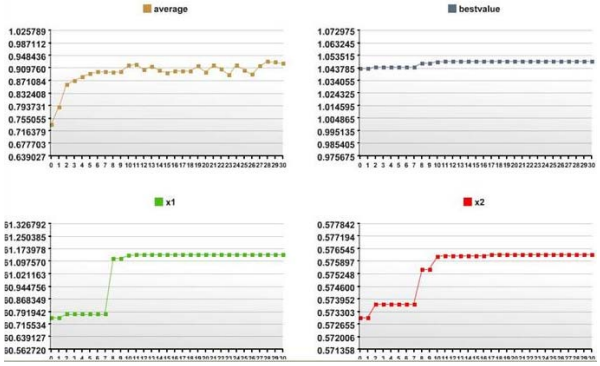

Figure 13: Test Result generated by Testbed

VI. CONCLUSIONS & FUTURE WORKS

Interconnecting the heterogeneous grid infrastructures in Europe and China, SIMDAT and CNGRID, respectively, is very important for scientific research and industrial applications which allows researchers of the two regions, Europe and China, to access computing and data resources which were not available before. This approach will certainly have a considerable positive impact to scientific research and industrial applications.

The interoperability solution proposed in this paper has proven to work well and can be considered as a valid solution to achieve transparent interoperability between GRIA and GOS. The activities to achieve higher level interoperability between GRIA and GOS will be continued. The development and the tests of the presented solution will be completed and a full deployment of our solution on the production grid infrastructures will be carried out to support the three application scenarios of the Bridge project, including aerospace, meteorology and pharmaceutical.

From the experience gained in this work, the major difficulty in middleware interoperability mainly came from the authentication & authorization aspects. This is primarily caused by different security mechanism adopted by different middleware platforms, i.e. SAML token in GOS and PBAC in GRIA. A special attention has been devoted and will be devoted in the next future to cross-domain security in order to allow interoperable access to computing and storage resources across different infrastructures.

VII. REFERENCES

[1] I. Foster and C. Kesselman. "The Grid: blueprint for a new Computing Infrastructure", Book, Ed. Morkan Kauffman, 1997.

[2] I. Foster, "The Grid: A New Infrastructure for 21st Century Science", in *Physics Today*, Vol. 55, pp. 42-27, 2002.

[3] "Bilateral Research and Industrial Development Enhancing and Integrating GRID Enabled Technologies", Bridge, , EC Sixth Framework Program, Information Society Technologies (IST), Specific Targeted Research Project Ref n. 045609, http://www.bridge-grid.eu/
[4] "Data Grids for Process and Product Development using Numerical Simulation and Knowledge Discovery", SIMDAT, EC Sixth Framework Program, Information Society Technologies (IST), Integrated Project Ref n. 511438, www.simdat.org/
[5] CNGrid project web site: http://www.cngrid.org/
[6] GIN: http://forge.ogf.org/sf/projects/gin
[7] Wang, Yongjian, Scardaci, Diego; Yan, Bingheng; Huang, Yuanqiang: Interconnect EGEE and CNGRID e-infrastructures through interoperability between gLite and GOS middleware , 3rd IEEE International Conference on e-Science and Grid Computing, p 553-560, Bangalore, India, 2007 Dec 10-13 2007
[8] Watkins, E. Rowland; McArdle, Mark; Leonard, Thomas; Surridge, Mike: Cross-middleware interoperability in distributed concurrent engineering, 3rd IEEE International Conference on e-Science and Grid Computing, p561-568, Bangalore, India, 2007 Dec 10-13 2007
[9] NextGRID project web site: http://www.nextgrid.org/
[10] JSDL (Job Submission Description Language): http://www.ogf.org/documents/GFD.56.pdf
[11] SSL/TLS:http://en.wikipedia.org/wiki/Secure_Sockets_Layer
[12] Shibboleth, open source toolkit for web single sign on, web site: http://shibboleth.internet2.edu/
[13] Grid Security Infrastructure (GSI): http://www.globus.org/toolkit/docs/4.0/security/key-index.html