



DELIVERABLE D5.2.5
DETAILED REPORT ON THE
CROSSGRID ARCHITECTURE
FINAL VERSION

WP5

Document Filename:	CG5.2-D5.2.5-v1.3-CYF501-CGArchitecture
Work package:	WP5
Partner(s):	CYFRONET
Lead Partner:	CYFRONET
Config ID:	CG5.2-D5.2.5-v1.3-CYF501-CGArchitecture
Document classification:	PUBLIC

Abstract: This document presents the second version of the software architecture which is under development in the CrossGrid Project. This architecture comprises a general overview of the software components and their dependencies. The components are applications, tools that support application development and new Grid services specific to the CrossGrid Project. Dependencies on external components, such as DataGrid and Globus software, are also presented in detail. In addition, this deliverable also discusses the implications of OGSA on our Project and presents recommendations for further development.

Delivery Slip

	Name	Partner	Date	Signature
From	Marian Bubak, Maciej Malawski, Piotr Nowakowski, Katarzyna Zajac	CYFRONET	9/6/2003	
Verified by	Piotr Nowakowski	CYFRONET	10/6/2003	
Approved by	Michał Turala	CYFRONET	10/6/2003	

Document Log

Version	Date	Summary of changes	Author
0.0	11/5/2003	Definition of contents and general structure	Marian Bubak, Maciej Malawski, Katarzyna Zajac
0.1	12/5/2003	Info on Globus Toolkit 3 and OGSA	Maciej Malawski
0.3	16/5/2003	Extensions following discussion	Maciej Malawski
0.4	17/5/2003	Description of architecture layers (based on initial definition and M15 deliverables)	Katarzyna Zajac
0.5	20/5/2003	Corrections following Steering Group conference	Katarzyna Zajac
0.6	21/5/2003	Updates on GT3	Maciej Malawski
0.7	22/5/2003	Improved use cases	Katarzyna Zajac
0.8	22/5/2003	Update on OGSA, Refinements	Maciej Malawski
0.9	23/5/2003	Further refinements	Marian Bubak, Maciej Malawski, Katarzyna Zajac
1.0	30/5/2003	Additional figures, Abstract, Executive summary, Proofreading and QA check	Marian Bubak, Maciej Malawski, Piotr Nowakowski, Katarzyna Zajac
1.1	05/6/2003	Additional changes after Mirek Kupczyk comments	Katarzyna Zajac
1.2	10/6/2003	Modifications according to Jorge Gomes review (many thanks for detailed comments)	Maciej Malawski
1.3	10/6/2003	Modifications according to Miroslaw Dobrucky	Katarzyna Zajac

CONTENTS

REFERENCES.....	5
LIST OF ABBREVIATIONS.....	7
EXECUTIVE SUMMARY.....	8
1. CHARACTERISTICS AND REQUIREMENTS OF INTERACTIVE APPLICATIONS.....	10
1.1. RUNTIME STEERING.....	10
1.2. CASCADE OF SIMULATIONS.....	11
1.3. INTERACTIVE DATA ANALYSIS.....	12
2. OVERVIEW OF CROSSGRID ARCHITECTURE.....	14
3. COLLECTIVE LAYER LIBRARIES.....	15
3.1. MPICH-G.....	15
3.2. HLA.....	15
3.3. MPICH-G vs. HLA.....	16
4. TOOLS.....	17
4.1. MPI DEBUGGING AND VERIFICATION TOOL.....	17
4.1.1. <i>Functionality</i>	17
4.1.2. <i>External Interfaces</i>	18
4.2. BENCHMARKS.....	18
4.2.1. <i>Functionality</i>	18
4.2.2. <i>External Interfaces</i>	19
4.3. GRID PERFORMANCE MEASUREMENT TOOL.....	19
4.3.1. <i>Functionality</i>	19
4.3.2. <i>External Interfaces</i>	19
4.4. PERFORMANCE PREDICTION TOOL.....	20
4.4.1. <i>Functionality</i>	20
4.4.2. <i>External Interfaces</i>	20
5. SERVICES.....	21
5.1. USER INTERACTION SERVICE.....	21
5.1.1. <i>Functionality</i>	21
5.1.2. <i>External Interfaces</i>	21
5.2. GRID VISUALIZATION KERNEL.....	22
5.2.1. <i>Functionality</i>	22
5.2.2. <i>External Interfaces</i>	22
5.3. PORTALS AND ROAMING ACCESS.....	22
5.3.1. <i>Functionality</i>	22
5.3.2. <i>External Interfaces</i>	23
5.4. GRID RESOURCE MANAGEMENT.....	24
5.4.1. <i>Functionality</i>	24
5.4.2. <i>External Interfaces</i>	24
5.5. MONITORING.....	25
5.5.1. <i>Functionality</i>	25
5.5.2. <i>External Interfaces</i>	25
5.6. DATA MANAGEMENT.....	25
5.6.1. <i>Functionality</i>	25
5.6.2. <i>External Interfaces</i>	26
6. DEPENDENCIES BETWEEN CROSSGRID COMPONENTS.....	28
7. CROSSGRID AND OGSA.....	29

7.1. OGSA AND GLOBUS TOOLKIT 3.0	29
7.1.1. OGSA (Open Grid Services Architecture).....	29
7.1.2. OGSi (Open Grid Services Infrastructure).....	29
7.1.3. GT3 (Globus Toolkit 3.0)	29
7.1.4. Current status of development in OGSA, OGSi and GT3.....	29
7.2. FUNCTIONALITY OF GT3 AND ITS RELATIONS TO SERVICES DEVELOPED IN CROSSGRID	30
7.3. OGSA-RELATED WORK WITHIN CROSSGRID	30
7.4. MIGRATION TO OGSA VS. MIGRATION TO GT3	31
7.4.1. Migration to GT3.....	31
7.4.2. Migration to OGSA	32
7.5. PLANS FOR THE SECOND YEAR OF THE PROJECT	32
7.6. OGSA ISSUES RELATED TO CROSSGRID TASKS	32
7.6.1. Task 2.2 (MPI code debugging and verification)	33
7.6.2. Task 2.3 (Metrics and benchmarks)	33
7.6.3. Task 2.4.1 (Performance evaluation tools).....	33
7.6.4. Task 2.4.2 (Performance prediction).....	33
7.6.5. Task 3.1 (Portals and Roaming Access).....	33
7.6.6. Task 3.2 (Scheduling).....	34
7.6.7. Task 3.3 (Monitoring).....	34
7.6.8. Task 3.4 (Optimization of Data Access)	35
8. SUMMARY	36

REFERENCES

- [1] CG-5-D5.2.2-0009-1-0-FINAL Deliverable 5.2.2 CrossGrid Architecture Requirements and First Definition of Architecture <http://www.eu-crossgrid.org/M3deliverables.htm>
- [2] The CrossGrid Project Homepage – <http://www.eu-crossgrid.org>
- [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] Foster, I., and Kesselman, C.: *The Grid: Blueprint for a New Computing Infrastructure*
- [5] The Globus Project Homepage – <http://www.globus.org>
- [6] Bubak, M., Malawski, M., Zając, K. Towards CrossGrid Architecture. In: Kranzlmüller, D., Kacsuk, P., Dongarra, J., Volker, J. (Eds.): Recent Advances in Parallel Virtual Machine and Message Passing Interface, Proc. 9th European PVM/MPI Users' Group Meeting, Linz, Austria, September/October 2002, LNCS 2474
- [7] M.Bubak, M.Malawski, K.Zając CrossGrid Architecture - 2nd Cracow Grid Workshop Proceedings, Kraków 2002
- [8] Baliś, B., Bubak, M., Funika, W., Szepieniec, T., and Wismüller, R.: An Infrastructure for Grid Application Monitoring In: Kranzlmüller, D., Kacsuk, P., Dongarra, J., Volker, J. (Eds.): Recent Advances in Parallel Virtual Machine and Message Passing Interface, Proc. 9th European PVM/MPI Users' Group Meeting, Linz, Austria, September/October 2002, LNCS 2474, pp. 41-49, 2002
- [9] Overview of Common Component and Grid Services Architectures – Materials from 1st EU CrossGrid Conference, Kraków, March 2002
- [10] Overview of OGSA and Globus 3.0 and their Application to CrossGrid - Materials from CrossGrid Workshop, Linz, September 2002
- [11] K. Zając et al. A Proposal of the Services For Managing Interactive Grid Applications – 2nd Cracow Grid Workshop Proceedings, Kraków 2002
- [12] K.Zając, M.Bubak, M.Malawski, P.Sloot: “Execution and Migration Management of HLA-based interactive simulations on the Grid.” Submitted to PPAM conference.
- [13] M. Malawski et al. Integration of CrossGrid Services into OGSA Model - 2nd Cracow Grid Workshop Proceedings, Kraków 2002
- [14] "The OGSA Platform", OGSA work group draft v3, <http://www.ggf.org/Meetings/ggf7/drafts/draft-ggf-ogsa-platform-2.pdf>
- [15] CrossGrid Deliverable D3.3 First WP3 Software Release <http://www.eu-crossgrid.org/M12deliverables.htm>
- [16] The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration (Draft 2.9, 6/22/2002) http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf
- [17] Open Grid Services Infrastructure v1.0 (Draft 29, April 5, 2003) http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-grid-service-29_2003-04-05.pdf
- [18] The European DataGrid Project Homepage - www.eu-datagrid.org
- [19] Global Grid Forum: <http://www.ggf.org>
- [20] HLA specification, <http://www.sisostds.org/stdsdev/hla/>
- [21] D1.1.2a Internal Progress Report WP 1.1 <http://www.eu-crossgrid.org/M12deliverables.htm>

- [22] D1.2.2.a Internal Progress Report WP 1.2
<http://www.eu-crossgrid.org/M12deliverables.htm>
- [23] D1.3.1.a Internal Progress Report WP 1.3
<http://www.eu-crossgrid.org/M12deliverables.htm>
- [24] Results of migration of data mining algorithms to Grid WP 1.4
<http://www.eu-crossgrid.org/M12deliverables.htm>
- [25] D2.2 WP2 Internal progress report (design).
<http://www.eu-crossgrid.org/M6deliverables.htm>
- [26] D3.2 WP3 Internal Progress report (design)
<http://www.eu-crossgrid.org/M6deliverables.htm>
- [27] D2.3 Demonstration and report on WP2 1st prototypes
<http://www.eu-crossgrid.org/M12deliverables.htm>
- [28] D3.3 prototype - first WP3 software release
<http://www.eu-crossgrid.org/M12deliverables.htm>

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network	MPICH-G	Grid enabled implementation of MPI
CE	Computing Element	OCM-G	OMIS-Compliant Monitoring for Grid
EDG	European DataGrid Project	OGSA	Open Grid Services Architecture
GBJ	Grid Batch Job	OGSI	Open Grid Services Infrastructure
GGF	Global Grid Forum	OMIS	On-line Monitoring Interface Standard
GIS	Grid Information System	PPC	Performance Prediction Tool
GM	Grid Middleware	RAS	Roaming Access Server
G-PM	Grid Performance Measurement	RB	Resource Broker
GRAM	Grid Resource Allocation and Management	R-GMA	Relational Grid Monitoring Architecture
GRM	Grid Resource Management	RTI	Runtime Infrastructure (HLA)
GT2	Globus Toolkit 2.0	SA	Scheduling Agent
GT3	Globus Toolkit 3.0	SOAP	Simple Object Access Protocol
GVK	Grid Visualization Kernel	SVG	Scalable Vector Graphics
HEP	High Energy Physics	UI	User Interface
HLA	High Level Architecture	UIS	User Interaction Service
ISM	Interactive Session Manager	UML	Unified Modeling Language
ISRB	Interactive Session Resource Broker	VO	Virtual Organization
ISW	Interactive Session Worker	VR	Virtual Reality
JDL	Job Description Language	WSDL	Web Services Description Language
LDAP	Lightweight Directory Access Protocol	XML	eXtensible Markup Language
MD	Migrating Desktop		
MDS	Monitoring and Discovery Service		
MPI	Message Passing Interface		

EXECUTIVE SUMMARY

The CrossGrid Project is oriented towards the following compute- and data-intensive applications, characterized by interaction with a person in the processing loop:

- simulation and visualization for surgical procedures,
- flooding crisis team decision support system,
- distributed data analysis in high-energy physics (HEP),
- air pollution combined with weather forecasting.

Each application requires a response from the Grid to actions initiated by a human agent, in different time scales.

In this document we present the second version of the CrossGrid software architecture, i.e., the general overview of software components (applications, tools, and CrossGrid-specific Grid services), their connections as well as dependencies on external components from the DataGrid project and Globus Toolkit. The first proposal of the CrossGrid architecture was defined at the very beginning of the Project and it guided the development of CrossGrid software for the first year. This, improved version, is based on the experience gained and on the evaluation of prototypes made available for the first integration meeting in February 2003.

Interactive applications pose specific requirements for the tool environment as well as for the Grid infrastructure, so the elaboration of the project architecture is preceded by a description of typical use cases for all applications.

Tools facilitate the process of code development, its optimization, tuning and execution of applications on the Grid. Tools currently under development within CrossGrid are Marmot for MPI verification, Grid Benchmarks for testing the behavior of applications in a Grid environment, a performance analysis tool (G-PM) that can perform on-line measurements on running applications and a performance prediction tool.

The requirements posed by applications and tools result in the necessity of creating new Grid services. User-friendly access to Grid resources for remote users is handled by portals together with the Migrating Desktop and the Roaming Access Server. The submission of parallel jobs is done by schedulers adapted for interactive applications. Grid monitoring services provide information on both infrastructure and application execution. Data Access services are used to optimize the time needed to transfer large data files from tertiary storage to computing nodes.

Architecture components are organized into six layers, according to their role in the Grid:

- fabric,
- generic Grid services (taken from Globus Toolkit and DataGrid),
- application-specific Grid services,
- supporting tools,
- collective layer libraries,
- applications.

They are described in detail in Sections 2 - 5 of the document. A complete description of the design of individual components is presented in the appropriate design documents (M9 and M15 deliverables).

Section 6 presents dependencies between components.

Section 7 discusses the consequences of the creation of OGSA and Globus Toolkit 3.x for the development of CrossGrid middleware. The analysis concerns two possible scenarios:

- migration to GT3 (i.e. using GT3 instead of GT2 as the basic set of services),

- migration to OGSA (i.e. making the services developed in CrossGrid OGSI-compliant).

Based on the evaluation of current status and plans for OGSA and GT3 in the near future, we propose to keep GT2 as the basic infrastructure for the CrossGrid testbed, in order to maintain compatibility with DataGrid. At the same time, an experimental installation of GT3 is proposed for evaluation and testing by the biomedical application.

It is expected that the new services and tools will continue to evolve in a way that will facilitate future migration to GT3 and OGSA. Therefore, another evaluation will be conducted after the end of the 2nd project year and a decision on a possible switchover will be made then.

1. CHARACTERISTICS AND REQUIREMENTS OF INTERACTIVE APPLICATIONS

The CrossGrid Project [2],[6],[7], is oriented towards applications which are characterized by interaction with a person in the processing loop. Each application requires a response from the Grid to actions initiated by a human agent in different time scales: from real time through intermediate delays to long waiting periods. The applications are simultaneously compute- and data-intensive. The following applications are currently being developed: simulation and visualization for surgical procedures, flooding crisis team decision support system, distributed data analysis in high-energy physics, air pollution combined with weather forecasting.

1.1. RUNTIME STEERING

The medical application [21] is a distributed near-real-time simulation with a user interacting in virtual reality and other interactive display environments. A 3-D model of arteries is the input to a blood flow simulation and the results are presented in a VR environment. The user alters the layout of the arteries and the effects are analysed in near-real time through VR. The medical application requires a distributed environment consisting of simulation, interaction and visualization components which will allow the user to change simulation parameters in near-real time. For that purpose, we propose an approach to problem solving environments consisting of a set of Grid services that allow the setup and interactive steering of complex Grid applications, with separate modules for simulation and visualization. This environment is a consistent, non-complex framework in which complex systems can be composed from reliable sub-units. The core of the proposed approach is an event system, the functionality of which will be based on existing standard infrastructures for distributed interactive simulations, called the High Level Architecture (HLA) [20]. The interaction between components is shown in the Fig. 1.

The use case scenario is presented below:

1. A physician submits a request for access to Grid resources, via the Portal.
2. The 3D Visualization System starts, using the Grid Visualization Kernel (GVK) Grid Service, showing previous simulation results.
3. The physician studies previous simulation results and proposes new simulation parameters (i.e., selects bypass designs and updates selected simulation parameters) using the interactive GVK.
4. The physician submits new simulation parameters via the Portal (i.e., once the new parameters are selected, the new job is submitted).
5. The Scheduling Agents (SA) schedules a new simulation job taking into account the available Grid resources.
6. The Application and GVK initialize simulation and visualization of the new job (i.e., once the Scheduling Agents has scheduled the new job, simulation starts and updates are fed into the GVK for visualization and monitoring by the physician).
7. The Grid Performance Monitoring (G-PM) tool make heartbeat and selected performance data accessible to an optional system administrator user, via the Portal ("monitoring" via Portal means simple plain-text updates on the state of the simulation).
8. The physician accesses the simulation state on the Portal.
9. The physician monitors ("monitoring" via GVK includes visualization) the progress and controls job execution via GVK, may change parameters, reviews results, and store comments as well as selected parameters in a local repository.
10. The physician stops the simulation.

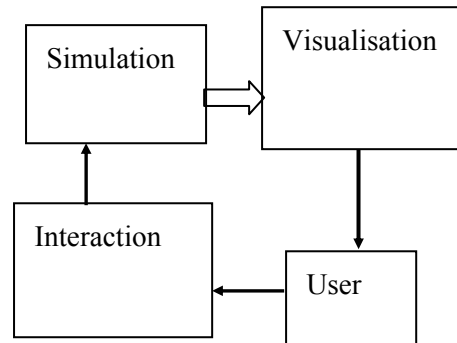


Fig. 1 Runtime control in the medical application

1.2. CASCADE OF SIMULATIONS

Flood forecasting [22] requires meteorological simulations of different resolutions, from mesoscale to storm-scale. Selected hydrological models are used to determine water discharges from the affected area, and with this information hydraulic models simulate flow through various river structures to predict the impact of the flood. The results of simulations can be interactively analyzed, some of them accepted and forwarded to the next step, some rejected or re-run with modified inputs. 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 (see Fig. 2). 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.

The use case scenario is presented below:

1. The expert prepares the meteorological simulation – choosing boundary conditions (for which time/date) and the model configuration file, setting command line parameters.
2. The expert prepares the hydrological simulation and adjust post-processing of meteorological output: partial meteorological results ('hourly files', 48 pieces) can be immediately (i.e. before the entire meteorological simulation finishes) post-processed, visualized and hydrological simulations can be started.
3. The meteorological simulation executes. If scripts are waiting for meteorological results, on-line visualization and/or automatic hydrology simulation can still start.
4. The expert checks if the simulations (meteo- and hydrological) results are valid based on their knowledge and experience.
5. The expert chooses if the last simulation in the cascade (hydraulic-2D) is required. If so, he starts the hydraulic simulation.

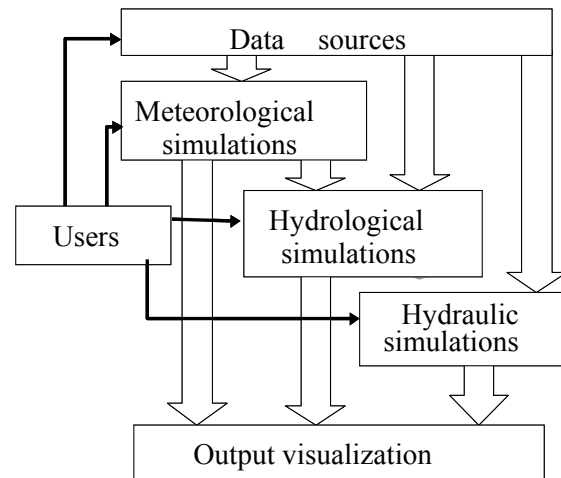


Fig. 2.Cascade of simulations

1.3. INTERACTIVE DATA ANALYSIS

Distributed data analysis in high-energy physics [23] addresses access to large distributed databases in the Grid environment and development of distributed data mining techniques suited to the high-energy physics field. Data mining services based on supervised and unsupervised learning with processing on worker nodes and on the database side will be elaborated. High Energy Physics and air pollution modeling applications require support from a Grid interactive system that allows for on-line progress monitoring of their results in order to help operators decide about further job execution (i.e. interrupting the execution or letting it finish). To track the progress of a distributed job execution on-line, the Interactive Session Manager (ISM) is proposed which communicates with the user interface using XML, sets up distributed jobs consisting of Interactive Session Workers, and collects information from all nodes to build a global result for the user. Fig. 3 shows the interaction between these components.

The use case scenario is presented below:

1. The user submits a request for access to Grid resources, via the Portal.
2. The user selects the relevant dataset (from a list built from the information contained in the Metadata catalogue) pointing out what constitutes data, and what background Monte Carlo method and signal Monte Carlo can be used.
3. The Interactive Session Resource Broker (ISRB) asks the Replica manager to provide the optimal (in network terms) physical location of the servers with the requested dataset.
4. The physicist submits the relevant variables to Grid Middleware (GM), via the Portal (i.e., once the new parameters are selected, the job is submitted).
5. The Scheduling Agents (SA) tool schedules a new simulation job taking into account the available Grid resources.
6. The Interactive Session Manager (ISM) receives the XML input from the UI and sends a simple distributed job that fills a histogram on each node, afterwards collecting the information from all nodes to build the global histogram. This information is sent back to the UI either as XML or in graphical form (SVG). The full duration of this process should be below 10 seconds.

7. The user can now train an ANN over several selected variables with a given architecture. Using the UI he selects the variables and the ANN architecture (in XML). The ISM receives the request and runs the master program with MPI calls to the Interactive Session Workers (ISW). A plot showing the relative errors in each step is updated and shown to the user via the UI.
8. The physicist monitors the progress and controls job execution; he may decide to interrupt the execution (CTRL-C) or let it finish and save (on the VO web server running the Portal) any XML flow between the UI and the ISM.
9. The physicist stops the simulation.

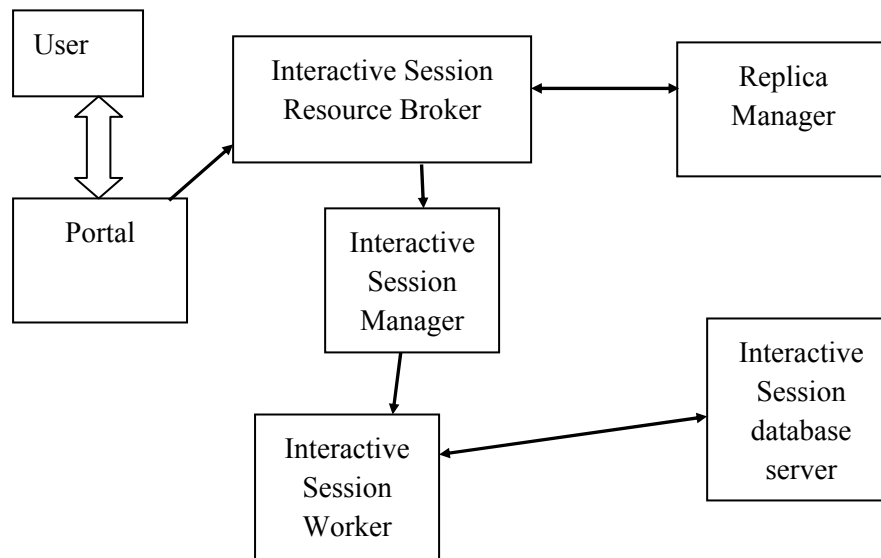


Fig. 3 Interactive data analysis

2. OVERVIEW OF CROSSGRID ARCHITECTURE

This chapter presents the second version of the architecture of the CrossGrid Project, i.e. the general overview of software components and their relationships. The components are applications, tools that support application development and new Grid services that are being elaborated within the Project. Dependencies on external components, such as DataGrid [18] and Globus software [5] are presented as well.

In the first description of the CrossGrid architecture [6] we followed the layer structure proposed in [3]. We distinguished four layers that are shown in Fig. 1. Below the applications and tools we placed Grid services that were divided into generic and application-specific.

The current CrossGrid architecture is the result of analyzing the software requirements, design documents and use cases of applications, services and tools [2] and it is presented using UML component diagrams (see Fig. 4). This figure presents dependencies between software components.

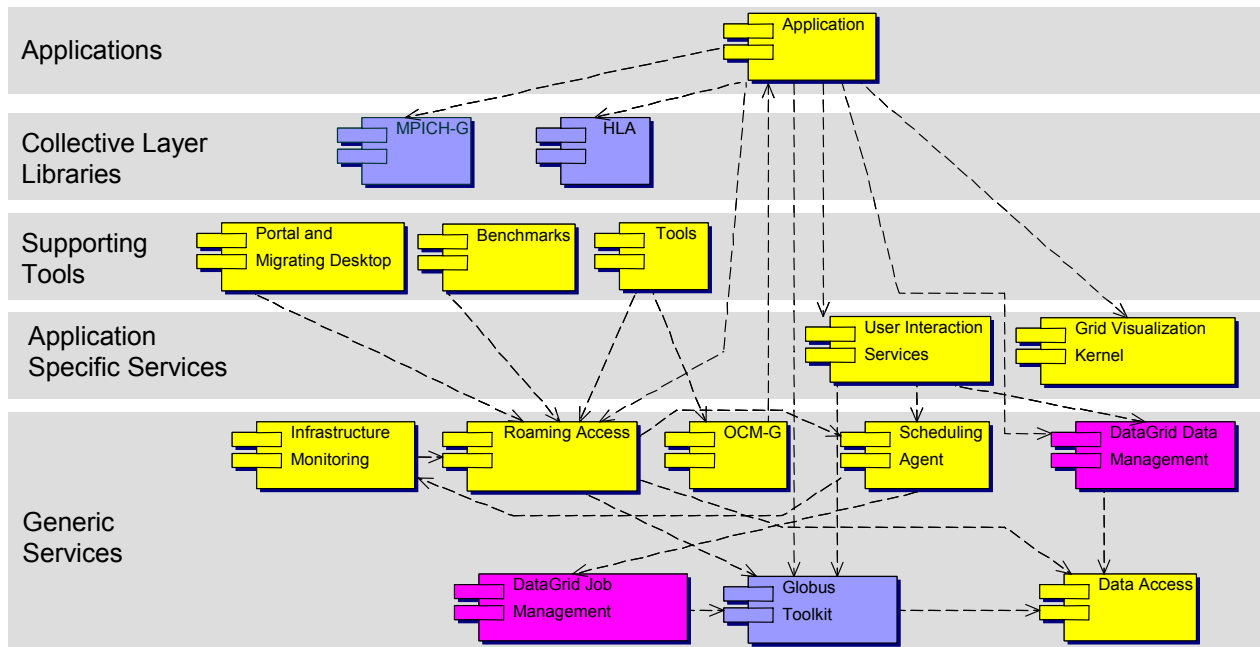


Fig. 4. Overview of the CrossGrid Architecture

3. COLLECTIVE LAYER LIBRARIES

3.1. MPICH-G

MPICH-G2 is a Grid-enabled implementation of the MPI v1.1 standard. This means that - using services from the Globus Toolkit (e.g., job startup, security etc.) - MPICH-G2 allows the user to couple multiple machines, potentially with different architectures, to run MPI applications. MPICH-G2 automatically converts data in messages sent between machines of different architectures and supports multiprotocol communication by automatically selecting TCP for intermachine messaging and (where available) vendor-supplied MPI for intramachine messaging.

MPICH-G is widely used within the CrossGrid project. All of the tools are oriented towards support of MPI applications. Most of the applications (flood prediction and interactive data analysis) will use MPICH-G as the basic collective layer library in the Grid environment.

The Grid scheduling system (WP 3.2) is aimed to support submission of parallel interactive jobs on Grid resources.

3.2. HLA

The need for an environment supporting development of complex distributed interactive simulations is an interesting issue in various fields of science, where simulation can replace experiment or help experience situations which are too expensive, impractical, or dangerous to recreate in the real world. (such as the CrossGrid medical application (WP 1.1) [21], which allows a surgeon to perform operations on patient data in virtual reality without actually bothering a patient).

The High Level Architecture (HLA) [20] for Modeling and Simulation was developed as an IEEE standard to facilitate interoperability among simulations and promote reuse of simulation components. Using HLA, a large-scale distributed simulation can be constructed using a huge number of geographically-distributed computing nodes.

HLA Runtime Infrastructure (RTI) federations are distributed systems that consist of multiple processes (federates) communicating across computer nodes. HLA provides the application developers with an RTI that acts as a tuple space. In order to send messages, the applications that are plugged into the RTI have to publish well-defined objects in that space. The applications that want to receive messages have to subscribe to those objects. Each time an object is to be sent, the application must call a method that updates the attribute values of this object; RTI then notifies subscribed applications that the object has been updated.

This approach allows for easy plugging of late arriving components into the running system. HLA also supports time synchronization (i.e. for time-driven or event-driven simulations) and data distribution management between distributed components of the interactive application.

Federates are controlled by the RTI Executive (RTIexec) process. The network address specified by the hostname or the Internet Protocol address of that machine plus the port number used, known as the endpoint of the process, is defined in a federate configuration file. If the endpoint is not specified for the RTIexec, a multicast protocol is used for finding the RTI control process. This option, however, cannot apply in general WANs and in particular on the CrossGrid testbed.

There are several issues while thinking about porting HLA into the Grid. The specification allows connection between multiple sites, therefore HLA can be used as a communication middleware between Grid sites. The firewalls and security issue depends on implementation and is not specified by the standard. Current implementation used by medical application does not support this issues, however those problems are noticed and the solutions are investigated [11].

3.3. MPICH-G VS. HLA

HLA can be seen as an API of a high-level communication library, which potentially places it on the same layer of the Grid architecture as MPI. Both standards supply the developer with an API that allows communication between distributed components of an application. The selection of library depends on application features and developer requirements. MPI is better suited for SPMD-type parallel programming, while HLA is explicitly designed as a support for interactive distributed simulations. Therefore, HLA provides various services needed for that specific purpose like time management useful for time-driven or event-driven interactive simulations. Furthermore, the MPI standard requires the specification of sender and receiver IDs, while HLA takes care of data distribution management and allows all application components to see the entire application data space in an efficient way.

such as low bandwidth and high latencies. Benchmarks will also be used to analyze and improve MARMOT performance during the second project year. Basic MARMOT dependencies are shown in Fig. 6.

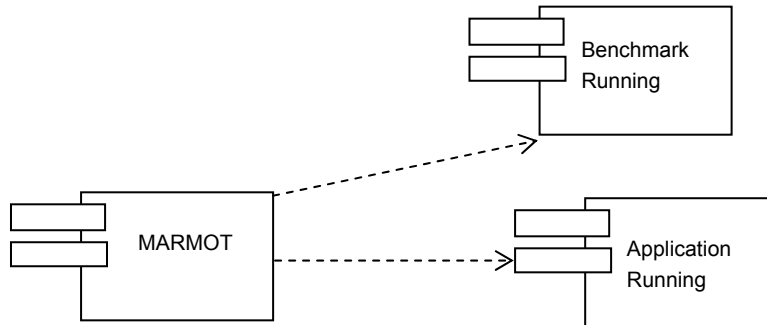


Fig. 6. MARMOT in the context of CrossGrid Architecture.

4.1.2. External Interfaces

Current application kernels are used for development and testing of the tool, and, eventually, the tool will be used to facilitate the application development process.

4.2. BENCHMARKS

4.2.1. Functionality

GridBench [25] is a software system that provides a suite of benchmarks (synthetic or application-based) for characterization of Grids. It also enables the specification of benchmarks for Grids, provides automatic generation of job descriptions suitable for execution on the CrossGrid testbed, archives the derived metrics for reference and comparison over time and publishes the derived metrics in Grid Information System (currently MDS, later R-GMA) for easy access by users or services. Basic GridBench dependencies are shown in Fig. 7.

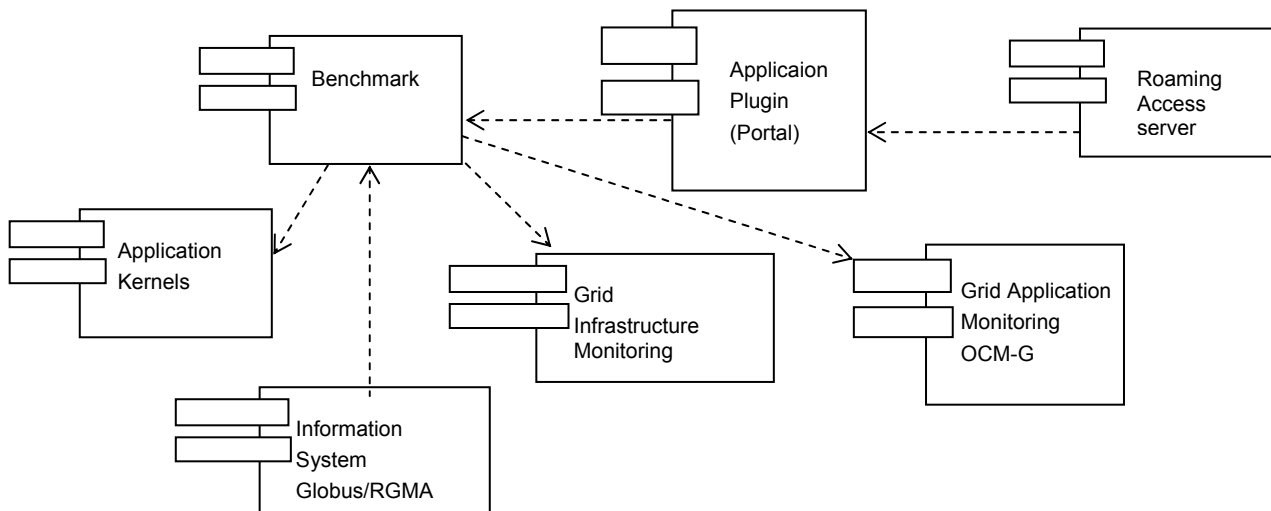


Fig. 7. GridBench in the context of the CrossGrid Architecture.

4.2.2. External Interfaces

GridBench requires the ability to submit MPI jobs through the Resource Broker. This implies that the Datagrid Job Description Language (JDL) must support MPI jobs, which it currently does not. Implementation of this is in progress (Task 3.2; Scheduling Agents) and is expected to become available soon.

GridBench also depends on the software that will be produced by Task 3.3 (Monitoring). While this monitoring software is not essential to the development or running of benchmarks, its availability will greatly enhance the user's ability to interpret benchmark results, i.e. to measure the CPU load and network traffic on the Grid resulting from the running benchmark. The interface to the monitoring data is through R-GMA and data will be collected from the infrastructure monitoring tools (SANTA-G and Jiro).

While a user interface component is to be developed for GridBench, it must be accessed through the CrossGrid Migrating Desktop. The user interface for GridBench will be a Java applet, which needs to be integrated into the portal by means of an application plugin. GridBench will rely on the availability of the OCM-G tools/API for the collection of a subset of metrics for application benchmarks. Finally, GridBench depends on the availability of kernels from WPI applications for the development of benchmarks representative of real interactive applications.

4.3. GRID PERFORMANCE MEASUREMENT TOOL

4.3.1. Functionality

The Grid Performance Measurement (G-PM) [25] tool will provide 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 input to the high level analysis component and the performance prediction component. Additionally, the tool will support application- and problem-specific analysis of performance data. On the one hand, it will allow the user to measure application-specific performance metrics, while on the other, it will provide a specification language which allows users to combine and correlate 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 visualisation component will allow requesting of performance measurements and will display the resulting performance information in various graphical ways. Basic G-PM dependencies are shown in Fig. 8.

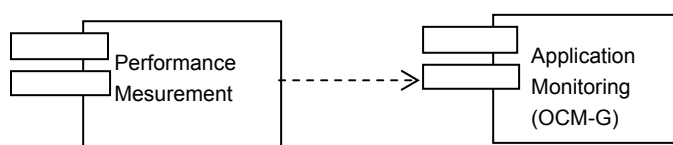


Fig. 8. G-PM in the context of the CrossGrid Architecture

4.3.2. External Interfaces

The performance analysis tool is not directly connected to the application process, but uses the OCM-G application monitoring system (see. Sec. 5.5). OCM-G and the tool communicate using the standard OMIS interface [8].

4.4. PERFORMANCE PREDICTION TOOL

4.4.1. Functionality

PPC [25] provides analytical models to predict the performance of MPI communication routines and specific application kernels in the Grid environment. PPC is intended to be used by application developers and users who are interested in analyzing the performance of selected kernels under different scenarios. The results can be used to modify the parallel execution parameters of the application, such as the number of nodes, the size of the problem, the distribution of data, etc. To enable users to quickly find their way in the multidimensional design space, PPC needs to be used interactively. In the long term, the tool can be also used by resource brokers and schedulers to select the best platform according to the predicted performance figures. This issue is beyond the scope of the project, but it could prove important in the future. Basic PPC dependencies are shown in Fig. 9.

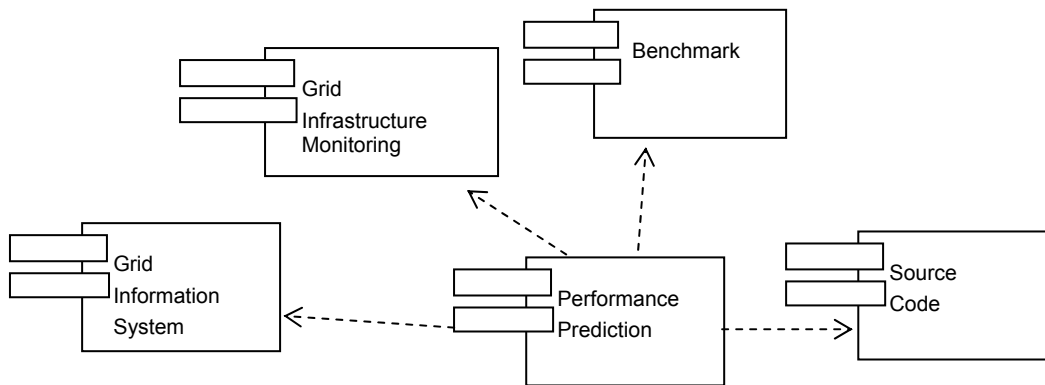


Fig. 9. PPC in the context of the CrossGrid Architecture.

4.4.2. External Interfaces

An important dependency for the development of PPC is the infrastructure monitoring component (WP 3.3). Performance prediction analysis parameters depend on parameters that characterize the Grid, and these parameters have to be obtained from Grid monitoring or from some information system. A set of kernels from Task 2.3 will be also included in this tool.

5. SERVICES

5.1. USER INTERACTION SERVICE

5.1.1. Functionality

The User Interaction Service (UIS) comprises a set of Grid Services which allow the setup and interactive steering of complex Grid applications consisting of modules for simulation and visualization. The service focuses on setting and controlling HLA [20] Runtime Infrastructure (RTI) processes that coordinate distributed application components. As a proof-of-concept example, they use the CrossGrid biomedical simulation application, which requires near-real-time steering of simulation parameters during execution. UIS dependencies are shown in Fig. 10.

The simulation and interaction of the medical application are connected by the HLA communication bus. Their execution is orchestrated by the OGSA Grid Services framework consisting of a Discovery and Information Index (for finding Grid resources with HLA installed) and the Migration Scheduler Service (for migration-related decisions based on information from Infrastructure Monitoring). The more detailed concept is explained in [12].

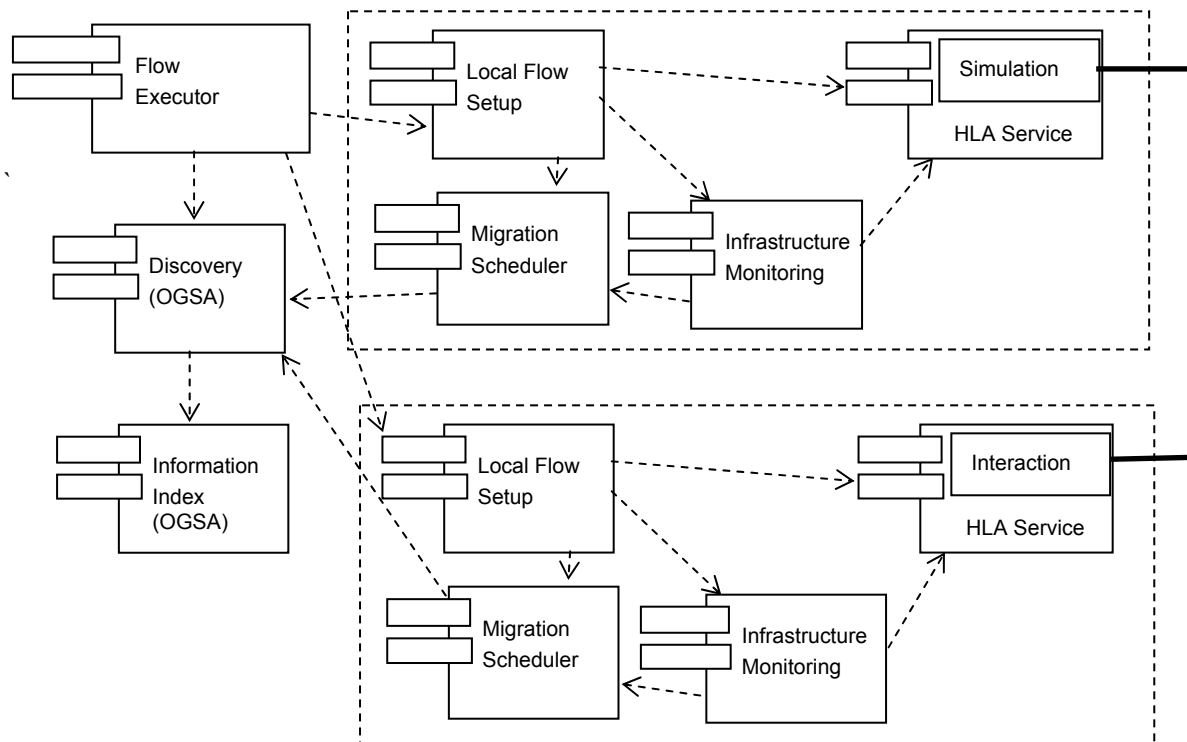


Fig. 10. UIS in the context of the CrossGrid Architecture.

5.1.2. External Interfaces

User Interaction Services support setting up and control of Task 1.1 application. They take advantage of service discovery mechanisms that are present in OGSA and in the future they will be integrated into the OGSA model, providing a robust environment for HLA-based interactive applications running in the Grid Services framework. UIS is an experimental service based on GT3 and interfaces between RAS will be defined after final decisions about migration to OGSA.

5.2. GRID VISUALIZATION KERNEL

5.2.1. Functionality

The Grid Visualization Kernel (GVK) [21] will interconnect distributed simulation with visualization clients. It will provide a uniform interface to various visualization devices and therefore allow interactive, near-real-time visualization for Grid applications. Basic GVK dependencies are shown in Fig. 11.

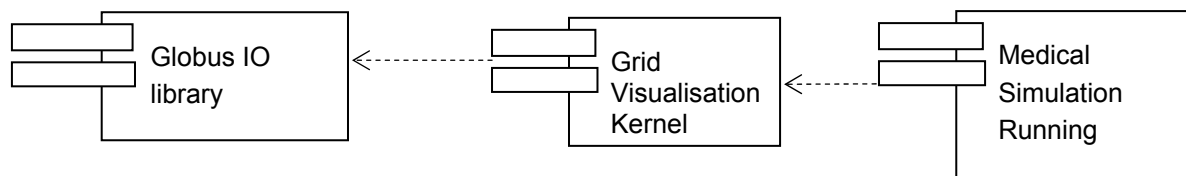


Fig. 11 GVK in the context of the CrossGrid Architecture

5.2.2. External Interfaces

The GVK will connect application elements (visualization and simulation), providing near-real-time interaction. The Globus IO library is used for communication.

5.3. PORTALS AND ROAMING ACCESS

5.3.1. Functionality

The Migrating Desktop and Portal [26] are separate front-ends which provide a set of components interfaced with the underlying Grid services, allowing users to use Grid resources. They will also offer users the possibility to access the same working environment from any workstation by means of the Roaming Access Server (RAS) using the following additional services:

- LDAP manager service – responsible for storing the user profile mechanism,
- session manager services – responsible for managing the application-user session.

Basic task dependencies are shown in Fig. 12.

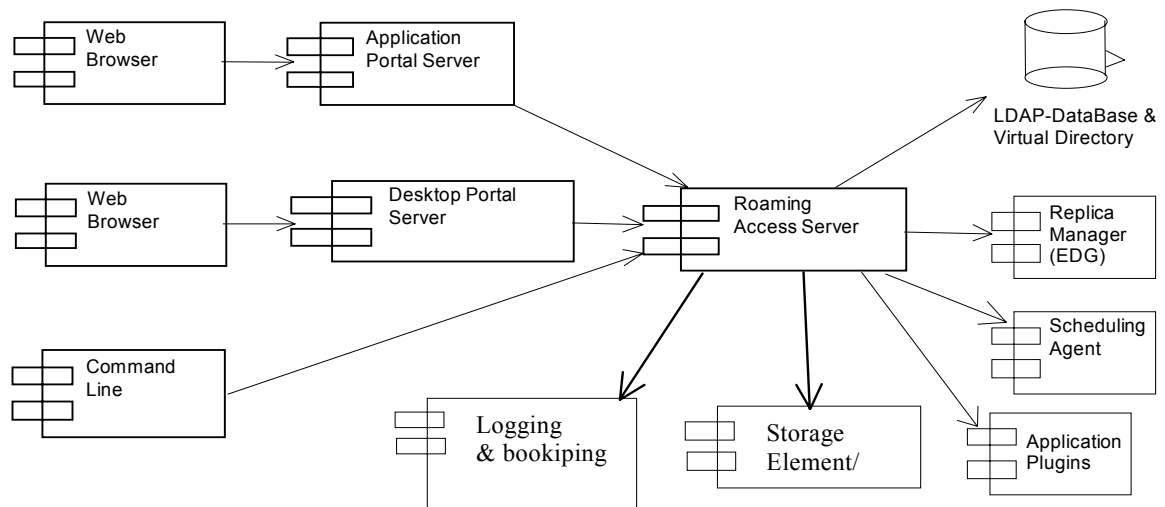


Fig. 12. Portal, MD and RAS in the context of the CrossGrid Architecture

5.3.2. External Interfaces

RAS is responsible for communicating with other CrossGrid modules used by the portal. Main external services of RAS include:

- the JobSubmission service - an interface to Scheduling Agents that allows for submitting parallel jobs in the Grid environment,
- file management services responsible for communication with Virtual Directory, Storage Elements, Private Storage and Replica Manager file copying services - responsible for communication with the Replica Manager,
- application plugin - the application developer's task is to write an application-specific plugin that can be placed in a container embedded in the portal.

The Roaming Access Server will use the Virtual Directory (based on MySQL) and DataGrid Replica Manager for managing and accessing files. The schema of interaction is as follows:

- upon receiving file request, RAS asks Virtual Directory about grid logical file name,
- RAS asks Replica Manager about best replica,
- Basing on the information obtained from the Replica Manager, Portal or Migrating Desktop downloads file using GridFTP.

Likewise when a file is being uploaded, the Replica Manager can be asked to provide a locale for storing the data, which is then copied via GridFTP to the appropriate Storage Element. Finally the corresponding entry in the Virtual Directory is updated.

5.4. GRID RESOURCE MANAGEMENT

5.4.1. Functionality

Scheduling Agents [26] are the components, which strictly depends on EDG project[18]. The task of scheduling user jobs and submitting them to Grid resources in CrossGrid is done through this SA. In the current design their role is to accept requests in the form of Job Description Language (JDL) files and, based on their optimization strategies which use information from monitoring systems, they decide where to submit the job. For actual job submission and management, DataGrid and Condor-G software is used. Basic GRM dependencies are shown in Fig. 13.

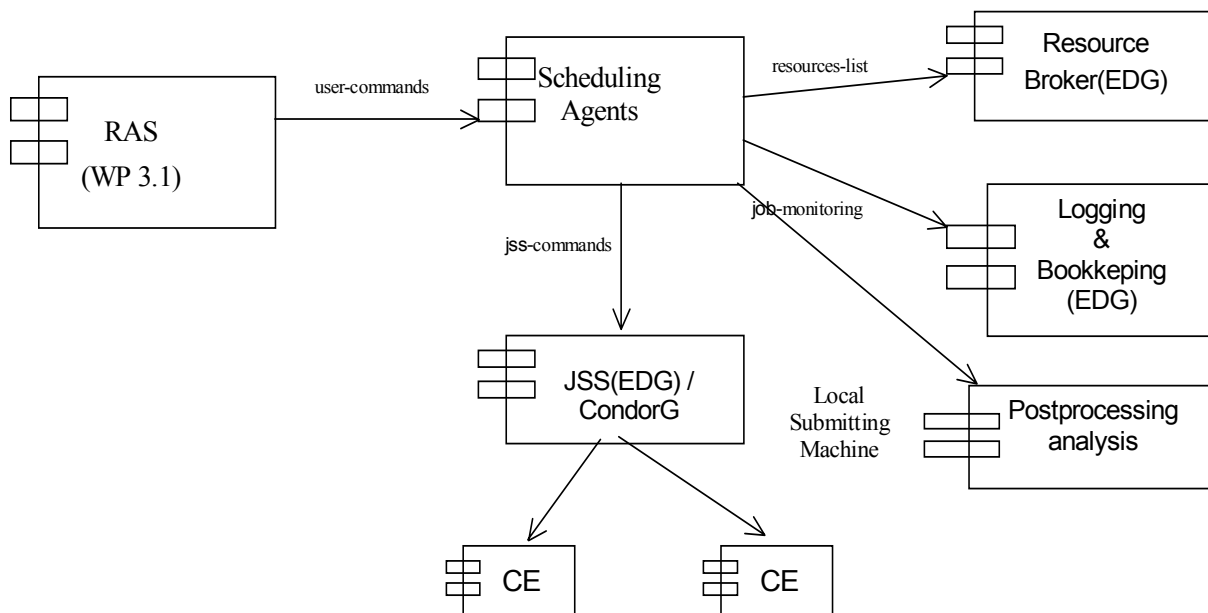


Fig. 13 GRM in the context of the CrossGrid Architecture

5.4.2. External Interfaces

As stated above, Scheduling Agents will be used by RAS to allow job submission. They, in turn, will use extended external DataGrid Software as described in detail below. Scheduling agents extend the EDG Resource Broker (RB) by providing additional functionality supporting parallel jobs (related to support for interactive applications and the “person in a loop” concept native to CrossGrid architecture).

The role of the Resource Broker is to match requests with appropriate resources on the Grid and to submit jobs for processing on these resources. In DataGrid, this is done via a special interface called a Job Description Language (JDL). JDL files constitute definitions of individual jobs and the resources needed for their completion. CrossGrid Task 3.2 will inherit the JDL concept and the associated job submission service from DataGrid, and extend it to cover new aspects of Grid processing.

The current version of JDL and the job submission service as implemented by DataGrid enables the user to submit Grid Batch Jobs (GBJ). These are accompanied by properly written JDL files and submitted to the Resource Broker for processing. The user may monitor the current status of the job, but there is no control over the execution process. Once completed, the job reports results back to the RB, which then makes them visible to the user.

5.5. MONITORING

5.5.1. Functionality

Under the common name of monitoring goes a set of services [26] used for different kinds of information gathering and processing as shown in Fig. 14. CrossGrid covers on-line monitoring of running applications [8], providing data for the performance analysis tool. There are two different infrastructure monitoring services: the Jiro-based invasive monitoring system gathers data on infrastructure and non-invasive methods (SANTA-G) are used for monitoring network traffic. Data from both services will be used by various tools as described in the previous chapter either directly or through various information system databases.

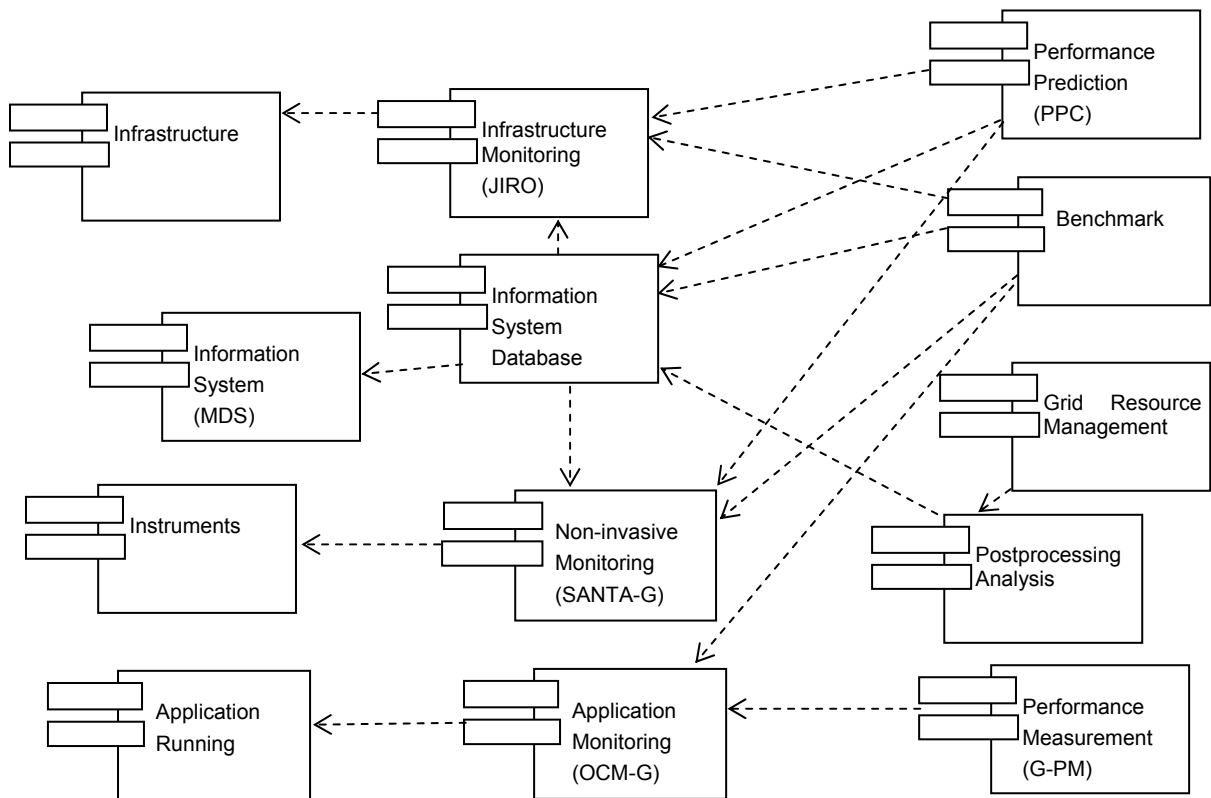


Fig. 14. Monitoring systems in the context of the CrossGrid Architecture

5.5.2. External Interfaces

Monitoring services will be used by various tools. Infrastructure Monitoring will be used by Benchmarks (2.3) and the Performance Prediction tool (2.4.2) while Application Monitoring will be used by the Performance Monitoring Tool (2.4.1). Information will be published in external Grid Information Services like MDS or R-GMA.

5.6. DATA MANAGEMENT

5.6.1. Functionality

The Data Access package [26] extends the basic functionalities of DataGrid software as shown in Fig. 15. It will extend the existing systems through optimization of access to tape-resident data and provide middleware for accessing large files stored in tape libraries. A component expert system provides the framework for deploying various access time estimators and will be integrated with the DataGrid framework for Replica Management and optimization (see Fig. 15).

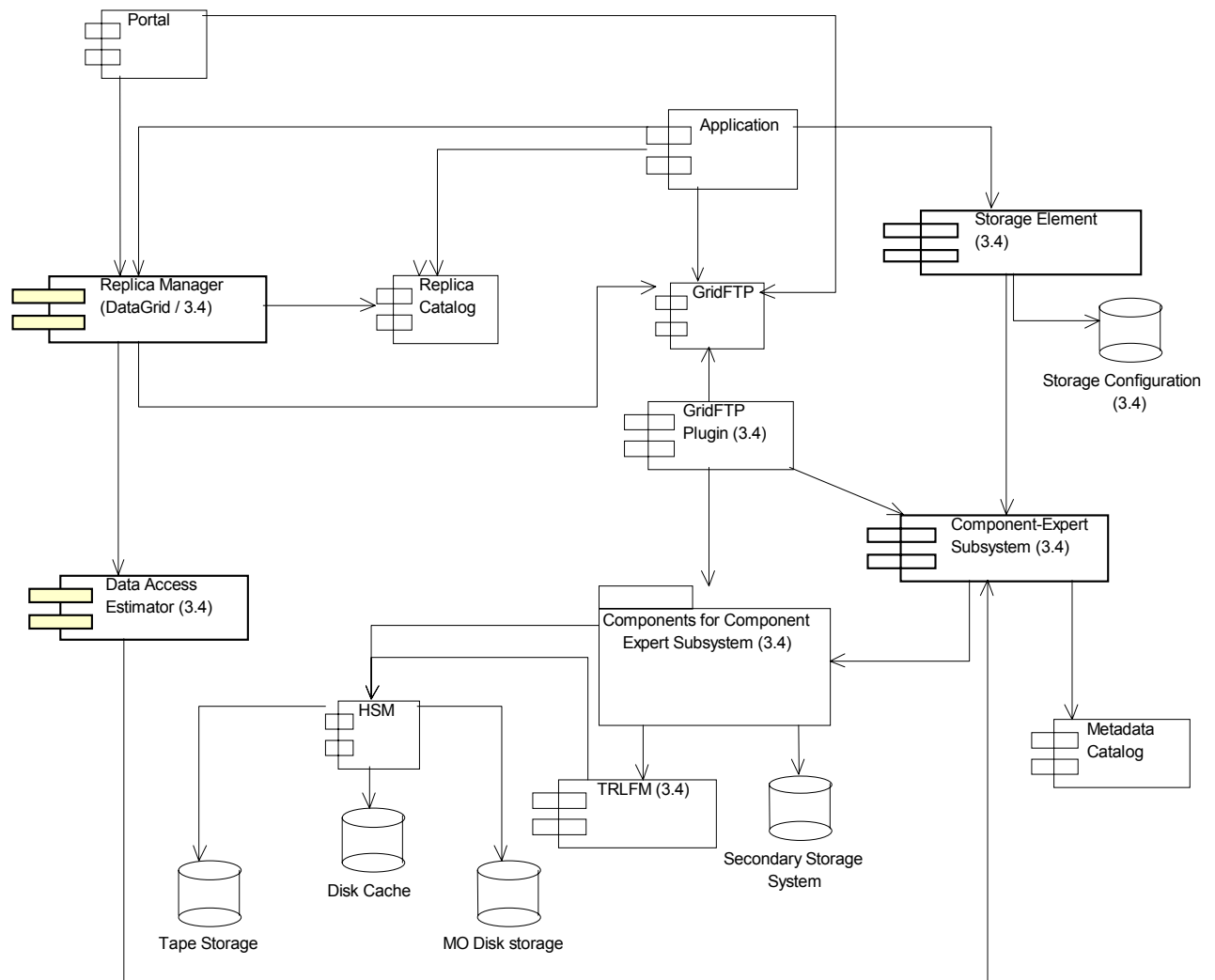


Fig. 15. Data Management in the context of the CrossGrid architecture.

5.6.2. External Interfaces

This software is placed on a relatively low level, and therefore is accessed by CrossGrid applications through higher-level services, such as GridFTP or the DataGrid Replica Manager.

There is active co-operation within CrossGrid Task 3.4, which deals with optimization of access to secondary and tertiary storage. This task is currently working on functions which would evaluate

access times to given storage systems/data sources. DataGrid will then be able to integrate this functionality in the optimization module of its own Replica Manager.

6. DEPENDENCIES BETWEEN CROSSGRID COMPONENTS

The detailed dependencies of the CrossGrid components are shown in the Tab. 1. “i” indicates an indirect dependency. Detailed descriptions of each dependency are provided in the previous sections of this document.

	2.2 MARMOT	2.3 Benchmarks	2.4.1 Perf. Anal.	2.4.2 Perf. Pred.	UIS (1.1 +5.2)	GVK (1.1)	3.1 Portal + RA	3.2 Sched.	3.3.1 OCMG	3.3.2 Santa-G	3.3.3 Jiro	3.3.4 Postproc.	3.4 Data access	Globus GT 2.x	EDG	Condor-G	Globus GT 3	MPICH-G
1.1 Biomed	x	x	x	x	x	x	x	x					i				x	
1.2 Flood	x	x	x	x			x	x			i		i					x
1.3 HEP	x	x	x	x			x	x					i					x
1.4 Pollution	x			x			x			x			i					x
2.2 MARMOT								x										x
2.3 Benchmark							x		x	x	x				x		x	x
2.4.1 Perf. Anal.									x									x
2.4.2 Perf. Pred.		x								x	x			x				x
UIS (1.1 +5.2)																	x	
GVK (1.1)														x				
3.1 Portal + RA								x						x	x			
3.2 Sched.							x					x			x	x		x
3.3.1 OCMG														x				x
3.3.2 Santa-G																		
3.3.3 Jiro														x	x			
3.3.4 Postproc.														x				
3.4 Data acc.							x							x	x			

Tab. 1. Basic dependencies between CrossGrid components.

7. CROSSGRID AND OGSA

7.1. OGSA AND GLOBUS TOOLKIT 3.0

When discussing the relationship between CrossGrid and recent developments related to Globus 3.0, one has to distinguish between OGSI, OGSA and GT3. These terms are closely related to each other and are often used as synonyms, but when it comes to details, they have very different meanings.

7.1.1. OGSA (Open Grid Services Architecture)

This name refers to the new concept for defining the Grid in terms of services, which was presented in the famed *Physiology of the Grid* [16]. It defines a Grid Service as a basic entity with specified behavior and protocols for accessing this behavior. The protocols are taken from the Web Services standards (XML, SOAP, WSDL). This architecture can be used for distributed systems integration both in e-science and in e-business areas. In GGF, the OGSA Working Group (OGSA-WG) is elaborating a document that is going to distinguish and describe a set of key services in future Grids. Such an integrated approach is described in the OGSA Platform document [14]. It includes higher-level services, i.e. Data Management, Messaging, Queuing and Logging, Events, Metering and Accounting, Transactions, Service Orchestration and Resource Management.

7.1.2. OGSI (Open Grid Services Infrastructure)

While OGSA deals with a general approach to Grid Services, OGSI is focused on the few basic functionalities, which are considered fundamental and necessary for building higher-level services. These basic services are defined in the Grid Services Specification document [17], prepared by the OGSI Working Group within GGF. The scope of this specification includes the definition of a Grid Service as an extension of a Web Service, the management of service instance's lifetime and the infrastructure for publishing, querying and notifying services of events. A service that complies with this specification is called an OGSI-compliant Grid Service.

7.1.3. GT3 (Globus Toolkit 3.0)

Globus Toolkit 3.0 includes the Java-based reference implementation of OGSI, together with an SDK and a framework for building and deploying Grid Services in the Java language. On top of this infrastructure sit the higher-level services, which provide functionalities analogous (or, in some cases extended) to those in GT2. One of them is the reimplemented GRAM service for remote job submission, as well as the redesigned information services, replacing GT2 MDS.

7.1.4. Current status of development in OGSA, OGSI and GT3

The OGSA-WG is now working on the definition of the preliminary requirements of a higher-level service-building OGSA Platform. This work is likely to evolve during the next GGF meetings, and no final statements are expected to be ready soon.

The OGSI-WG presented version 1.0 of the Grid Service Specification during GGF7 in March 2003. While this version may be considered stable and is now passing the formal process of acceptance in the GGF, the OGSI-WG is working on a new specification, taking into account the most recent W3C guidelines in the Web Services area (WSDL), which are evolving.

The Globus project is working on the implementation of the OGSI-compliant GT3. Alpha releases of OGSI have been available since mid-2002 and alpha versions of GT3 have been appearing since January 2003. *Alpha* means that the software that is not functionally complete, still unstable and may change in a considerable manner. A beta version of GT3 was released in the beginning of June 2003. As *beta*, it provides all the functionality of the final version, but will lack the stability and documentation expected of a "gold" release. The final GT 3.0 is not expected before July 2003. The GT3 development phases with respect to CrossGrid development phases are shown in the Fig. 16 .

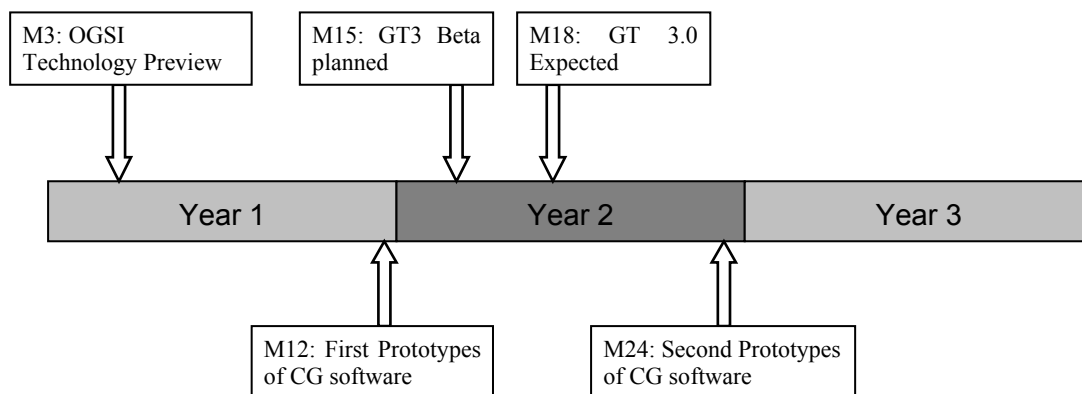


Fig. 16 GT3 development timeline vs. CrossGrid development

7.2. FUNCTIONALITY OF GT3 AND ITS RELATIONS TO SERVICES DEVELOPED IN CROSSGRID

One may ask whether the services developed in CrossGrid may overlap with those provided in the new release of Globus 3.0. The answer is negative.

First of all, CrossGrid in general aims to enable Grid computing in the area of interactive applications. Globus, on the other hand, is batch-oriented. The services for job management (GRAM), data transfer (GridFTP) and information systems (GIS) are located on the lower layer, supporting other, more advanced services, such as those developed in CrossGrid. No high-level services are provided by the Globus Toolkit as such.

Globus 3.0 does not differ in that regard from GT2. The OGSi implementation that is included in GT3 is placed on an even lower level than GT3 services. The remaining GT3 services offer the same functionality as the ones from GT2. GRAM services for job management have been completely redesigned and reimplemented using the OGSi infrastructure and XML as basis for RSL (Resource Specification Language). However, from the user's viewpoint the functionality enabling job submission to remote machines remains unchanged. Similar changes apply to Information Services – they have been redesigned to comply with the OGSi-based information model using an XML data representation, but the functionality is similar to MDS known from GT2. The third basic element of Globus (GridFTP) remains unaltered.

The higher level services that are planned to be shipped with GT3 are the Replica Location Service (RLS), the Reliable File Transfer (RFT) and DataBase Access and Integration (OGSA-DAI). CrossGrid Task 3.4 (Optimization of Data Access) is collaborating with the DataGrid project on developing RLS together with Globus. CrossGrid Task 1.3, involved in distributed data access for HEP, is evaluating the possibility of using OGSA-DAI.

Neither OGSi nor GT3 provide any services regarding portals, scheduling, application monitoring or data access optimization. The same applies to development tools from CrossGrid WP2.

7.3. OGSA-RELATED WORK WITHIN CROSSGRID

The beginning of the CrossGrid project on March 1, 2002 nearly coincided with the announcement of OGSA in February 2002. While it was impossible to take OGSA into account during the technical planning of the project, the Architecture Team has decided, from the beginning, to keep track of the most recent development of the OGSA specification and its implementation status. During the First Project Conference (Kraków, March 17-20, 2003) a presentation on that topic was given by the TAT and a working document was prepared for internal use [9]. In this document, the idea of possible application of OGSA to CrossGrid services was mentioned for the first time. As no implementation

was publicly available then and the Globus team had announced that it would not be delivered earlier than in 2003, it was decided to rely on GT2.0 during the first year of the project (initial prototypes of CrossGrid software having been scheduled for the end of 2002). A suggestion was also made to take into account the possible future migration to GT3. Such approach was expressed in the first definition of CrossGrid architecture, published in June 2002[1].

When the alpha Release of OGSA technology was made publicly available in June 2002, CrossGrid began actively testing and evaluating this technology preview. Cooperation with the Globus Team was established and confirmed by the visit of a CrossGrid representative to Argonne in September 2002. This visit resulted in the creation of an internal document [10] which contained the latest status update on the GT3 release plan and the analysis of possible migration of CrossGrid services to the OGSA Grid Service model. A discussion on that issue begun during the CrossGrid Workshop in Linz (September 2002). At the same time, the design of selected services was presented, including the use of OGSA-friendly protocols like SOAP and XML where possible.

During the 2nd Cracow Grid Workshop in December 2002 an analysis of CrossGrid and OGSA was presented and published [13]. At the same time, the first proposal of building OGSA-based services for interactive applications was presented [11]. It focused on support for the Biomedical Application developed within CrossGrid Task 1.1.

During the 2nd Project Conference and integration meeting in Santiago, Feb. 2003, the discussion on OGSA and CrossGrid continued. A presentation of the CrossGrid Architecture included the outline of possible migration in the second year of the project. GT2 continued to be the basis for the production testbed and the basic technology for software prototypes. Web Service protocols were suggested as communication interfaces where possible. Plans for migrating to HLA-based services for WP1 interactive applications were presented as well.

The first prototypes of CrossGrid software that were presented after the first year of the project at the integration meeting in Santiago used Web Services protocols where applicable. This usage was described in Deliverable D3.3 [15]. The SOAP protocol was used by Task 3.1 (Portals and Roaming Access) and Task 3.4 (Optimization of Data Access).

With the release of the alpha version of GT3, which contained basic prototypes of higher-level OGSA-based Globus services, evaluation of this software also begun.

7.4. MIGRATION TO OGSA VS. MIGRATION TO GT3

When speaking about the migration of CrossGrid to OGSA or GT3, one has to take into account the differences between the meanings of these statements and to distinguish any possible aspects of such migration. Basing on the knowledge of the topic, two different aspects of the migration can be suggested:

- migration to GT3: Using GT3 instead of GT2 as a set of basic services in CrossGrid,
- migration to OGSA: making the services developed in CrossGrid OGS-compliant.

7.4.1. Migration to GT3

Migration to GT3 meant by replacing GT2 with GT3 as the basic infrastructure in the CrossGrid testbed is more of a technical than an architectural issue. As GT3 offers basically the same functionality as GT2 and no new features that may be crucial for interactive applications, the demand for switching to GT3 is weak. However, as some of the new services developed in CrossGrid are designed with future OGSA compatibility in mind, they will eventually require GT3, which contains the OGS implementation. This translates into a need for both a stable testbed operating GT2 and an experimental testbed running GT3.

To solve these requirement issues, three aspects need to be considered:

- dependency on DataGrid,
- backward compatibility between GT3 and GT2,

- the possibility of running in parallel both installations (GT2 and GT3) on the same sites.

The first issue is the most important one. The CrossGrid project uses DataGrid software as the basis for its testbed. DataGrid plans are to maintain GT2 in their own testbed and not to migrate to GT3. Also, the projects that are planned as follow-ups to EDG are not migrating to GT3.

The second issue is not so easy to put into practice. While the functionalities of GT3 are roughly the same as GT2, the services are accessed in different way. In CrossGrid, when submitting jobs, users do not interact with Globus services directly, but by the means of intermediate services from DataGrid. Switching to GT3 would require changes in these services, which are external to CrossGrid.

A promising alternative is the possibility of running both installations of GT2 and GT3 on the same sites. Such an option has to be tested once the full implementation of GT3 is released; however it can be assumed that the move would be essentially possible. GT3 uses the same security configuration files as GT2 while the services are running on different ports. This means that one can use the same configuration and no conflicts between services will occur. It is, therefore, possible to have a stable testbed running GT2 and, at the same time, develop and test services and applications that will require GT3. It is also suggested to upgrade GT2 to version 2.4 which contains security mechanisms compatible with GT3.

Such experimental migration requires a lot of effort related with learning and testing the new technology, which is still immature and will certainly be less stable than older versions.

7.4.2. Migration to OGSA

Migration to OGSA is not only a technical question, but more of an architectural and conceptual one. OGSA assumes that the Grid is a set of dynamically-changing services accessible by standard protocols. Migrating to OGSA means:

- designing the offered functionalities in the form of services,
- making these services OGS-compliant.

In the case of CrossGrid, the new functionalities offered by tasks from WP3 were, from the very beginning, designed in the form of services. Roaming Access, Scheduling, Monitoring and Data Access expose their functionalities by means of some protocols. It follows, that from the architectural point of view the design of CrossGrid is compatible with the OGSA model.

Speaking in the language of services is not a satisfactory condition for OGS compliance. Another requirement is compatibility with the Grid Service Specification [17], which means the usage of specified protocols and conventions. This issue is more technical and in some cases can be easily addressed, while other instances may prove complicated. In the case of services that are implemented in Java, it is more straightforward to add OGS interfaces using the GT3 development framework. However, in the case of C/C++ based services it is worth noting that no C language support for Grid Services currently exists. A description of issues related to specific tasks is included in Sec. 7.6.

Another issue is that services or tools can be more general or more specific and they can also be independent from OGSA usage. This applies to many of the tools developed within WP2.

7.5. PLANS FOR THE SECOND YEAR OF THE PROJECT

Taking into account the current evaluation of OGSA and GT3, the decision for the second project year is as follows: GT2 is kept as the basic infrastructure for the testbed in order to preserve compatibility with DataGrid. An experimental installation of GT3 will be evaluated and tested by the biomedical application, which relies on GT3 to some extent. The new services and tools in WP2 and WP3 are developed in a way that will facilitate future migration to GT3 and OGSA (see Sec. 7.6). Further decisions are planned following another evaluation at the end of the 2nd project year.

7.6. OGSA ISSUES RELATED TO CROSSGRID TASKS

Comments on OGSA from the documents of the specific tasks are extracted below.

7.6.1. Task 2.2 (MPI code debugging and verification)

MARMOT is implemented as a library that has to adhere to the interface defined by the MPI standard. The implementation of a tool like MARMOT as an OGSA-compliant web service is neither planned nor useful. However, MARMOT could make use of OGSA-compliant services. One point of interest would be the retrieval of information about the runtime environment where the application is executed. MARMOT is, however, targeted to provide information of a general nature, not specific to a specific environment or platform; hence the use for this concept is rather limited.

7.6.2. Task 2.3 (Metrics and benchmarks)

OGSA was kept in mind during the revision of the design, after the first prototype and the associated experiences [27]. The updated design for GridBench provides for the publication of benchmarking results to the MDS. With the introduction of OGSA and Globus Toolkit 3.0, MDS will become obsolete, but care has been taken to design software that will be OGSA-enabled and not MDS-dependent.

A full evaluation of the potential of an OGSA-enabled GridBench suite is still in progress.

7.6.3. Task 2.4.1 (Performance evaluation tools)

The G-PM tool is basically independent of OGSA [27]. Moreover, the functionality of G-PM is a novel approach to performance evaluation, focused on the automatic detection of performance bottlenecks in interactive applications. This issue is not covered by OGSA at the moment.

G-PM is implemented as a tool that has to adhere to the OMIS interface. Since the operation of the tool is dependent on the underlying OCM-G monitoring system, which can, in principle, migrate to OGSA, this could motivate the implementation of the Measurement Interface as an OGSA service. However, a special study would be required in this regard, and only after a C++ GT3 environment becomes available.

7.6.4. Task 2.4.2 (Performance prediction)

The analytical models developed in PPC can be adapted to OGSA in a straightforward way [27]. Only the features which influence performance in the execution of the kernels have to be considered. For example, if the way in which OGSA gives a hierarchical structure to communications is different than the one in earlier Globus versions, it might imply changes in the analytical model, but not in the functionality of PPC itself.

The tool is executed locally, so it does not need to access Grid services for getting monitoring information.

7.6.5. Task 3.1 (Portals and Roaming Access)

OGSA and OGSi technology are not key issues in the area of this task. MD and the portal have nothing in common with OGSA, and RAS is only scantily involved with it [28].

The MD and the Portal, as user interfaces placed on the highest level of the CrossGrid architecture, may use OGSA technology only in a very limited way, and only as the standard of communication with RAS Web services. For the first prototype, communication between RAS and co-operating modules is based on the SOAP protocol. To generate interfaces and use the SOAP communication layer, the open-source Axis tool is used, so Task 3.1 components will be transparently compatible with the OGSA standard. Integration with the CrossGrid application and other modules migrating to OGSA, will likely require no special effort for Task 3.1 modules

Obviously, the SOAP protocol is not the only element determining whether the components are OGSA-compliant. However, Task 3.1 components can easily fulfill other requirements stated by the OGSA standard in case a migration is required.

7.6.6. Task 3.2 (Scheduling)

The services that are currently under development in Task 3.2 do not have OGSA counterparts and therefore are original in nature. Globus Toolkit 3 (GT3) provides the same set of major features as GT2. Specifically, it includes software that provides Grid security (GSI), remote job submission and control (GRAM), high-performance secure data transfer (GridFTP), and consistent interfaces to system and service information (MDS). As GT3 maintains the current GRAM functionality, external schedulers are required for managing jobs.

In terms of compatibility with OGSA, the CrossGrid scheduling agents are currently dependent on other intermediate services that go between the scheduler and Globus services[28]. In particular, instead of using GRAM services directly, Condor-G is used as a reliable job submission service. Moreover, access to MDS is not carried out directly, but through an Information Index developed in DataGrid. These two intermediate elements are complex, as they provide a key set of services and cannot be easily replaced without suffering a significant degradation of functionality. Therefore, compatibility with OGSA will require that these two services are updated and made compatible with the new interfaces that OGSA provides for GRAM and MDS services.

On the other hand, Scheduling Agents themselves can migrate to OGSA. This will require a non-negligible effort to change the current interface to a new one based on WSDL and SOAP, and turn the Scheduler Agent itself into a Web service. This change will not provide any extra functionality in terms of scheduling and planning capabilities. A more detailed analysis of OGSA technology is currently being carried out in order to implement a prototype of the Resource Selector as a Web service. At this stage, experience with Globus Toolkit 3 reveals that this version of OGSA is still quite unstable and error-prone. Hopefully, the first release will provide a more stable set of services.

7.6.7. Task 3.3 (Monitoring)

7.6.7.1. Application Monitoring

OCM-G

A recent trend in Grid technology is to implement services as Web services, one of the frameworks supporting this approach being OGSA [16]. However, in the current state of development, OGSA does not fit/support the CrossGrid approach to monitoring. First, OGSA does not yet support C/C++. Since the system is written in C (which is important for efficiency reasons), OGSA servers exposing OCM-G services cannot easily be implemented. Second, though OGSA already provides an infrastructure for secure communication on the Grid, the communication layer of the OCM-G, based on an efficient direct TCP/IP solution, is already implemented, and secure communication mechanisms are currently in the final stage of development. Another reason to develop a custom communication layer is that the efficiency provided by OGSA is unclear.

Nevertheless, the current research of Task 3.3 includes a feasibility study and the design of an interface to OMIS monitoring services based on OGSA[28]. Preliminary results of this research are expected soon.

7.6.7.2. Instruments, Infrastructure, Derived Results

Instruments: SANTA-G

The DataGrid R-GMA is aware of the expected shift towards OGSA, and is currently investigating a move to this architecture. Because R-GMA is conceptually already quite close to Web services, this move should be straightforward. As OGSA progresses, R-GMA will naturally evolve to fit this structure. A consequence of this is that through SANTA-G making use of the R-GMA, full OGSA compliancy can be achieved when the changeover to this architecture occurs. A number of presentations on R-GMA plans regarding OGSA have been prepared [28].

Infrastructure: Jiro-based Monitoring

The Jiro-based infrastructure monitoring system will expose interfaces to external systems on two layers: the agent layer and the database layer. The interface on the agent layer will be implemented as a Web service, as this will mean integration with other OGSA-based components should prove straightforward [28]. The collector will form a producer in the GMA terminology and will communicate with its sensors via Java RMI. However, as no direct communication takes place between external systems and the sensors, this should not impose any problems with integration.

The database layer will also expose Web services wrapping Enterprise Java Beans that will access the databases directly. Similarly, integration with other Web services should not introduce any problems.

Derived Results: Postprocessing

All interfaces to external modules, with respect to the Postprocessing Analysis tool, will be implemented using R-GMA. As the result of the fact that R-GMA will soon evolve to accommodate OGSA, the migration of the Postprocessing Analysis module should prove quite seamless [28].

7.6.8. Task 3.4 (Optimization of Data Access)

OGSA compliance is one of the most important features, and it has been taken into account during the whole development process [28]. First of all, the SOAP protocol was chosen as the inter-component communication language and, as a result, all Task 3.4 components use SOAP. In order to generate interfaces and use the SOAP communication layer, the open-source gSOAP tool was selected. Since work is well underway to integrate gSOAP with GSI and to make it OGSA-compliant, Task 3.4 components will be transparently compatible with the OGSA standard.

Obviously, the SOAP protocol is not the only aspect in determining whether components are compliant with OGSA. However, Task 3.4 components can easily fulfill other requirements stated by the OGSA standard should integration with GT 3.x become necessary.

8. SUMMARY

Basing on the analysis of developments of CrossGrid applications, tools and Grid services, along with the experience gained during the initial integration of the software in the CrossGrid testbed, and the as evaluation of tools and Grid middleware reported on in M15 deliverables, it follows, that all CrossGrid software components are necessary for running interactive applications on the Grid. The only possible changes, which aim for better technical focus of the developments, are as follows:

- concentration of the effort on applications which work in near-real time (biomedical, flood, HEP),
- bringing together benchmarks and performance prediction tools, as they are closely related,
- better coordination of postprocessing with the scheduling task.

The discussion on OGSA can be summarized with the following remarks:

- the project has been evaluating the development of OGSA since its inception,
- in order to preserve compatibility with DataGrid, the CrossGrid testbed is based on GT2,
- as some of the applications require OGSA, an experimental installation of GT3 is being prepared,
- the services being developed in CrossGrid are designed to be easily adaptable to the OGSI standard,
- web services protocols are used for communication purposes to facilitate future migration,
- possible migration will be discussed after the 2nd year of the project.