Gridge-GridR integration

Project Number:	FP6-2005-IST-026996
Deliverable id:	D 4.4
Deliverable name:	Gridge-GridR integration
Submission Date:	28/11/2008

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	D 4.4
Document name:	Gridge-GridR integration
Document type (PU, INT, RE)	INT
Version:	0.3
Submission date:	28/11/08
Editor: Organisation: Email:	Juliusz Pukacki PSNC pukacki@man.poznan.pl

Document type PU = public, INT = internal, RE = restricted

ABSTRACT

This deliverable presents work done in integration R environment with grid technologies deployed in ACGT testbed. The result of the activities is GridR solution that provides computational capabilities of the grid for the biostatisticians and data miners familiar with R package.

KEYWORD LIST: GRIDGE, GRMS, GDMS, GAS, R, GRIDR

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	25/11/2008	Draft	J. Pukacki
0.2	28/11/2008	Draft	D. Wegener
0.3	29/11/2008	Draft	J. Pukacki

List of Contributors

- Dennis Wegener, FhG
- Tomasz Piontek, PSNC

Contents

Executive Summary

ACGT is an Integrated Project (IP) funded in the 6th Framework Program of the European Commission under the Action Line "Integrated biomedical information for better health". The high level objective of the Action Line is the development of methods and systems for improved medical knowledge discovery and understanding through integration of biomedical information (e.g. using modelling, visualization, data mining and grid technologies). Biomedical data and information to be considered include not only clinical information relating to tissues, organs or personal health-related information but also information at the level of molecules and cells, such as that acquired from genomics and proteomics research.

ACGT focuses on the domain of Cancer research, and its ultimate objective is the design, development and validation of an integrated Grid enabled technological platform in support of post-genomic, multi-centric Clinical Trials on Cancer. The driving motivation behind the project is our committed belief that the breadth and depth of information already available in the research community at large, present an enormous opportunity for improving our ability to reduce mortality from cancer, improve therapies and meet the demanding individualization of care needs.

1. Introduction

ACGT focuses on the domain of Cancer research, and its ultimate objective is the design, development and validation of an integrated Grid enabled technological platform in support of post-genomic, multi-centric Clinical Trials on Cancer. The driving motivation behind the project is our committed belief that the breadth and depth of information already available in the research community at large, present an enormous opportunity for improving our ability to reduce mortality from cancer, improve therapies and meet the demanding individualization of care needs.

On the architecture picture of ACGT environment, there are two bottom layers representing Grid infrastructure. The lowest one is the place of basic grid services providing remote access to physical resources. The second grid layer consists of more advanced grid services responsible for resource management, data management, and security.

That kind of Grid platform can be used by any other ACGT specific services dealing with bio-research.

The important thing is that all services come under unified authorization framework based on the commonly used Grid technologies.

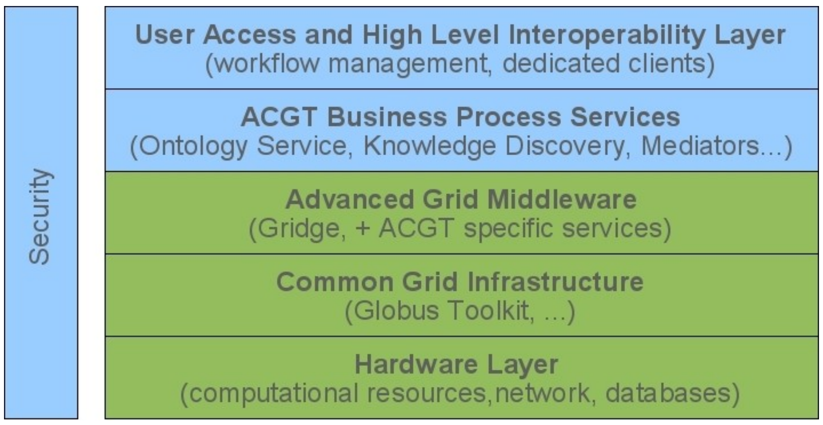
Security
User Access and High Level Interoperability Layer
(workflow management, dedicated clients)
ACGT Business Process Services
(Ontology Service, Knowledge Discovery, Mediators...)
Advanced Grid Middleware
(Gridge, + ACGT specific services)
Common Grid Infrastructure
(Globus Toolkit, ...)
Hardware Layer
(computational resources,network, databases)

Fig. 1 ACGT architecture overview

2 Gridge Technology

2.1 Overview

The Gridge Toolkit [2] is an open source software initiative which aims at helping users to deploy ready-to-use grid middleware services and to create productive grid infrastructures. All Gridge Toolkit software components have been integrated and form a consistent distributed system following the same interface specification rules, license, quality assurance and testing procedures. Gridge is a Grid-In-The-Box solution that can be easily deployed on a new infrastructure.

In detail, the Gridge tools and services enable applications to take advantage of a dynamically changing grid environment. These tools have ability to deliver dynamic or utility computing to both, the application users and developers and resource owners. Through supporting the shared, pooled and dynamically allocated resources and services, managed by the automated, policy-based Grid Resource Management System (GRMS) that interfaces with services as monitoring system, adaptive component services, data and replica management services and others, Gridge offers state of the art dynamic grid features to applications.

2.2 Resource Management

The component responsible for resource management within Gridge Toolkit is GRMS (Gridge Resource Management System). It is an open source meta-scheduling system, which allows developers to build and deploy resource management systems for large scale distributed computing infrastructures. GRMS is based on dynamic resource selection, mapping and an advanced scheduling methodology, combined with feedback control architecture, and deals with dynamic grid environment and resource management challenges, e.g., load-balancing among clusters, remote job control or file staging support. Therefore, the main goal of the GRMS is to manage the whole process of remote job submission to various batch queuing systems, clusters or resources. It has been designed as an independent core component for resource management processes which can take advantage of various low-level core services and existing technologies. Finally, GRMS can be considered as a robust system which provides abstraction of the complex grid infrastructure as well as a toolbox which helps to form and adapts to distributing computing environments.

GRMS is a central point for resource and job management activities and tightly cooperates with other services responsible for authorization, monitoring, and data management to fulfill the requirements of the applications. The main features of GRMS are job submission, job control (suspending, resuming, cancelling), the ability to chose "the best" resource for the job execution using multi-criteria matching algorithm, support for job checkpointing and migration, support for file staging, storing information about the job execution, user notifications support, workflow jobs support etc.

GRMS has been designed as an independent set of components for the resource management processes which can take advantage of various low-level core services as, e.g., GRAM [3], GridFTP [3] and the Gridge Monitoring System, as well as various grid middleware services, e.g., the Gridge Authorization Service and the Gridge Data Management Service. All these services working together provide a consistent, adaptive and robust grid middleware layer which fits dynamically to many different distributing computing infrastructures. The GRMS implementation requires the Globus software [3] to be installed on the grid resources, and uses the following core Globus services deployed on the resources: GRAM, GridFTP, and MDS (optional). GRMS supports the Grid Security

Infrastructure by providing GSI-enabled web service interfaces for all clients, e.g., portals or applications, and thus can be integrated with any other compliant grid middleware.

One of the main assumptions for GRMS is to perform remote job control and management in the way that satisfies users (job owners) and their applications requirements. All user requirements are expressed within an XML-based resource specification document and sent to GRMS as SOAP requests over GSI transport layer connections. Simultaneously, resource administrators (resource owners) have full control over owned resources on which the jobs and operations will be performed by an appropriate GRMS setup and installation. GRMS together with the core services reduces operational and integration costs for administrators by enabling grid deployment across heterogeneous (and maybe previously incompatible) cluster and resources. Technically speaking, GRMS is a persistent service within a Tomcat/Axis container. It is written completely in Java so it can be deployed on various platforms.

2.3 Data Management

Data storage, management and access in the Gridge environment are supported by the Gridge Data Management Suite (DMS). This suite, composed of several specialized components, allows building a distributed system of services capable of delivering mechanisms for seamless management of large amounts of data. It is based on the pattern of autonomic agents using the accessible network infrastructure for mutual communication. From the external applications point of view DMS is a virtual file system keeping the data organized in a tree-like structure. The main units of this structure are meta-directories, which allow creating a hierarchy over other objects and metafiles. Metafiles represent a logical view of data regardless of their physical storage location.

Data Management System consists of three logical layers: the Data Broker, which serves as the access interface to the DMS system and implement the brokering of storage resources, the Metadata Repository that keeps information about the data managed by the system, and the Data Container, which is responsible for the physical storage of data. In addition, DMS contains modules which extend its functionality to fulfill the enterprise requirements. These include the fully functional web based administrator interface and a Proxy to external scientific databases. The Proxy provides a SOAP interface to the external databases, such as for example those provided by SRS (Sequence Retrieval System).

The Data Broker is designed as an access point to the data resources and data management services. A simple API of the Data Broker allows to easily access the functionality of the services and the stored data. The Data Broker acts as a mediator in the flow of all requests coming from external services, analyses them and eventually passes to the relevant module. The DMS architecture assumes that multiple instances of the Data Broker can be deployed in the same environment, thus increasing the efficiency of data access from various points in the global Grid environment structure.

The Metadata Repository is the central element of the Gridge distributed data management solution. It is responsible for all metadata operations as well as their storage and maintenance. It manages metadata connected with the data files, their physical locations and transfer protocols that could be used to obtain them, with the access rights to the stored data and with the meta-descriptions of the file contents. Currently each DMS installation must contain a single instance of the Metadata Repository, which acts as a central repository of the critical information about the meta-catalogue structure, user data and security policy for the whole DMS installation.

The Data Container is a service specialized towards the management of physical data locations on the storage resources. The Data Container API is designed in a way to allow easy construction and participation in the distributed data management environment of storage containers for different storage environments. The Data Containers currently available in the DMS suite include a generic file system Data Container, a relational database Data Container and a tape archiver Data Container. The data stored on the various storage resources can be accessed with one of the many available protocols including such as GASS, FTP and GridFTP.

The Proxy modules are services that join the functionality of the Metadata Repository allowing to list the available databanks, list their content, read the attached metadata attributes and to build and execute queries, and of the Data Container to provide the data using the selected data transfer protocol. Such Proxy container are highly customized towards the specific platform they are working with to allow building complex queries and executing operations on the found entries.

2.4 Security Services

The most important element of security infrastructure in Gridge is authorization service called GAS. The Gridge Authorization Service (GAS) is an authorization system which can be the standard decision point for all components of a system. Security policies for all system components are stored in GAS. Using these policies GAS can return an authorization decision upon the client request. GAS has been designed in such a way that it is easy to perform integration with external components and it is easy to manage security policies for complex systems. The possibility to integrate with the Globus Toolkit and many operating system components makes GAS an attractive solution for grid applications.

Generally, an authorization service can be used for returning an authorization decision upon the user request. The request has to be described by three attributes: user, object and operation. The requester simply asks if the specific user can perform the operation on the specific object. Obviously, the query to an authorization service can be more complex and the answer given by such service can be complicated, too. One of the services which can work in such scenario is the Gridge Authorization Service (GAS). GAS has been designed in a form which enables many possible applications. GAS can communicate in many ways with other components. By using the modular structure of GAS it is easy to write a completely new communication module. The GAS complex data structure can be used to model many abstract and real world objects and security policies for such objects. For example, GAS has been used for managing security policies: for many Virtual Organizations, for services (like Gridge Resource Management Service, Mobile Services and other) and for abstract objects like communicator conferences or computational centers. These and many other features give a possibility to integrate GAS with many existing solutions. Such integration can be very important, because it raises the security level of the existing solutions and makes it possible to use the newest security technologies.

The main goal of GAS is to provide a functionality that would be able to fulfill most authorization requirements of grid computing environments. GAS is designed as a trusted single logical point for defining security policy for complex grid infrastructures. As flexibility is the key requirement, it is to be able to implement various security scenarios, based on push or pull models, simultaneously. Secondly, GAS is considered as independent of specific technologies used at lower layers, and it should be fully usable in environments based on grid toolkits as well as other toolkits. The high level of flexibility is achieved mainly through the modular design of GAS and usage of a complex data structure which can model many scenarios and objects from the real world. It means that GAS can use many different ways for communication with external components and systems, use many security data models and hold security policy on different types of storage systems. These features make GAS attractive for many applications and solutions (not only for those related with grids). GAS has to be the trusted component of each system in which it is used and it brings about that the implementation of GAS was written in ANSI C. This choice makes GAS a very fast and stable component which uses not much CPU power and little amount of memory. The main problem of many authorization systems is their management. It is not easy to work with a complex system in a user-friendly way. Based on many experiences and the end user comments together with GAS, the GAS administration portlet (web application) is provided, which makes management as easy as possible. Flexibility of this solution gives users a full possibility of presenting only these security policies which are important for them.

3 Gridification of R environment - GridR

GridR [1] consists of an R package and a web service which allow using the statistical software R [1] in a grid environment. In the following, both components are described with respect to the integration with the ACGT grid layer.

3.1 The GridR package

The GridR package provides an R programming language interface that supports the access to the ACGT services. This means that R users and developers will have access to distributed resources in a transparent fashion, as if those resources were local. The complexity of the grid is thus hidden from the user. Accessing the ACGT grid environment requires no changes in the core R implementation. In practice grid access is performed through the call of predefined R functions loaded from a package. R users can make use of the grid technology in a transparent way by initializing the grid environment once (grid.init) based on a pre-defined configuration and passing the functions to be executed in the grid as input to one of those predefined functions (grid.apply) in their local code.

In the following the process of executing a single R function in such a grid environment is described [3]:

· Function loading. The GridR functions are loaded from the GridR package into the workspace of the R client.

· Grid initialization. The grid environment is initialized by calling the function grid.init. This function sets all needed settings for the client side components of the resource management systems to contact (e.g., including the information on the Myproxy) as well as some temporary directories to use on the remote hosts that will be taken as execution machine. To avoid the specification of all setting on each submission once again a configuration file can be used.

· Code writing. The R code which is to be executed in the grid is written and wrapped as single R function in the local R environment.

· Grid submission. The grid.apply function is called, which launches the process of submission. At first, the function to be executed in the grid and the needed parameters are written into a file (uniqueID-fx). Then the R script which is actually executed on the remote machine is generated (uniqueIDRemoteScript. R).Next an R file is created, which specifies the “workflow” which is performed on the client side (uniqueID-LocalScript.R). After that, the R client executes this file, which results in connecting to the ACGT services including the Gridge components responsible for data and resource management. During the GridR stage-in phase, all files needed for the job submission (uniqueID-fx and uniqueIDRemoteScript. R) are uploaded to the grid and a GRMS job is generated. The GRMS system then takes care of staging-in files to the execution machine, launching the processing and managing the handling of the results. During the remote execution the created R script (uniqueID-RemoteScript.R) is executed on the remote machine, which reads the parameters and the function from uniqueID-fx, executes y=f(x) and writes the result or any errors into a file (uniqueID-y.dat).

· Waiting for result. There are two ways of waiting for the result, a blocking one and a non-blocking one (specified with wait=TRUE or FALSE). While the remote execution is active and the R client waits for result (by checking if the file y.dat is created) the variable y is locked or if the blocking mode is used, R is blocked until the result is available.

· Result processing. When the result file (uniqueID-y.dat) was created on the remote machine it is, together with the other result files, transferred back to the client during the stage-out phase. In the GridR client, this file is loaded. The exit status is checked and – if the job was successful – the value is assigned to y and the variable is unlocked.

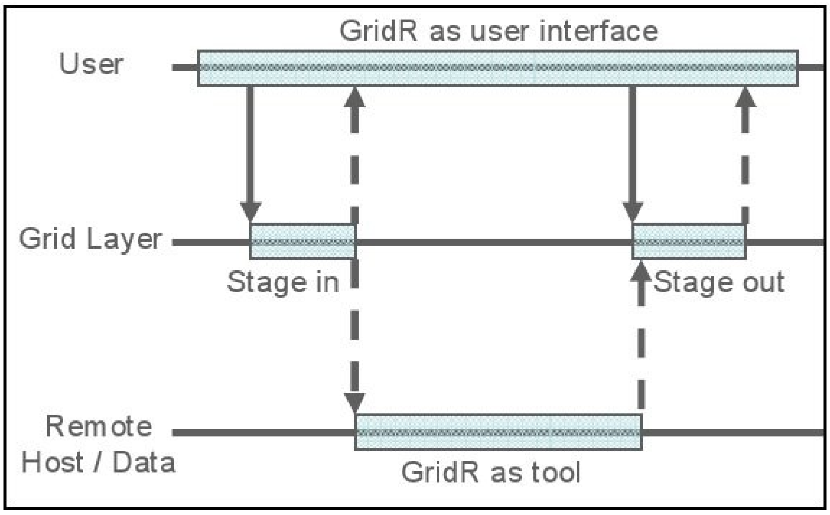

Fig. 2 Steps of execution of R function in the Grid

3.2 The GridR service

In the following, we will focus on the web service component of GridR [4], which is supposed to make the GridR environment usable from within an ACGT workflow. The interface of the GridR service supports the execution of user-defined scripts as well as the execution of scripts that had been pre-registered in a repository. In the ACGT platform, the GridR service is implemented as a GSI-secured grid service based on the Globus Toolkit 4 libraries [1] and on the Gridge Toolkit [5]. In detail, the GridR service includes clients to the Gridge data management system (DMS), which is a virtual file system for data organization in grid environments, and to the Gridge grid resource management system (GRMS), which is responsible for grid resource management and scheduling of grid jobs.

The interface to the DMS is based on files; this implies that all input and output data have to be passed to and from GridR by physical files. For this purpose, the GridR service attaches a header to each script which makes the contents of input files accessible in the R session on the execution machine as elements of a predefined R list. The interface for the output is a list of file or directory names that the user can use to export data from the session.

The developer of a parallel GridR script is offered a “directive”-like mechanism for the annotation of the parts of the script that can run in parallel. With the help of these annotations, the GridR service can split the script into parallel or non-parallel sections that are or are not to be run in parallel as grid jobs.

Technically, the GridR service translates the parallelization information into a complex GRMS job description representing the workflow to be performed for the execution of the parallel and non-parallel script sections (e.g., including the dependency of the different script sections or the specification of which files have to be staged in and out), and submits it as GRMS job.

References

[1] D. Wegener, T. Sengstag, S. Sfakianakis, S. Rüping, A. Assi, “GridR: An R-based grid-enabled tool for data analysis in ACGT clinico-genomic trials”. In: Proc. of the 3rd IEEE International Conference on e-Science and Grid Computing (eScience 2007), Bangalore, India, 2007, pp. 228-235.

[2] R Development Core Team (2005), “R: A Language and Environment for Statistical Computing”, R Foundation for Statistical Computing, Vienna, Austria

[3] Dennis Wegener; Thierry Sengstag; Stelios Sfakianakis; Stefan Rueping; Anthony Assi; „GridR: An R-based tool for scientific data analysis in grid environments“, accepted for publication in FGCS

[4] D. Wegener, T. Sengstag, S. Sfakianakis, S. Rüping, “Supporting parallel R code in clinical trials: a grid-based approach”. Accepted for publication at the HiPGCoMB 2008 workshop

[5] I. Foster, “Globus Toolkit Version 4: Software for Service-Oriented Systems”, IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006

[6] J. Pukacki, M. Kosiedowski, R. Mikołajczak, M. Adamski, P. Grabowski, M. Jankowski, M. Kupczyk, C. Mazurek, N. Meyer, J. Nabrzyski, T. Piontek, M. Russell, M. Stroiński, M. Wolski “Programming Grid Applications with Gridge”, Computational Methods in Science and Technology vol. 12, Poznan 2006.

Appendix A - Abbreviations and acronyms

SOA	Service Oriented Architecture
GRMS	Gridge Resource Management System
GAS	Gridge Authorization Service
GDMS	Gridge Data Management System
RFT	Reliable File Transfer
MDS	Monitoring & Discovery Service
WSDL	Web Service Definition Language