



DELIVERABLE D5.2.7

FINAL CROSSGRID ARCHITECTURE AND INTEROPERABILITY REQUIREMENTS DESCRIPTION

WP5.2 CrossGrid Technical Architecture Team

Document Filename:	CG5.2-D5.2.7-v1.0-CYF104-CGArchInterop.doc
Work package:	WP5
Partner(s):	CYFRONET
Lead Partner:	CYFRONET
Config ID:	CG5.2-D5.2.7-v1.0-CYF104-CGArchInterop
Document classification:	PUBLIC

Abstract: This document presents the final architecture of the CrossGrid project, with emphasis on services and APIs that are crucial for Grid architecture definition. In addition to a general overview of the Project architecture, a detailed description of interfaces, support for interaction and compatibility with the OGSA approach are given. This description of the CrossGrid architecture should provide general orientation regarding software development and integration during the final phase of the Project. This document should be read in conjunction with the users' and installation guides for specific CrossGrid products, as a means of describing the environment in which these products are intended to operate.



Delivery Slip

	Name	Partner	Date	Signature
From	CrossGrid Technical Architecture Team	CYFRONET	24/11/2004	
Verified by	CrossGrid Internal Review Board	n/a	26/11/2004	
	Robert Pająk	CYFRONET	01/02/2005	
Approved by	Michał Turała	CYFRONET	01/02/2005	

Document Log

Version	Date	Summary of changes	Author
0.1	24/11/2004	Draft version	Marian Bubak, Katarzyna Rycerz, Maciej Malawski, Piotr Nowakowski
0.2	6/12/2004	Improvements according to information from technical brochures and user guides, workflow of interactive job submission in CrossGrid and support for interactivity	Katarzyna Rycerz
0.3	06.12.2004	Updating sections order, editing	Maciej Malawski
0.4	21.12.2004	Improvements according to developers guides	Katarzyna Rycerz
1.0	1.02.2005	Final version	Marian Bubak, Tomasz Gubała, Piotr Nowakowski, Katarzyna Rycerz
		Verified by the Quality Engineer	Robert Pająk

CONTENTS

LIST OF ACRONYMS	4
REFERENCES.....	6
1. EXECUTIVE SUMMARY.....	8
2. INTRODUCTION.....	9
3. FINAL CROSSGRID ARCHITECTURE	10
3.1. OVERALL PICTURE	10
3.1.1. <i>Subsystems</i>	10
3.1.2. <i>Dependencies</i>	11
3.2. WORKFLOW OF INTERACTIVE JOB SUBMISSION.....	12
3.3. ON-LINE INPUT CONTROL/OUTPUT MONITORING.....	12
3.4. RUNTIME STEERING IN INTERACTIVE APPLICATIONS.....	14
3.5. DETAILED DESCRIPTION OF INTERFACES.....	15
3.5.1. <i>Medical Application</i>	15
3.5.2. <i>Flood Application</i>	16
3.5.3. <i>HEP Application</i>	16
3.5.4. <i>Integration of Parallel Code for Air Quality Models into the GRID Structure</i>	16
3.5.5. <i>Wave Models</i>	16
3.5.6. <i>Data Mining – SOM</i>	17
3.5.7. <i>Grid Visualisation Kernel</i>	17
3.5.8. <i>MARMOT</i>	17
3.5.9. <i>GridBench</i>	17
3.5.10. <i>PPC Performance Prediction Tool</i>	18
3.5.11. <i>G-PM Performance Measurement Tool</i>	18
3.5.12. <i>Portal</i>	19
3.5.13. <i>Migrating Desktop</i>	19
3.5.14. <i>Roaming Access Server</i>	20
3.5.15. <i>CrossGrid Scheduler (CrossBroker)</i>	20
3.5.16. <i>Application Monitoring</i>	21
3.5.17. <i>SANTA-G</i>	21
3.5.18. <i>JIMS</i>	22
3.5.19. <i>Postprocessing</i>	22
3.5.20. <i>Optimisation of Data Access</i>	22
4. UNDERLYING MIDDLEWARE.....	23
5. CURRENT TRENDS IN GRID TECHNOLOGY	24
5.1. OGSA AND WSRF	24
5.2. GRID COMPONENTS	24
6. INTEROPERABILITY REQUIREMENTS	26
6.1. INTEROPERABILITY WITH OTHER APPLICATIONS DRIVEN PROJECTS	26
6.2. TESTBED INTEROPERABILITY.....	27
6.3. INTEROPERABILITY WITH OGSA – EVOLUTION THROUGHOUT THE PROJECT	27
6.4. INTEROPERABILITY WITH NEXT GENERATION GRIDS	28
7. SUMMARY	30

LIST OF ACRONYMS

API	Application Programming Interface
CA	Certificate Authority
CG	Shorthand for CrossGrid
CrossGrid	The EU CrossGrid Project IST-2001-32243
DataGrid	The EU DataGrid Project IST-2000-25182
EDG	Shorthand for Datagrid
GRAM	Globus Resource Allocation Manager
GSI	Grid Security Infrastructure
G-PM	Grid Performance Measurement
GT	Globus Toolkit
GUI	Graphical User Interface
GVK	Grid Visualization Kernel
HLA	High Level Architecture
HLAC	High Level Analysis Component
HPC	High Performance Computing
HTC	High Throughput Computing
JIMS	JMX-based Infrastructure Monitoring System
JMX	Java Management Extensions
JS	Job Shadow
MD	Migrating Desktop
MDS	Meta Directory System
MPI	Message Passing Interface
MPICH	Implementation of the Message Passing Interface library
OCM-G	Grid-enabled OMIS-compliant Monitoring System
OGSA	Open Grid Services Architecture
OGSI	Open Grid Service Infrastructure
OMIS	Online Monitoring Interface Specification
PMC	Performance Measurement Component
PPC	Performance Prediction Component
QoS	Quality of Service
RAS	Roaming Access Server
RGMA	Relational Grid Monitoring Architecture
RTI	Runtime Infrastructure
SANTA-G	Grid-enabled System Area Network Trace Analysis
SOAP	Simple Object Access Protocol
SOM	Self-Organizing Maps

T	Task
UDAL	Unified Data Access Layer
UIVC	User Interface and Visualization Component
UML	Unified Modelling Language
VNC	Virtual Network Computing
WP	Work Package
WSRF	Web Services Resource Framework
XML	Extensible Markup Language

REFERENCES

- [1] Bubak, M., Malawski, M., Zajac, K., “Towards the CrossGrid Architecture”, in: Kranzlmüller, D., Kascuk, P., Dongarra, J., Volkert, J. (Eds.), *Recent Advances in Parallel Virtual Machine and Message Passing Interface - 9th European PVM/MPI Users' Group Meeting Linz, Austria, September 29 - October 2, 2002*, LNCS 2474, Springer, 2002, pp. 16-24
- [2] Berman, F., Fox, G., and Hey, T. (Eds): *Grid Computing. Making the Global Infrastructure a Reality*. Wiley 2003; pp. 873-874
- [3] Foster, I., Kesselman, C., Tuecke, S. The Anatomy of the Grid. Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications* 15 (3) 200-22 (2001) <http://www.globus.org/research/papers/anatomy.pdf>
- [4] Rycerz, K., Baliś, B., Szymacha, R., Bubak, M. and P.M.A. Sloot: *Monitoring of HLA Grid Application Federates with OCM-G*, in S.J. Turner; D. Roberts and L. Wilson, editors, *Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'04)*, pp. 125-132. IEEE, Budapest, Hungary, October 2004. ISBN 3-540-22116-6
- [5] Zajac, K., Bubak, M., Malawski, M., Sloot, P: *Framework for HLA--based Interactive Simulations on the Grid*, in: *Simulation: Transactions of the Society for Modeling and Simulation International*. Special Issue: Applications of Parallel and Distributed Simulation in Industry.(accepted)
- [6] D3.5 Report on the results of the WP3 2nd and 3rd prototype–
<http://www.crossgrid.org/M24deliverables.htm>
- [7] Bubak, M., Malawski, M., Zajac, K., “Architecture of the Grid for Interactive Applications”, in: Sloot, P. M. A., et al. (Eds.), *Proceedings of Computational Science - ICCS 2003, International Conference Melbourne, Australia and St. Petersburg, Russia, June 2003*, LNCS Vol. 2657, Part I, Springer, 2003, pp. 207-213
- [8] S. R. Kohn, G. Kurfert, J. F. Painterand, and C. J. Ribbens, Divorcing Language Dependencies from a Scientific Software Library, in *Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific*
- [9] The WS-Resource Framework, <http://www.globus.org/wsrfl/>
- [10] Globus Project HomePage <http://www.globus.org>
- [11] DataGrid Project HomePage <http://www.eu-datagrid.org>
- [12] Grid Visualisation Kernel page <http://www.gup.uni-linz.ac.at/~hr/UVA+GVK/>
- [13] DataGrid WP3 website <http://hepunix.rl.ac.uk/edg/wp3/documentation/doc/api/java/>
- [14] JIRO technology <http://www.jini.org/>
- [15] JMX technology <http://java.sun.com/products/JavaManagement/>
- [16] Condor-G <http://www.cs.wisc.edu/condor/condorg>
- [17] VNC <http://www.realvnc.com/what.html>
- [18] HLA specification <http://www.sisostds.org/stdsdev/hla/>
- [19] GrADS project <http://hipersoft.cs.rice.edu/grads/>
- [20] GridLAB project <http://www.gridlab.org/>

- [21] Antonio Delgado Peris, Flavia Donno, Patricia Mendez Lorenzo, Andrea Sciaba, Simone Campana, Roberto Santinelli LCG2 User Guide <http://lwg.web.cern.ch/LCG/>
- [22] European DataGrid <http://www.eu-datagrid.org>
- [23] Network Weather Service <http://nws.cs.ucsb.edu/>
- [24] OGSA Overview, http://www.ggf.org/L_WG/News/OGSA%20Flyer_v31.pdf
- [25] Open Grid Services Architecture V1 draft document, <http://www.ggf.org/documents/Drafts/draft-ggf-ogsa-spec.pdf>
- [26] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D.: Web Services Architecture. W3C, Working Draft, 2003. <http://www.w3.org/TR/2003/WD-ws-arch-20030808/>
- [27] Next Generation Grids 2, Requirements and Options for European Grids Research 2005-2010 and Beyond, ftp://ftp.cordis.lu/pub/ist/docs/ngg2_eg_final.pdf
- [28] D3.6 Internal progress report on WP3 software evaluation and testing <http://www.crossgrid.org/M27deliverables.htm>
- [29] D2.6 Internal progress report on WP2 software evaluation and testing, <http://www.crossgrid.org/M27deliverables.htm>
- [30] D1.5 Evaluation of the WP1 2nd prototype <http://www.crossgrid.org/M27deliverables.htm>
- [31] DataGrid WP3 website <http://hepunix.rl.ac.uk/edg/wp3/documentation/doc/api/java/>
- [32] Next Generation Grids 2, Requirements and Options for European Grids Research 2005-2010 and Beyond, ftp://ftp.cordis.lu/pub/ist/docs/ngg2_eg_final.pdf
- [33] CoreGrid project <http://www.coregrid.net/>
- [34] The Common Component Architecture Forum. Available: <http://www.cca-forum.org/>
- [35] C. Szyperski, Component Software: Beyond Object-Oriented Programming. Addison-Wesley, 1999.
- [36] Enterprise JavaBeans technology. <http://java.sun.com/products/ejb>
- [37] CORBA Component Model, v3.0. Object Management Group, Inc. <http://www.omg.org/technology/documents/formal/components.htm>
- [38] COM: Component Object Model Technologies. Microsoft Corporation. [Online]. Available: <http://www.microsoft.com/com/default.msp>
- [39] S. Lacour, C. Perez, and T. Priol, Deploying CORBA components on a computational grid: General principles and early experiments using the Globus toolkit, in Component Deployment: Second International Working Conference, CD 2004, Edinburgh, UK, May 20-21, 2004. Proceedings, ser. LNCS, W. Emmerich and A. L. Wolf, Eds., vol. 3083. Springer-Verlag, 2004, pp. 35-49.
- [40] F. Baude, D. Caromel, and M. Morel, From distributed objects to hierarchical grid components, in International Symposium on Distributed Objects and Applications (DOA), Catania, Sicily, Italy, 3-7 November, ser. LNCS, vol. 2888. Springer-Verlag, 2003, pp. 1226 -1242.
- [41] Fractal component model; <http://fractal.objectweb.org/>

1. EXECUTIVE SUMMARY

This document constitutes the final definition of CrossGrid architecture, as mandated by the Technical Annex. As such, it describes the current status of the CrossGrid software (applications, tools and services) implemented and refined during the course of the Project. It should be understood as a general introduction to the CrossGrid software suite, while more detailed technical descriptions will be contained in separate final deliverables for each Work Package as well as in the installation, users' and developers' manuals accompanying the final batch of CrossGrid documentation. This document also describes the interoperability of CrossGrid components with external tools and software packages, as well as with emerging technologies in the area of Grid computing and software development.

The document is structured as follows:

- Section 1 contains this executive summary
- Section 2 contains an introduction to the final definition of the CrossGrid architecture and
- Section 3 presents a general picture of the CrossGrid architecture, updated from the previous deliverable (D5.2.6) and reflecting the final status of the Project. It lists inter-WP and inter-task dependencies and presents the workflow of an interactive job submission in the CrossGrid environment. Additionally, it describes in detail the interfaces exposed by each CrossGrid task and it also describes the elements of interactivity in CrossGrid: online input control and output monitoring (using both Java clients and legacy applications) as well as runtime steering in interactive applications.
- Section 4 describes the underlying middleware used for CrossGrid operations, with focus on the LCG and Globus suites, presenting the interoperability of CrossGrid components with these basic software modules.
- Section 5 describes how CrossGrid software is adapted for further development in accordance with the emerging trends in Grid technologies such as OGSA, WSRF and component technologies.
- Section 6 is devoted to CrossGrid interoperability with other major projects/undertakings in the Grid technologies field. Special attention is devoted to the evolution of OGSA interoperability throughout the Project's lifecycle as well as to interoperability with Grid testbeds and other application-driven projects.
- Section 7 contains closing remarks.

This document is envisioned as an addition to the specific installation, users' and developers' manuals mentioned earlier, providing an introduction and overview of the environment in which individual CrossGrid software components are intended to operate.

2. INTRODUCTION

CrossGrid has developed new Grid services and tools for interactive compute- and data-intensive applications like the biomedical simulation and visualization for vascular surgical procedures, the flooding crisis team decision support system, distributed data analysis in high energy physics and air pollution combined with weather forecasting. One of main objectives of the CrossGrid was to propose and develop a unified approach to running interactive distributed applications on the Grid.

The architecture of the CrossGrid software was defined as the result of detailed analysis of requirements from applications [1], and in its first form it was recently presented in an overview [2]. During the progress of the Project the architecture was refined [7]. The usage of Globus Toolkit 2.x was decided upon at the beginning of the Project for stability reasons and because of close collaboration with the DataGrid project. Nevertheless, the software is developed in such a way that it will be easy to use in future Grid systems.

The tools and services of the CrossGrid are complementary to those of DataGrid, GridLab (CG has a close collaboration) and US GrADS.

The main objectives of this document are

- to present the current (final) description of the CrossGrid subsystems, dependences between them and their interfaces,
- to give general picture of the flow of control and data of interactive jobs,
- to describe how the CrossGrid software interacts with the underlying Grid middleware,
- to show how the CrossGrid tools and services may be used in the future grids based on OGSA and WSRF as well as component-based Grids.

3. FINAL CROSSGRID ARCHITECTURE

3.1. OVERALL PICTURE

3.1.1. Subsystems

The CrossGrid architecture consists of a set of self-contained subsystems, developed within the Project. These subsystems can be divided into layers that contain applications, software development tools and Grid services. The components and layers are shown in Fig. 1. The application subsystem represents all the applications from WP1. The supporting tools are those developed within WP2 plus the user interfaces for accessing CrossGrid environment, such as portal and Migrating Desktop. The Grid services layer consists of all the subsystems from WP3, one from WP1 (GVK) and external ones (EDG, Globus). In addition to the CrossGrid components, we also distinguish the most important external subsystems used, namely Globus Toolkit, EU DataGrid and MPI libraries.

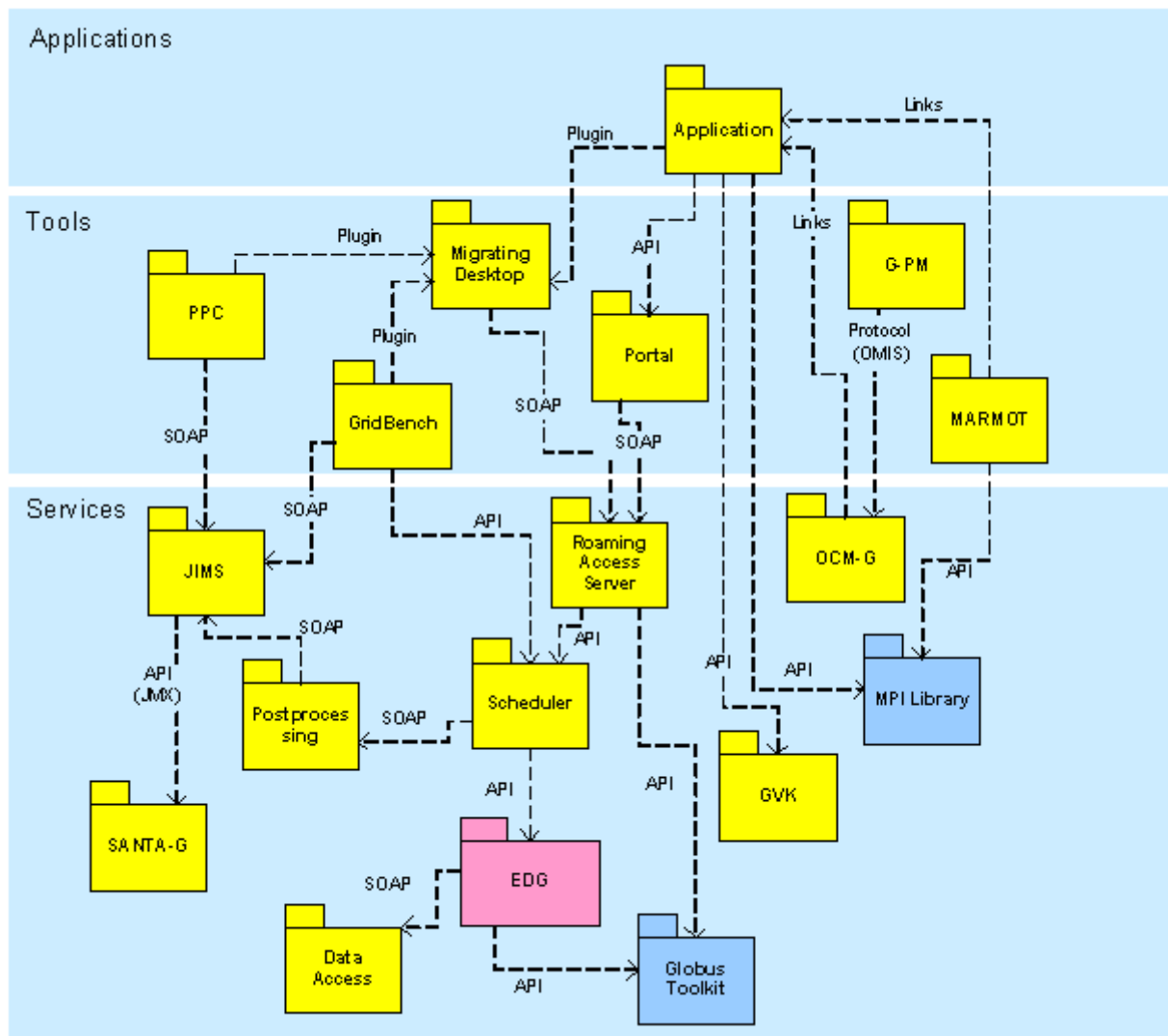


Fig. 1 The layered architecture of CrossGrid

The subsystems are shortly described in the table below, while detailed descriptions follow in subsequent sections of this document.

Subsystem	Short Description	WP/Task
Application	CrossGrid Applications	1.1-1.4
GVK	Grid Visualization Kernel	1.1
MARMOT	MPI Verification Tool	2.2
GridBench	Grid Benchmarks	2.3
PPC	Performance Prediction Tool	2.3
G-PM	Performance Analysis Tool	2.4
Migrating Desktop	User GUI Front-end to CrossGrid	3.1
Roaming Access Server	Server for Migrating Desktop	3.1
Portal	Lightweight Web-based frontend	3.1
CrossBroker	CrossGrid scheduling extensions to EDG Resource Broker	3.2
Postprocessing	Service for postprocessing of monitoring data for the CrossBroker	
SANTA-G	System for network infrastructure monitoring	3.3
JIMS	JMX-based Infrastructure Monitoring System	3.3
OCM-G	Application Monitoring System	3.3
Data Access	Subsystem for optimization of data access	3.4
EDG	EU DataGrid Software	
Globus Toolkit	Globus Toolkit components from EDG distribution	

3.1.2. Dependencies

All the subsystems of CrossGrid are developed as independent software pieces to facilitate the distribution of work between partners, to allow code reusability and exploitation of produced software in the future. However, there are many connections between these subsystems that make CrossGrid an integrated software environment for Grid application developers and users. These connections are depicted in Fig. 1 using an UML dependencies diagram. Arrows run from subsystems that use other CrossGrid modules to the ones being directly used by them. The picture also includes types of interfaces. Thick lines correspond to interfaces that are developed and running, while thin lines show the interfaces that are agreed upon, but still under development.

The following types of interfaces are used:

- SOAP – the Web Service interface is used,
- Plugin – the applications and tools use plugins that are installed in the Migrating Desktop,
- Protocol – the components communicate using a specific protocol,
- Links – the tools are directly linked to the application executable,
- API – the components use the Application Programmer Interfaces to communicate directly.

3.2. WORKFLOW OF INTERACTIVE JOB SUBMISSION

The workflow of interactive application submission is shown in Fig. 2. An interactive job is prepared using the graphical tools of the Migrating Desktop and submitted to the Grid infrastructure via the Roaming Access Server, which passes it to the CrossGrid Scheduler. Next, the job is directed to EDG and Globus middleware. Graphical output is returned to the user's desktop and can be automatically updated. The user can additionally monitor his/her application and infrastructure using development tools.

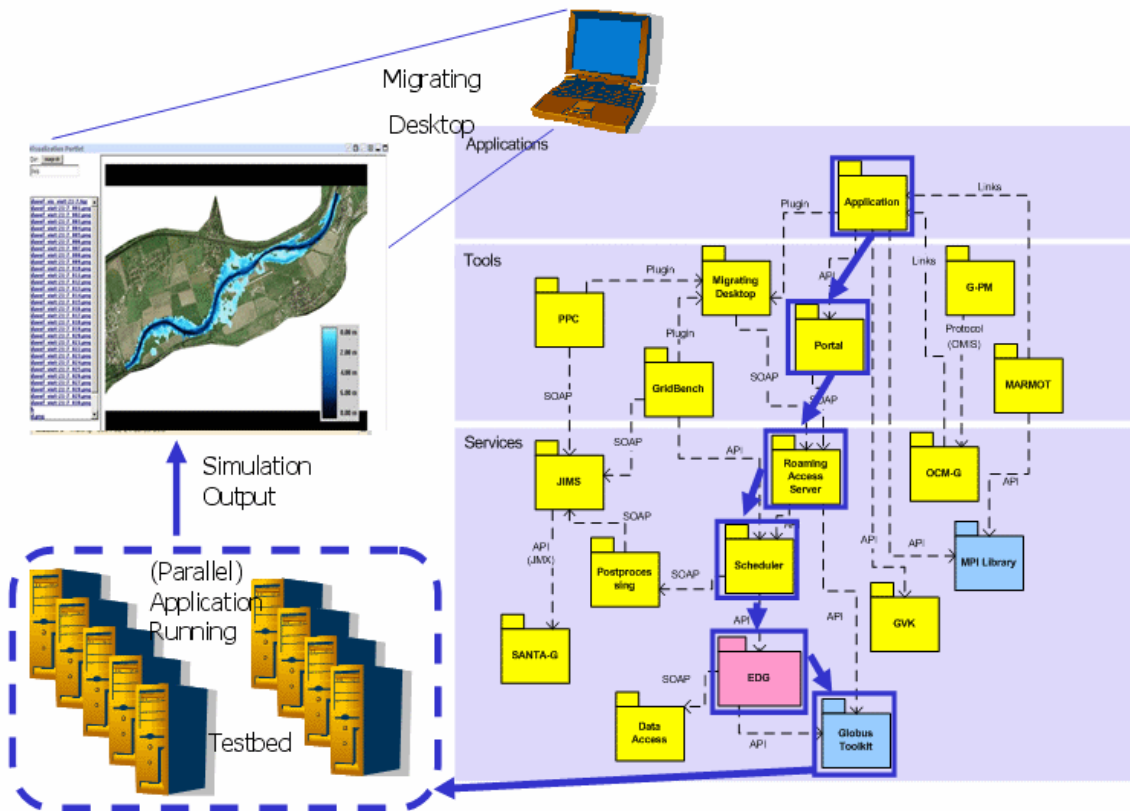


Fig. 2 The workflow of interactive job submission

3.3. ON-LINE INPUT CONTROL/OUTPUT MONITORING

Once the application is started, it is desirable to monitor its output on-line and to allow the user to control it while running. This can be achieved with streaming techniques shown in the Fig. 3a (the case of the Java Visualisation customized client) and in the Fig. 3b (the case of the legacy client). In order to submit interactive jobs in CrossGrid, the user has to download the Migrating Desktop (MD) to his/her local machine. The MD allows the user to interact with the Roaming Access Server (RAS). To allow users to submit interactive jobs and see the application GUI, either a VNC mechanism is put in place between the RAS and the MD (which downloads the VNC Client as Java applet), via ssh tunnelling or a Java Visualisation customized client is plugged into the MD. In the case of legacy applications, both VNC server and application GUI is run on an Application Client Machine (ACM) different from the RAS. VNC can present the Virtual Desktop of the RAS machine where an instance of the application GUI tool is launched. Following job submission (done by CrossBroker), the streams (in, out and error) of a job running on a Worker Node of a Computing Element are exchanged with the RAS machine or to the legacy application machine by means of a Console Agent (CA)/ Job Shadow

(JS) pair. The CA is a process that is launched together with the user job on the Worker Node. The CA opens a connection towards a JS running on the RAS machine or on the ACM for legacy application machine and they start exchanging data.

Interactivity for Java Visualisation customised client

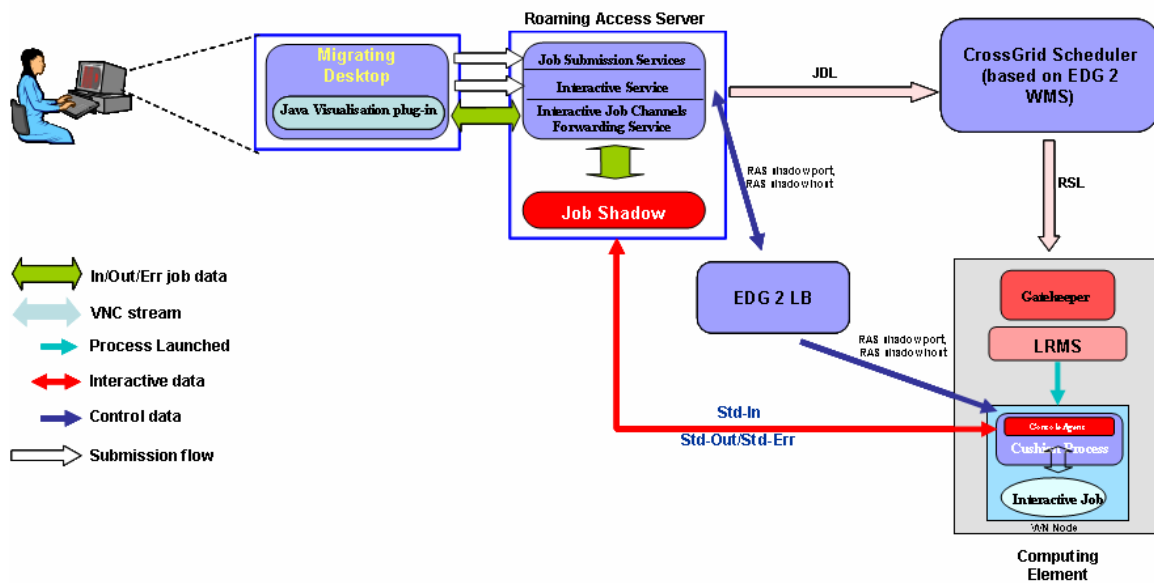


Fig. 3a On-line Input Control/Output Monitoring in CrossGrid for the Java client

Interactivity for legacy client applications

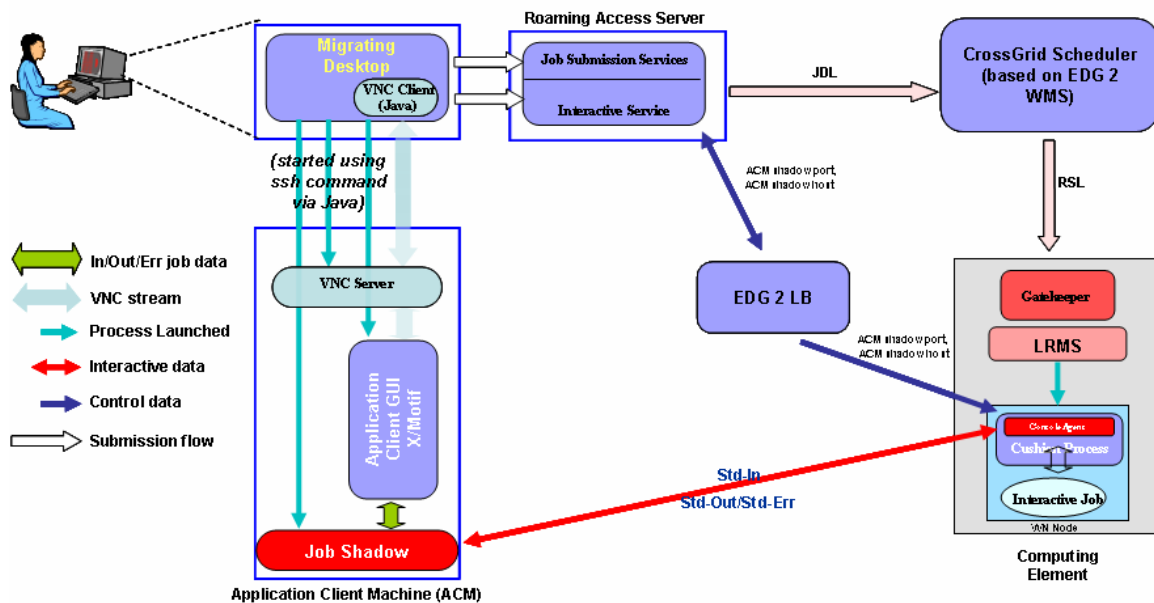


Fig. 3b On-line Input Control/Output Monitoring in CrossGrid for legacy clients.

3.4. RUNTIME STEERING IN INTERACTIVE APPLICATIONS

CrossGrid involves experimental research to support distributed interactive simulations on the Grid basing on the Grid Services architecture. The experimental system supports the requirement of the medical application for efficiently changing the simulation execution parameters while it is running and efficiently receiving responses to those changes.

This requirements can be fulfilled either by middleware mechanisms like the ones described above, or on the application level by using libraries designed for this purpose.

In CrossGrid, we have decided to use the High Level Architecture (HLA) [4] standard suitable for those purposes and its implementation called the Runtime Infrastructure (RTI). This work is done mainly in the Medical Application, since it is the only application within CG that actually requires such functionality. The agent system built over HLA supports development and execution of interactive distributed components. Also, there is experimental research on porting HLA-based applications to the Grid. The experimental system basing on the OGSA concept has been partly implemented and the results can be found in [3]. The tests were performed on the Dutch testbed DAS2 and not on CrossGrid testbed, because of the experimental nature of tests and the differences in technology; moreover, the HLA is not an open source project (this creates intellectual property issues).

It should be noted that the system is experimental and not involved in the mainstream development process of the medical application. The aim of the research is to check if OGSA solutions are suitable for such purposes. It should also be noted that this system aims at supporting efficient execution of HLA-based applications and this requirement cannot currently be fulfilled by existing implementations of CrossGrid Scheduler, GT2 GRAM and Condor-G (namely we need a support of migration of parts of HLA applications that are connect with other parts using the HLA bus). However, there are other components of CrossGrid software that we are using (see below).

The High Level Architecture (HLA) is one of the most widely used standard for distributed interactive applications and it supports development of simulations (called *federations* in HLA nomenclature) comprised of distributed components (called *federates*). It provides the simulation developer with many useful features such as time and data management needed by time-driven distributed simulations. However, the HLA standard was developed assuming a certain quality of service in the underlying environment of simulation execution, which is not provided by the Grid.

We have proposed and partially implemented an OGSA-based system that supports running distributed interactive application based on the HLA standard in a Grid environment.

The current architecture of the system is shown in Fig. 4.

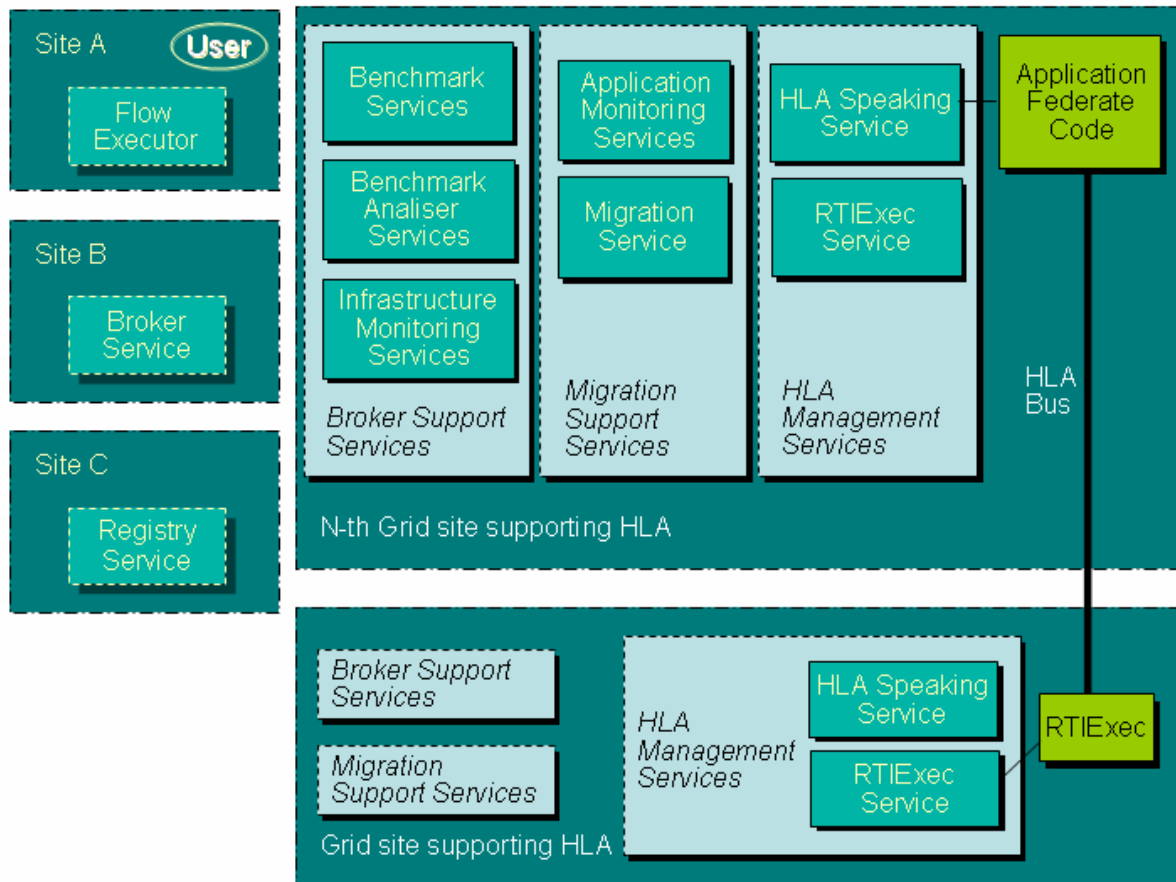


Fig. 4 The architecture of Grid Services framework for HLA-based application support

Each site that supports HLA in a Grid environment should provide the following services:

- **HLA Management Services** that interface and manage actual HLA installations on that site[5],
- **Migration Support Services** that facilitate direct support for fault-tolerant and effective performance of HLA-based applications (OCM-G for performance monitoring and migration decision support)[4],
- **Broker Support Services** which aim at providing the Broker Service with information necessary for decisions about setup and migration of application components.

3.5. DETAILED DESCRIPTION OF INTERFACES

3.5.1. Medical Application

The medical application [30] requires a distributed environment consisting of simulation, interaction and visualization components which allows the user to change simulation parameters in near-real time (**remote steering**). The details about solutions developed within this task are better described in Section 3.4

Dependencies:

The medical application is integrated with the Migrating Desktop and GVK [12].

3.5.2. Flood Application

A second type of interactive applications is required for Prediction and Flood Protection system [30]. An interactive Grid system fulfilling the needs of this application should allow experts to prepare cascades of meteorological, hydrological and hydraulic simulations basing on the assumption that each preceding step of the cascade produces input for the next simulation. After each of the steps is completed, the expert should be allowed to decide whether there is a need for the next simulation step in the cascade to be performed.

Dependencies:

The flood application is integrated partially with the MD and the Portal (for submitting single simulations). Also, it uses RAS client API and the JetSpeed library for developing its own portlet for submitting automatic cascades.

Marmot and GP-M have been used during testing of Aladin meteorological model and DaveF hydraulics model, and will be used for the MM5 meteorological model.

The performance of clusters connected to the Crossgrid testbed will be compared and evaluated using the GridBench tool.

3.5.3. HEP Application

The High Energy Physics application [30] requires support from a Grid interactive system that allows for online progress monitoring of their results in order to help operators decide about further job execution (i.e. interrupting the execution or letting it finish). The jobs are mainly MPICH-G distributed jobs.

Dependencies:

The application is integrated with the Portal and Migrating Desktop via a VNC mechanism.

3.5.4. Integration of Parallel Code for Air Quality Models into the GRID Structure

This application consists of two major components: weather prediction, providing input to air pollution simulations. The purpose of this application is to show the usefulness of Grid tools in air pollution modelling. STEM-II, an Eulerian air quality model, is used to simulate the environment. Model forced by meteorological data generated by atmospheric model COAMPS simulates gaseous and aqueous concentration fields of each modelled species, reaction rates, amount of deposited species and ionic concentrations.

Dependencies:

The application is integrated with the Migrating Desktop using the plugin mechanism. The code of this program has been tested with MARMOT and the kernel of the model is being analyzed by the PPC tool and the GridBench teams.

3.5.5. Wave Models

The purpose of this application is to show the usefulness of Grid tools for maritime applications. The user responsible for prevention of sea pollution and warnings will use the atmospheric model to simulate the nested spectrum of weather conditions over the Baltic Sea and to produce initial and boundary data for sea wave models.

Dependencies:

The application is integrated with the portal.

3.5.6. Data Mining – SOM

The purpose of the application is to implement Grid tools for a sophisticated statistical pattern recognition algorithm. The method called SOM (self-organising maps) is one of clustering algorithms in which the training process includes a neighbourhood adaptation mechanism so neighbouring clusters are quite similar, while more distant clusters become increasingly diverse. The main goal is to design an adaptive scheme for distributing data and computational load according to the changing resources available for each Grid job submitted. The data mining algorithms have been migrated to Grid and clusters.

Dependencies:

The application is integrated with the Portal.

3.5.7. Grid Visualisation Kernel

This tool [12] provides a visualization engine running in the Grid environment.

Dependencies:

The tool is used by the medical application.

3.5.8. MARMOT

The MPI verification tool MARMOT [29] is a tool that tries to verify the correct usage of MPI. Both C and Fortran language binding of MPI standard 1.2 are supported. The Fortran interface is implemented as a wrapper on top of the C interface.

MARMOT is a library that is linked to the MPI application in addition to the existing MPI library and that allows a detailed analysis of this application at runtime. The MPI profiling interface gives the user the possibility to replace MPI calls by routines with different functionality, and provides a second, name shifted version of all MPI routines. The verification tool uses this interface to check the MPI calls for consistency and correctness before subsequently passing it to the real MPI library for further processing.

MARMOT has been tested with C and Fortran applications, namely with the Biomedical application from Task 1.1, the Flood application from Task 1.2, the High Energy Physics application from Task 1.3 or the Meteo applications from Task 1.4:

Interfaces:

Output is written to standard output streams (stdout and stderr) by default and can be redirected to a file by the user. The format is human-readable.

Dependencies:

As MARMOT is intended to be a portable tool running on any environment, there is no specific hardware required for that purpose. The software required to compile MARMOT and link it to applications includes the following components: an MPI implementation, a C++ compiler, a Fortran Compiler.

3.5.9. GridBench

The CrossGrid benchmark suite, GridBench [29] allows investigating the performance of Grid constellations as well as Grid applications by users or administrators. In this way, benchmarks may provide reference data for the high-level analysis of applications as well as parameters for performance prediction. Among others, the GridBench suite provides:

- Benchmarks derived from Crossgrid and other Grid applications
- MPI micro-benchmarks
- Data-transfer benchmarks

- High-Performance Linpack
- CPU micro-benchmarks
- Memory micro-benchmarks
- Local Disk-IO benchmarks

Interfaces:

The GridBench provides the user with the GUI that is integrated in the Migrating Desktop. The GUI gives the following operations:

- Defining a benchmark
- Browsing the Results
- Creating charts

The output of the benchmark is an XML document that is stored in a Xindice database. This database can be queried in order to retrieve the results of benchmarks.

Dependencies:

The GridBench is launched like a normal grid application via Migrating Desktop. It uses JIMS to measure infrastructure load during benchmark execution.

3.5.10. PPC Performance Prediction Tool

The tool enables predicting performance of applications running on the Grid [29]. After analysis of performance of application kernels depending on input parameters, the tool can predict how long it will take to run a specific application on a specific site. The information provided by PPC includes: computational cost, number of FLOPs, number and size of send and received messages, memory cost, predicted execution time and load balance. The tool includes a set of performance models obtained from exhaustive executions of the main kernels. These models are analytical expressions that correlate grid parameters with features of the kernels to produce performance information. To get monitoring data about the Grid infrastructure, PPC uses JIMS as a tool.

Interfaces:

The interface to the tool is a GUI that is integrated in Migrating Desktop. The tool allows choosing the available application kernels and site configuration parameters. The output is presented in the form of charts.

Dependencies:

The PPC is a standalone tool and runs as a whole inside Migrating Desktop. It uses JIMS to provide necessary information about infrastructure.

3.5.11. G-PM Performance Measurement Tool

The performance measurement tool, called G-PM [29], consists of three components:

- a performance measurement component (PMC),
- a component for high level analysis (HLAC),
- a user interface and visualization component UIVC.

The PMC component provides the functionality for basic performance measurements of both Grid applications and the Grid environment. The results of

these basic measurements can be directly visualized by the visualization component. In addition, they can serve as an input to the high level analysis component and the performance prediction component.

The HLAC component supports an application and problem specific analysis of the performance data. On the one hand, it allows to measure application specific performance metrics. On the other hand, it provides a specification language which allows combining and correlating different performance measurements in order to derive higher-level metrics. The objective of this approach is to provide more meaningful data to application developers.

The UIVC component allows requesting performance measurements and displays the resulting performance information in various graphical ways.

Interfaces:

The interface to the GP-M is provided as a GUI (UIVC). It is an X11 application running on Linux.

The input to the HLAC is file containing measurement defined in the Performance Measurement Specification Language (PMSL).

Dependencies:

The G-PM tool uses the OCM-G application monitoring service. It connects using OMIS interface, which is a text-based protocol using direct TCP connection.

3.5.12. Portal

The Portal [28] is a very simple user interface for that allows running DataGrid commands for job submission. It exposes only interface for user. It uses web Service interface (SOAP over HTTP) to communicate with the Roaming Access Server.

Interfaces:

The portal provides a graphical interface to users.

In addition, the portal provides an interface to the flood application by using JetSpeed and RAS client (Java API) for job submission (flood application wrote separate portlet using this libraries).

The portal is also used as a user interface to application of Wave models and Weather prediction (Neural Networks.)

Dependencies:

The portal uses the interface of the Roaming Access Server - RAS through Web services (SOAP/HTTP)

3.5.13. Migrating Desktop

The Migrating Desktop [28] acts as an advanced user interface to the Roaming Access Server.

The various CrossGrid applications and tools can be integrated with Migrating Desktop by means of plugins that are designed to act similar to those used in popular browsers. A plugin is a module that can be easily implemented and integrated with its “parent” application (Migrating Desktop).

Plugins are developed by application and tool developers and the main aim is visualization of Grid application output and integration of Java applications (like e.g. Grid tools implemented within WP2, the VNC viewer, etc) with the Migrating Desktop

Application plugins are used to input application-specific input parameters. This enables a portal framework that is independent of application types so that it is simple to extend it by adding new applications. Tool plugins serve to modify job parameters according to the specific tool’s needs.

Each plugin is delivered as a set of Java archive files (available at some network location) from which the Migrating Desktop loads classes dynamically using the Java reflection mechanism. Details can be found in task 3.1 deliverables [6].

Additionally, for interactive application it is possible to use MD to launch a VNC Server on a RAS machine. A VNC client is integrated with the MD. In this way, advanced graphical visualisation tools can be used to interact with the application.

Interfaces:

Provides interface to application and tools by Java API, JAR plugins

Applications that use the MD:

- Medical (application plugin - development phase)
- Flood (single job submission - application plugin)
- HEP (VNC Server)
- Air pollution (application plugin)
- Atmospheric models - weather prediction (application plugin)

Tools that use the MD:

- MARMOT (tool plugin - development phase))
- GridBench (tool plugin)
- PPC (tool plugin)
- OCM-G (tool plugin)

Dependencies:

Uses the RAS interface (Web services SOAP/HTTP)

3.5.14. Roaming Access Server

This component provides access to the Grid [28]. In general, its functionality should be accessible for tools and applications via the Migrating Desktop or Portal as described below. RAS exposes its interface as a Web Service (SOAP over HTTP). It uses Scheduler API for job submission and GridFTP API for data transfer.

Interfaces:

Provides interface to MD, Portal by Web services (SOAP/HTTP).

Dependencies:

Uses CrossBroker interface via Java EDG Job Submission Service API and Globus GridFTP via Java CoG API.

3.5.15. CrossGrid Scheduler (CrossBroker)

CrossBroker is the component that manages all jobs submitted by users. It has been built by extending the original functionality provided in DataGrid's Workload Management System [22] available in LCG2 release. DataGrid components that have been modified are the following: User Interface, Workload Manager, Matchmaker, Resource Broker, Job Adapter and Job Controller [28]. CrossBroker exposes its interface as an EDG JSS Java API, which is used by RAS (see above). This interface allows job submission and retrieval of its status and output.

CrossBroker supports submission of sequential and parallel jobs. Parallel jobs can be MPI applications that run only on a single cluster, MPI applications that run on multiple clusters, or computational workflows made of separate jobs that exhibit control and data dependencies. All types of jobs can be submitted in a batch-like way, but interactive input/output streaming is also supported for sequential and both types of MPI jobs. CrossBroker consists of:

Scheduling Agent that is a permanent job queue supporting multiple users accepting sequential and parallel jobs expressed in Job Description Language (JDL). It supports a Job pre-emption mechanism that allows simultaneous management of batch and interactive jobs.

Resource Selector that allows for automatic selection of computational resources according to requirements and preferences of user's application. It provides selection of resources belonging to multiple sites (set matchmaking) for MPICH-G2 jobs.

Application Launcher that provides reliable submission mechanisms of parallel applications on Grid resources and co-allocation of tasks on multiple resources. The best effort is made to automatically recover from Grid failures

The support for interactive jobs is described in detail in Section 7.1

Interfaces:

Provides interface to RAS by Java EDG JSS API. It provides also a C++ API for monitoring tools (in particular, this API is used by Postprocessing Tool).

Dependencies:

Uses interface of DataGrid by code modification. It also uses Postprocessing Tool for gathering monitoring data (C++ API interface).

3.5.16. Application Monitoring

Application monitoring provided by the OCM is used by the GPM Tool and is available via a standard protocol - OMIS [28]. The OMIS 2.0 specification defines also a set of six API functions, but they are merely for connecting to the monitoring system, sending the OMIS requests, etc. There is no real API which encapsulates the protocol - the user must know it and specify the monitoring requests practically at the level of protocol messages. For a detailed description of the API mentioned above, refer to [28].

Interfaces:

Provides interface to GPM tool via OMIS protocol.

Dependencies:

Uses MD by means of plugins.

3.5.17. SANTA-G

The Santa-G system [28] for non-invasive monitoring of infrastructure publishes its data into DataGrid R-GMA by using the CanonicalProducer API.

The Viewer module provides a Java Swing GUI that presents a graphical interface to users. It allows users to collect data from the R-GMA, by using the Consumer API, and to graphically view packets stored in the log files.

The API for the CanonicalProducer and CanonicalProducer Servlet form part of the DataGrid R-GMA and can be found at the DataGrid WP3 website [31].

Interfaces:

Provides interface to DataGrid R-GMA via Java API and JIMS by JMX interface.

Dependencies:

Does not use external components of CrossGrid (low layer of CG architecture).

3.5.18. JIMS

JIMS provides infrastructure monitoring data using Java Technology. JIMS is capable of dynamically discovering monitored stations, dynamically deploying monitoring agents that monitor Linux or Solaris hosts. It uses SNMP (Simple Network Management Protocol) to gather data from monitored networking devices. Monitored parameters include host information like number of CPU-s, CPU statistics, memory usage, memory I/O map and information about network interfaces (bandwidth, etc.) and statistics.

Jims exposes its interface as a SOAP/HTTP Web Service interface, that is going to be used by GridBenchmarks, Performance Prediction Tool and Postprocessing. A detailed description of this interface can be found in [28].

Interfaces:

Provides interface to Grid Benchmark, Performance Prediction Tool and Postprocessing via Web services (SOAP/HTTP)

Dependencies:

Uses Santa-G by JMX interface.

3.5.19. Postprocessing

This task processes data derived from infrastructure monitoring. The tool is a lightweight monitoring and data analysis system, working on 24h/day basis, being able to deliver data describing Grid status to the CrossGrid scheduler. Three main parts of the tool are: sensors, installed on each computer in the Grid; central database gathering summary information from the clusters; and the data analysis module for preparation of the predictions of the future Grid status. Existing solutions were used and extended wherever it was possible. Sensors are built on top of the Ganglia monitoring system. The data analysis module is written from scratch and utilizes Kalman filter to predict the behavior of the Grid.

For details see [28]

Interfaces:

The tool is used by the CrossGrid Scheduler.

Dependencies:

Uses interface to JIMS via Web services (SOAP/HTTP).

3.5.20. Optimisation of Data Access

The Optimisation of Data Access provides Unified Data Access Layer (UDAL) that allows for easy to use and extend data access framework, capable to provide data stored inside Storage Node in a unified way together with estimations of access-factors to the data.

Additionally, this task provides routines for estimated bandwidth and latency for physical data access (for details see [28]). The system exposes its interface as a Web service interface (SOAP/HTTP) that is used by DataGrid Optor [11].

Interfaces:

Provides interface to DataGrid Optor via Web services (SOAP/HTTP).

Dependencies:

Does not use external components of CrossGrid (low layer of CG architecture).

4. UNDERLYING MIDDLEWARE

Crossgrid components are built over LCG [21] and Globus [10] middleware.

CrossGrid Applications include High Energy Physics (HEP) applications, therefore cooperation with DataGrid [11] project and its successor LCG [21] built over Globus, is very natural. CrossGrid extends EDG/LCG and Globus ideas in order to support interactive and parallel applications. In Fig. 5 we present the components from LCG2 [21] and Globus that cooperate with CrossGrid components.

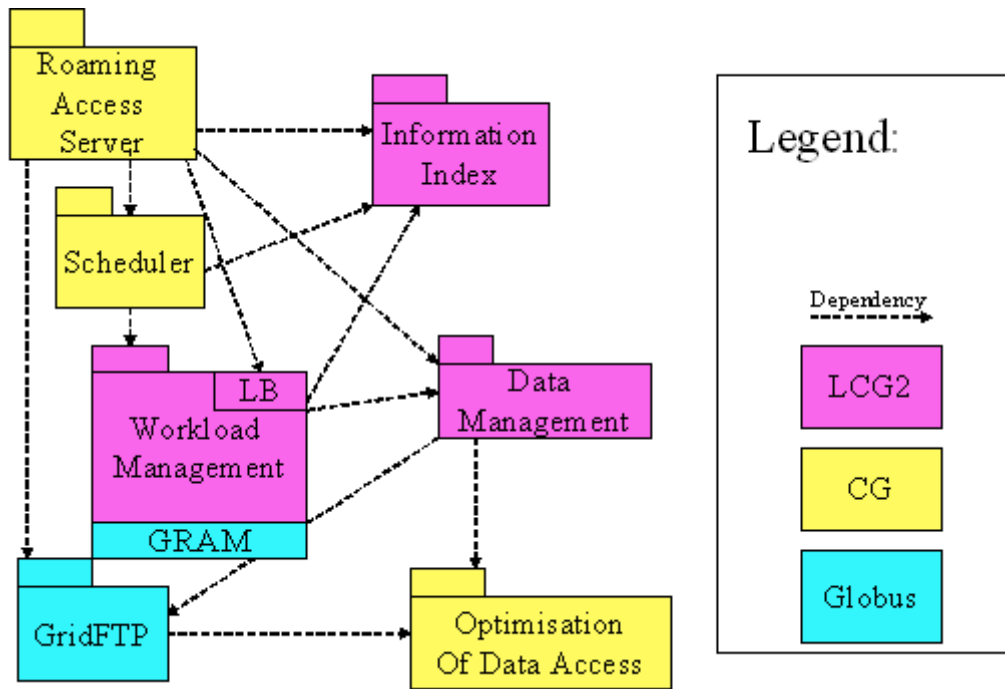


Fig. 5 CrossGrid Connections with LCG2 components.

Interoperability with LCG2 Workflow Management System (WMS) and Globus GRAM

WMS is used by Roaming Access Server (RAS) that takes data from its Logging and Bookeeping Service (LB) that stores information about actual running jobs. Also, CrossGrid Scheduler is an extension for LCG2 WMS allowing for running parallel MPICH-G2 jobs as well as interactive jobs. WMS uses Globus GRAM for job submission.

Interoperability with LCG2 Information Index (II)

Information Index is used directly by RAS and CrossBroker to get data about available resources. LCG2 II describes resources (Computing Elements and Storage Elements using Glue Schema). Therefore, compatibility of CrossGrid with GLUE and DataTag 1.1 is assured.

Interoperability with LCG2 Data Management (DM)

Data Management is used directly by RAS allowing a user for accessing files on Storage Elements (SE). Data Management uses CrossGrid Optimisation of Data Access to measure access time to various data storage systems and decide about accessing optimal replica of the file.

Interoperability with GridFTP

Globus GridFTP is used for file transfer by Roaming Access Server and Data Management.

5. CURRENT TRENDS IN GRID TECHNOLOGY

5.1. OGSA AND WSRF

At the very same time when CrossGrid project started, the Globus Project together with IBM and other industrial partners announced an initiative aiming at joining the Grid and Web Services technologies. This initiative included the Open Grid Services Architecture (OGSA) as a general vision of future Grid systems and the Open Grid Services Infrastructure (OGSI), as a basic standard. OGSI is based on Web Services standards, including XML, SOAP and WSDL. OGSI introduces mechanisms for state management in a distributed environment by defining transient and stateful Grid Services as extensions of stateless Web Services. The mechanisms defined in OGSI include lifetime management of service instances, service data for manipulating and querying on service state as well as notification. OGSI was standardized by the Global Grid Forum and OGSI v.1 was announced in July 2003.

In January 2004, the Globus Alliance announced an evolution of OGSA-related standards. The OGSI standard would evolve towards the new Web Services Resource Framework (WSRF) [9] and other specific standards such as WS-Notification. This evolution is a step towards the unification of (scientific) Grid Services and (commercial) Web Services communities. While it may become an important step towards standardization of both scientific and commercial Grid technologies, it delays the development of Grid systems that are based on these new standards.

OGSA, under the GGF standardization process, is currently one of the most important frameworks, aiming to be applicable and adopted for distributed system integration, virtualization, and management [25]. OGSA proposes a Service Oriented Architecture to be an underlying framework for Grid systems, and it specifically mandates usage of Web services standards, defined in WS-Architecture [26] and in WSRF [9] as important extension.

OGSA specifies 8 important categories of services, which we enumerate citing [24]:

- Infrastructure Services: enabling communication between resources (computer, storage, applications, etc.) removing barriers associated with shared utilization.
- Resource Management Services: enables the monitoring, reservation, deployment, and configuration of grid resources based on quality of service requirements
- Data Services: enables the movement of data where it is needed - managing replicated copies, query execution and updates, and transforming data into new formats if required.
- Context Services: describe the required resources and usage policies for each customer that utilizes the grid - enabling resource optimization based on service requirements.
- Information Services: provide efficient production of, and access to, information about the grid and its resources including status and availability of a particular resource.
- Self-Management Services: support the attainment of stated levels of service with as much automation as possible, to reduce the costs and complexity of managing the system.
- Security Services: enforce security policies within a (virtual) organization, promoting safe resource-sharing and appropriate authentication and authorization of users.
- Execution Management Services: enable both simple and more complex workflow actions to be executed including: placement, provisioning and management of the task lifecycle.

5.2. GRID COMPONENTS

Another important trend in Grids is to adapt distributed component technologies for both underlying grid middleware and as a higher-level programming model. These trends are expressed mainly in [32],

and are addressed by the currently running project CoreGrid [33]. Below we try to briefly summarize these trends.

The foundation of component-based systems is

“composition of applications from software units with specified interfaces and dependencies. The components can be deployed independently and can be composed by a third party” [35].

Component-based systems have gained popularity in industry, leading to standards such as EJB [36], CCM [37] (CORBA) and DCOM [38]. COM model gained popularity for Microsoft Windows programming, while EJB is widely used in J2EE enterprise applications. CCM is a component standard extending CORBA model proposed by OMG, but it has not gained wide acceptance yet. All these standards define some means for describing component interfaces (called ports or facets), both for the functionality provided by a component, and for dependencies.

Following the success of component model in industry, scientific community also expressed their interest in these areas. There are examples of adapting existing industrial standards, as well as proposing new ones, specifically suited for scientific applications. The work to adapt CORBA for scientific applications [39] belongs to the first group. It was shown, that it is possible to achieve high performance by efficient implementation of the CCM standard. The Common Component Architecture (CCA) [34] aims to adapt the component model for high performance scientific computations. The CCA offers the Scientific Interface Description Language (SIDL), which defines such additional datatypes as complex numbers and multidimensional arrays. Such types are necessary for high performance applications. CCA standard is implemented in frameworks, which examples are XCAT, CCAFFEINE and Scirun. CCA allows interoperability between different programming languages by exploiting the Babel tool [8] which generates multilingual bindings to SIDL interfaces.

Another approach to building component frameworks is based on other principles, such as active objects [40]. ProActive is a Java distributed component framework targeting Grid systems. It is based on the Fractal [41] component model, which allows hierarchical components.

None of aforementioned standards and frameworks have gained wide acceptance in the scientific community yet. One of the goals of the CoreGrid project is to propose a new, common standard for Grid components. This standard will later be used to build the common middleware for Next Generation Grid, as envisioned in [32]. The CrossGrid approach can be easily adapted to any of the standards described above, should one of them gain wider popularity in the scientific arena.

6. INTEROPERABILITY REQUIREMENTS

In this section, we discuss the interoperability issues related to the architecture of the CrossGrid project. The interoperability with EU DataGrid project (EDG) and its successors (LCG, EGEE) was addressed in details in previous sections. Here, we outline the ways towards interoperability with other application-oriented grid projects, and finally describe the efforts for interoperability with emerging grid standards in the form of OGSA and the Next Generation Grids envisioned in [27].

6.1. INTEROPERABILITY WITH OTHER APPLICATIONS DRIVEN PROJECTS

The middleware of CrossGrid project was designed basing on demands of its applications. It was based on GT2 in order to assure interoperability with other Grid projects, since Globus became de facto standard for Grid computing. Other applications driven Grid Projects like GrADS [19] or GridLab [20] are proposing their own Grid higher level infrastructure over GT 2.

The goal of the Grid Application Development Software (GrADS) [19] project is to simplify distributed heterogeneous computing in order to make Grid application development and performance more easy.

GrADS research is focused on six inter-institutional efforts:

- Program Execution System that includes scheduler application manager and monitoring,
- Program Preparation System (PPS) & Libraries which goal is to assist in the construction of configurable interfaces used to coordinate the launch and execution of Grid applications,
- ScaLAPACK library for linear algebra,
- Cactus framework,
- MacroGrid which is an environment to support the development of GrADS-specific software and components and to provide a large scale, heterogeneous, experimental execution environment for testing GrADS applications,
- MicroGrid that include Grid simulation tools.

GridLAB [20] produces a set of application-oriented Grid services and toolkits providing capabilities such as dynamic resource brokering, monitoring, data management, security, information, adaptive services and more. Services are accessed using the Grid Application Toolkit (GAT). The GAT provides applications with access to various GridLab services, resources, specific libraries, tools, etc. in a way that the end-users and especially application developers can build and run applications on the Grid without needing to know details about the runtime environment in advance. Applications use the GAT through a fixed GAT API.

Although not all of the components they developed could be useful for the CrossGrid applications (and similarly not all CrossGrid components could be useful for other projects applications), there are some services that are of general purpose and can be easily used without problems since they have the same underlying Grid technology (GT2). An example can be CrossGrid development tools like GPM or the OCM-G application monitoring system. Another example of this concept is the GrADS Network Weather Service (NWS) [23] that is used by CrossGrid Performance Prediction Component (PPC).

Then there are project components developed in each project, but in a different way – i.e. scheduler services. Each of the projects: CrossGrid, GrADS and GridLAB have developed their own schedulers suitable for the kind of project applications being considered. Again, a common underlying Grid infrastructure allows for easy adaptation of different solutions, if there is a need.

6.2. TESTBED INTEROPERABILITY

One of the most important objectives of the CrossGrid project was to spread the Grid testbed across new European countries. Such testbed is needed by developers of middleware and tools, as well as by application scientists. CrossGrid has made substantial effort to achieve interoperability with EDG and currently LCG and EGEE European Grid projects. This interoperability can be seen in the areas of architecture, security, deployment of software, and management.

The architecture interoperability was assured by choosing common middleware for the testbed, which was the Globus toolkit enriched with substantial number of higher-level services for job management, data management and information management. This has proven to be a good choice since it allows easy migration of CrossGrid testbed sites to EGEE and LCG testbeds.

The Grid security was not the mainstream research and development subject of the CrossGrid project. There was no special workpackage and task focused on security. Therefore as the security infrastructure the EDG software was used, which is based on GSI public key infrastructure. This required to set up the national Certificate Authorities, VO servers, local policies and procedures, and to coordinate them across the project. Now it is possible to use this infrastructure to continue using the Grid in the scope of the next projects.

Deployment of software on a testbed comprising multiple sites in different countries is a non trivial task. The reusing of EDG software and adapting it to CrossGrid specifics facilitated this process in considerable way. By relying on the common software base, it was possible to perform upgrades when it was needed by the applications, mainly from the HEP community, to be interoperable with the latest versions of EDG and later LCG software.

The integration and interoperability of the CrossGrid testbed with other important pan-european Grid projects, is one of the achievements of the Project. It enables scientists using the Grid infrastructure to conduct their research with unprecedented computing power.

6.3. INTEROPERABILITY WITH OGSA – EVOLUTION THROUGHOUT THE PROJECT

From the beginning of the Project, CrossGrid has been evaluating both the possibility of migrating to the new OGSI-based Globus Toolkit and the necessity of gearing the scientific work towards the service-oriented architecture of the Grid that OGSA envisioned. However, the first implementation of OGSI-based Globus Toolkit 3.0 was only released in June 2003 - the middle of the second year of CrossGrid development. Because of the dependency on DataGrid software, which is based on Globus 2, and also for stability reasons we have decided not to migrate to the new toolkit. Nevertheless, the Project focused on preparing software in such a way that it would be “OGSA-ready”. This entailed the usage of Web Services and XML where appropriate to facilitate the possible future migration to an OGSA-based Grid environment. Moreover, many CrossGrid partners initiated scientific and development work in this area.

From the development point of view, the current plans of the Globus Alliance are to concentrate on a new release, GT4.0, based on the new WS-RF standard. This release is announced to be available in 2005, putting it out of reach of the CrossGrid project.

The table below summarizes the current status of work on usage of Web Services within CrossGrid tasks. It can be seen that Web Services and XML are extensively used in many tasks; hence the interoperability may be facilitated.

CG Component	Web Services
1.1 Biomed app.	Experiments with Web Services interfaces and OGSi extensions to set up and control interactive simulations based on HLA
1.2 Flood app.	Uses Web Services for portal integration and OGSi extensions for metadata and workflow management
1.3 HEP app.	Uses XML as a format for data output
1.4 Meteo app.	No specific data
2.2 MARMOT	Low-level tool, no plans for WS interfaces
2.3 Benchmark	Uses Web Service interfaces and XML, produces output into XML database.
2.3 PPC	Tool executed locally, no plans for WS interfaces.
2.4 G-PM	Initiated research towards performance of service-based applications
1.1 GVK	No WS interfaces.
3.1 Portal	Uses Web Services.
3.1 Migrating Desktop	Uses Web Services as primary interfaces and XML.
3.1 RAS	Uses Web Services as primary interfaces and XML.
3.2 CrossGrid Scheduler + Postprocessing	Uses Web services for communication
3.3.1 OCMG	Initiated research towards monitoring of Grid Services
3.3.2 Santa-G	Uses WS interfaces for integration with JIMS
3.3.3 JIMS	Uses Web Services for internal and external communication
3.4 Data access	Uses Web Services for internal and external communication

We consider the approach CrossGrid has taken towards OGSA as a good solution for maintaining a long-term project in the midst of substantial changes and evolution of Grid technologies. The usage of GT2 is important for stability reasons and for the interoperability with LCG and EGEE (previously EU DataGrid) projects. On the other hand, it is important to prepare the new software in such a way that it will be easy to use in future Grid systems based on OGSA standards. The usage of Web Services not only makes integration of components an easy task, but also prepares them for migration to future OGSA standards. As Grid Services and Web Services merge, it should become possible to migrate to the future common technology. Finally, we have to mention that there are CrossGrid components that do not use explicitly the Web Services technologies, but are designed in a service-based style (as in the case of OCM-G). For such components, the interoperability with the future Grid service-based standards requires the adaptation on the external interface level only, without changing the semantics and internal implementation.

6.4. INTEROPERABILITY WITH NEXT GENERATION GRIDS

Next Generation Grids, as envisioned in [32] proposes a new general Grid architecture, consisting of Grid Foundations Middleware layer and Grid Services Middleware layer. These layers are currently defined on a very high level of abstraction without technical details, so it is impossible to precisely reason about interoperability with this model. However, since the CoreGrid project is aiming to conduct research in this area, focusing strongly on component models, we can deduce that the Grid

Foundations Middleware is going to be component-oriented to some extent. Therefore we can discuss briefly the possible interoperability of CrossGrid with such middleware.

As it was discussed before in context of OGSA interoperability, the architecture of CrossGrid can be represented in terms of a service-based architecture. Main components the CrossGrid software consists of have well defined interfaces and dependencies. Many of them have Web Services interfaces. It is known that the distributed component systems have a lot of common features with service-based systems, such as independence, well defined interfaces and composability. This is also reflected by the interchange of using terms “service” and “component” in this document. Therefore the interoperability of CrossGrid services with component-oriented grid middleware can be substantially facilitated by exploiting these similarities.

The general experience from the CrossGrid interoperability efforts may be following. As it is impossible to predict the future directions of Grid standards evolution, and moreover these standards may be subject to further changes, we have to emphasize that it is the generosity of solutions that is important and makes the software usable outside the project. In terms of interoperability, general conclusion is that the service-based approach proves to be well suited for such a large project involving distributed computing, as is in the case of CrossGrid.

7. SUMMARY

This document presents the overview of the final architecture of the software developed in the framework of the CrossGrid Project. It describes subsystems, their dependencies and their interfaces. The detailed description of the software elements is presented in the Users', Developers' and Installation Manuals which were elaborated by each Task of this Project. We have also described how the CrossGrid software interacts with basic grid software, such as LCG2.

The following conclusions can be drawn with respect to the CrossGrid approach to interactive applications:

- The complexity of the architecture is a result of the complex nature of the Grid environment and of dependencies on technologies used.
- Various application requirements and usage of legacy technologies result in diversity of interfaces; successful development of the Project in such a complex environment can be regarded as a great achievement of all the partners.
- Experience from the Project clearly justifies the need for standardization in the area of Grid technologies.

The CrossGrid software may be also used by the projects of similar nature, namely GridLab and GrADS.

The document explains the following aspects of the CrossGrid interoperability:

- with OGSA and WSRF-based Grids,
- with component-based Grids,
- of the CG testbed.

The conclusion is that effort invested in the CrossGrid software development may be preserved and the CrossGrid modules may be usable in the future Grids. This is already confirmed by the fact that a number of Project partners have started work on extensions to the CrossGrid software in K-WfGrid and CoreGRID projects. Another possibility in this regard may be the elaboration of pilot Grids for a larger group of users. where the software might be improved and tuned to the specific requirements.