IST-2004-004475

DataMiningGrid

Data Mining Tools and Services for Grid Computing Environments

Specific Targeted Research or Innovation Project
2.3.2.8 Grid-based Systems for Complex Problems Solving

**D52: Implementation of Grid Resource Management and Data Movement with Grid Mining Workflow**

Due date of deliverable: M20 (30 April 2006)
Actual deliverable submission date: 5 June 2006

**Start date of project:** 1 September 2004  **Duration:** 24 months

Technion – Israel Institute of Technology (TECH)

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>Public</td>
</tr>
<tr>
<td>PP</td>
<td>Restricted to other programme participants (including the Commission Services)</td>
</tr>
<tr>
<td>RE</td>
<td>Restricted to a group specified by the Consortium (including the Commission Services)</td>
</tr>
<tr>
<td>CO</td>
<td>Confidential, only for members of the Consortium (including the Commission Services)</td>
</tr>
</tbody>
</table>

Revision: 06
Deliverable D52: Implementation of Grid Resource Management and Data Movement with Grid Mining Workflow
DATA MINING TOOLS AND SERVICES FOR GRID COMPUTING ENVIRONMENTS

Deliverable D52: Implementation of Grid Resource Management and Data Movement with Grid Mining Workflow

Responsible author(s): Valentin Kravtsov
Co-author(s): Martin Swain, Thomas Brolly, Vlado Stankovski, Thomas Niessen

Project funded by the European Community under the "Information Society Technology" Programme Contract IST-2004-004475
Revision History

The DataMiningGrid ® Consortium has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

<table>
<thead>
<tr>
<th>Date</th>
<th>Edited by</th>
<th>Status</th>
<th>Changes made</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.10.2004</td>
<td>Jernej Trnkoczy,</td>
<td>Plan</td>
<td>Report template setup and pre-filled</td>
</tr>
<tr>
<td>02.04.2006</td>
<td>Werner Dubitzky</td>
<td>Plan</td>
<td>Update of template</td>
</tr>
<tr>
<td>05.04.2006</td>
<td>Valentin Kravtsov</td>
<td>Draft</td>
<td>Filling the main paragraphs of the paper</td>
</tr>
<tr>
<td>06.04.2006</td>
<td>Martin Swain</td>
<td>Draft</td>
<td>Filling chapter 3 – Data Services Integration</td>
</tr>
<tr>
<td>30.04.2006</td>
<td>Valentin Kravtsov</td>
<td>Working</td>
<td>Submission to the quality manager</td>
</tr>
<tr>
<td>15.05.2006</td>
<td>Werner Dubitzky</td>
<td>Final</td>
<td>Minor modifications</td>
</tr>
<tr>
<td>22.05.2006</td>
<td>Alice McQuillan</td>
<td>Submitted</td>
<td>English expression and grammar check</td>
</tr>
</tbody>
</table>

Notice that other documents may supersede this document. A list of latest public DataMiningGrid deliverables can be found at the DataMiningGrid Web page at www.DataMiningGrid.org/dissemination.

Copyright

This report is © DataMiningGrid Consortium 2006. Its duplication is restricted to the personal use within the consortium, funding agency and project reviewers.

Citation


Acknowledgements

The work presented in this document has been conducted in the context of the EU Framework Programme VI project IST 2004 004475 DataMiningGrid. DataMiningGrid is a 24-month project that started on September 1st, 2004 and
D52: Implementation of Grid Resource Management

is funded by the European Commission as well as by the industrial partners. Their support is appreciated.

The partners in the project are University of Ulster (UU), Fraunhofer Institute for Autonomous Intelligent Systems (FHG), DaimlerChrysler (DC), Israel Institute of Technology (TECH) and University of Ljubljana (LJU). The content of this document is the result of extensive discussions within the DataMiningGrid© Consortium as a whole.

More information

Public DataMiningGrid reports and other information pertaining to the project are available through DataMiningGrid public project Web site under www.DataMiningGrid.org.
Executive Summary

This document specifies the implementation details of the integration of the main architecture components and services of DataMiningGrid system into the workflow framework. It describes the way users will utilise all the main services of the system, the relation between the services in terms of data flow and discusses the technique used to present those services to users.

The DataMiningGrid system offers data miners, who are usually not grid experts, the ability to easily construct data mining tasks and execute them on the grid. The system can be roughly divided to two main components: (a) a user-friendly environment for defining complex data mining tasks (GUI), and (b) a grid middleware that supports execution of such tasks, while utilising mechanisms for managing data and computational resources (engine). This document describes how the above components (a) and (b) are integrated.

Deliverable D32 explains in detail all the aspects of DataMiningGrid workflow framework. It provides the rationale why the Partners selected Triana as the main workflow editor and manager tool for this project and outlines all its key features and functions. Deliverable D51 describes the implementation of the Resource Broker service, its functionality, inputs and interfaces, jobs description files, and the execution stages involved. Deliverable D21 outlines the Data Service and Data Interfaces implementation of the DataMiningGrid system, focusing on data management aspects.

This deliverable is based on all three deliverables (D21, D32 and D51), describing the details of the integration process. Here we show that the developed stand-alone services and components can be combined together in order to execute complex tasks while hiding grid-related details from the system’s users. This is particularly important to users that are not grid experts.
Table of Contents

Executive Summary .................................................................................................................. 6
Table of Contents .................................................................................................................... 7
1  Introduction ............................................................................................................................. 9
2  Integration of the Resource Broker ....................................................................................... 10
   2.1  Resource Broker client as Triana unit ............................................................................... 10
        2.1.1  Inputs ....................................................................................................................... 11
        2.1.2  Unit functionality .................................................................................................... 12
        2.1.3  Outputs ................................................................................................................... 12
3  Data Services Integration ........................................................................................................ 14
   3.1  Data services as Triana units ............................................................................................ 14
        3.1.1  Inputs ....................................................................................................................... 15
        3.1.2  Unit functionality .................................................................................................... 16
        3.1.3  Outputs ................................................................................................................... 16
4  Integration of the Application Discovery and Control Centre .............................................. 17
   4.1  Application query unit ....................................................................................................... 17
        4.1.1  Inputs ....................................................................................................................... 17
        4.1.2  Unit functionality .................................................................................................... 18
        4.1.3  Outputs ................................................................................................................... 18
   4.2  Application selector unit .................................................................................................... 18
        4.2.1  Inputs ....................................................................................................................... 18
        4.2.2  Unit functionality .................................................................................................... 18
        4.2.3  Outputs ................................................................................................................... 18
   4.3  Control Centre Unit .......................................................................................................... 18
        4.3.1  Inputs ....................................................................................................................... 18
4.3.2 Unit functionality ................................................................. 19
4.3.3 Outputs .................................................................................. 19

5 Conclusions and Future Work ...................................................... 20

References ..................................................................................... 22

Abbreviations .................................................................................. 22
1 Introduction

It is clear that any solution that will eventually be adopted by data mining researchers needs to focus on user-friendly interfaces that hide detailed grid aspects from the user. The success of such a grid-based data mining framework is dependent on the successful integration of all DataMiningGrid components into a user-friendly, highly flexible and extensible graphic user interface.

As it was already mentioned in previous deliverables, the partners selected Triana as the major user interface to the DataMiningGrid system. Triana is a graphical environment that allows powerful computer programs, or workflows, to be constructed from a set of building blocks or components. Construction of the workflows is performed by selecting the building blocks required from a menu, dragging them onto a workspace panel, and connecting them together in order to define the flow of execution. Triana is designed to be used by people who have little or no experience in computer programming. One of the advantages of Triana workflows is that they represent a type-safe system. Each unit has an input and an output data type and only units with the compatible output/input data types may be connected. This feature prevents the user from combining units in an incorrect order. Thus, all the units described in this document will be annotated with their input and output data types.
2 Integration of the Resource Broker

In a distributed system, a resource broker is an essential component that will make "intelligent" decisions regarding various grid aspects, such as the location of job\(^1\) execution, destination of data movement, error handing, error propagation etc. The design and the implementation of the resource broker was thoroughly discussed in Deliverable D51 (Implemented Adaptation of a Grid Resource Manager for Data Mining Tasks). In this document, we will describe the DataMiningGrid Resource Broker’s integration into DataMiningGrid workflow framework.

2.1 Resource Broker client as Triana unit

Following the design document, the DataMiningGrid Resource Broker was implemented as a WSRF-compliant service, which exposes its functionality through Web interfaces. The main client of the Resource Broker service is the JobLauncher Triana unit. Below we will describe implementation details of the unit. The first version of the JobLauncher unit is depicted in Figure 1.

\(^{1}\) Usually, a grid “job” is a binary executable or command to be run in a remote resource (machine). The remote server is sometimes referred to as a "contact" or "gatekeeper". When a job is submitted to a remote gatekeeper (server) for execution, it can run in two different modes: batch and non-batch. Usually, when a job runs in batch mode, the remote submission call will return immediately with a job ID, which can later be used to obtain the output of the call. In non-batch job submission mode, the client will wait for the remote gatekeeper to follow through with the execution and return the output. Batch mode submission is useful for jobs that take a long time, such as process-intensive computations.
2.1.1 Inputs

The input of the unit is the JobDescription data type. JobDescription represents the job(s) that need to be matched to and executed by the resources available in the grid. Currently, job description is presented in XPML format, which is the native job description format of the GridBus Resource Broker. However, in future versions, when the GridBus team has implemented JSDL\(^2\) interfaces, the JobDescription-compliant format will be transferable to the JSDL-compliant format. The XPML is a very flexible and rich description language, which meets a wide range of the requirements, without complicating the implementation of the JobDescription unit. A sample job description in the XPML format is presented below:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xpml xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="/..\xml\XPMLSchema.xsd">
  <parameter name="X" type="integer" domain="range">
    <range from="1" to="5" type="step" interval="1"/>
  </parameter>
  <parameter name="time_base_value" type="integer" domain="single">
    <single value="0"/>
  </parameter>
  <task type="main">
    <copy>
```

\(^2\) Job Submission Description Language
The job description example contains two main sections: (a) the definition of parameters, and (b) the job execution description. The definition of parameters determines the number of times the job is executed and its parameters, whereas the job execution description represents the file staging in/out tasks and command execution tasks.

### 2.1.2 Unit functionality

The JobDescription unit’s functionality can be seen as the main client to the Resource Broker service. This client is built so as to hide the grid complexity from the users. The main functionality of the unit is divided into three steps: (a) job(s) submission, (b) job(s) monitoring, and (c) result location propagation. The execution of the unit is started immediately when the JobDescription is received in the composed workflow from the preceding unit. This description is submitted through a Web interface to the Resource Broker service for further execution of the associated job or jobs. From that moment on, the user can monitor the execution progress of the job until it is completed (either successfully or unsuccessfully). If the job execution fails, the whole workflow execution is terminated and the error is propagated to the user for further error handling. Otherwise, if all job(s) are successfully completed, the URI location of the results is passed to the next unit in the workflow.

### 2.1.3 Outputs

The JobDescription unit has two output data types: URI and the Provenance data type. The URI represents the location of the results stored as files in a pre-defined location (usually it will be a GridFTP file server). These files include standard output and standard error recorded during each jobs execution, as well as the (results) files produced directly by user’s jobs. The Provenance data type includes all the information regarding the job execution, such as:

- Data set file format (flat file/database table);
- Location on data server (file/table location);
- Information on data set size;
- Editor/producer information (enterprise/organisation or person);
- Description of test data, including statistics describing performance like precision and recall (if applicable);
• Information on required computing resources;
• All the execution parameter values.
3 Data Services Integration

The DataMiningGrid Data Services have been thoroughly discussed in deliverable D21(1). They can be conceptually divided into two types of service:

1. **Core data management services**: These are data management services that are common to many grid environments, and that cover many of the essential data services currently required by the demonstrators. These services can mostly be addressed by existing technology.

2. **Data-mining-aware data services**: These focus on enabling data mining to be performed in a grid environment and are developed by this project. They will facilitate well-defined and more efficient interfaces between the data sources and the data mining algorithms, and will contain generic components that can be used by other data mining grids.

Here, we describe the integration of the data services units into the Triana workflow.

### 3.1 Data services as Triana units

OGSA-DAI [OGSA-DAI04] is the underlying data middleware technology used by the DataMiningGrid Data Services (see Deliverable D21(1)). It provides powerful functionality for interacting with distributed and heterogeneous data resources. As described in D21(1), there are many different Triana units required to integrate all the functionality (and flexibility) of OGSA-DAI: using these units, a user can construct “mini-workflows” in order to select, access, transform, merge, update and transfer data.
3.1.1 Inputs

The Triana units require a wide range of user-generated inputs. Such inputs are broadly grouped into the following types of functionality depending on the type of data resource:

- Relational databases, e.g., an SQL query;
- XML databases: e.g., an Xpath statement;
- Files: e.g., commands to list directories or append data;
- Data delivery: e.g., using HTTP or GridFTP;
- Data transformations: e.g., compression with Gzip or an XSLT transformation;
- Miscellaneous functionality.

In order to access a data service, an OGSA-DAI client needs a document called a Perform Document. Such a document essentially lists a number of instructions, like those listed above, for a data service or data resource to execute. An OGSA-DAI client must also know the relevant information required to locate a data service (i.e., a URL and data resource name).
In addition to OGSA-DAI, a user may use Triana to transfer files with GridFTP. This data transfer interfaces directly to GridFTP (it does not use OGSA-DAI). Its input will be a directory or list of files on the client machine, and a GridFTP server where the data should be transferred.

### 3.1.2 Unit functionality

There are three types of units:

1. Units that write a portion of the Perform Document file that will be processed by the OGSA-DAI data services. These units simply write some XML lines to a file;
2. Units that interface to an OGSA-DAI client and execute that client. Typically, a client will send a Perform Document to an OGSA-DAI service;
3. Units that interact with a GridFTP client and issue commands to the GridFTP client in order to transfer files.

### 3.1.3 Outputs

Again, there are a variety of outputs, depending on the unit. However, we will concentrate on the final output of the data “mini-workflow”, as this is where the data services interface to the other elements of a DataMiningGrid workflow.

It is currently envisioned that the data services portion of the workflow will end with the execution of a data services client. Such a client will be either:

1. An OGSA-DAI client; or

In both cases the corresponding Triana unit will output only a single parameter. This parameter will be a URI describing either a file, or if there is more, a directory containing a number of files. However, although each of these Triana units outputs only a single URI, there maybe more than one of these units. In this case, the output from the data services portion of the workflow will be a set of URIs.
4 Integration of the Application Discovery and Control Centre

The application discovery and control centre units, components, and services deal with all aspects that concern the discovery, selection, and control of applications on the grid. They consist of three individual units – ApplicationQuery, ApplicationSelector, and ControlCentre. Figure 3 shows these units in Triana.

Figure 3: The ApplicationQuery, ApplicationSelector, and ControlCentre units in Triana.

4.1 Application query unit

This unit is part of the application discovery process, which enables users to query MDS4 as central registry for available applications on the grid.

4.1.1 Inputs

This unit does not require inputs from any other unit. Before execution of the workflow the user is required to specify properties that they want to search for. Currently, this GUI allows users to specify the application’s unique ID, name or group. Furthermore, users can also specify the whole XPath query themselves, allowing them to search for all properties included in the application descriptions, e.g., version of the software, data mining technique used, or certain options provided.
4.1.2 Unit functionality

The ApplicationQuery unit uses the DataMiningGrid client-library to query the MDS4. During execution of the workflow, an XPath query is generated according to the inputs specified by the user. The query is then sent to the ApplicationQuery service, which in turn uses this query to receive the descriptions of the corresponding applications from the MDS4.

4.1.3 Outputs

This unit’s output consists of a list of applications returned by the ApplicationQuery service.

4.2 Application selector unit

This unit is also part of the application discovery process, which enables users to query the MDS4 at central registry for available applications on the grid.

4.2.1 Inputs

This unit takes at input the list of application descriptions found by the ApplicationQuery unit.

4.2.2 Unit functionality

The ApplicationSelector unit displays the most essential information contained in the descriptions passed to it as input at runtime in a table-like format. Users select a single application they find most appropriate for their work from this list.

4.2.3 Outputs

This unit’s output consists of the description of the selected application. The ControlCentre unit further processes this output.

4.3 Control Centre Unit

This unit acts as a simple graphical user interface for specifying the options of the selected application, its requirements, and data inputs, producing corresponding job descriptions.

4.3.1 Inputs

This unit has at least one input – the description of the application selected at the ApplicationSelector unit. However, in most cases the selected application will also require a certain number of data inputs, representing input data to work on, parameter files, and the like. Each data input is represented by a single URI, whereas such an URI may represent a file or a whole directory structure. This means that the number of inputs is not limited, but is specified implicitly by the selected application.
4.3.2 Unit functionality

With this unit the users specify all options, resource requirements (e.g., minimum amount of memory, disk space, etc.), and map all the different URIs representing the remote input files, required by the application for the application’s input slot. Such an input slot is mainly described in the application’s meta-information by a name, while starting the application, and a tool tip explaining to the user what the application demands as input for the respective slot. Users may also specify multiple values for the individual options that determine the application’s behaviour at runtime. Specifying multiple values for a single option results in repeated executions of the application on the grid, each with a different value for the respective option.

After a user has submitted the specifications, the unit calls client-side libraries to produce the corresponding job descriptions.

4.3.3 Outputs

This unit’s output consists of the job descriptions produced using the application’s meta-information and the values specified by the user.
5 Conclusions and Future Work

Deliverable D52 outlines in detail the initial integration phase of the project. In this document, we have specified the implementation details of the integration of all the main services into the workflow framework (namely Triana), and showed that such integration is feasible and viable. Triana has proven to be flexible enough to serve as a client to all services and components described above. Offering a wide range of error-proof and user-friendly capabilities to enable and ease the complicated task of the execution of complex data mining workflows in the grid.

This document has been discussed with the clients for all the main services as they are represented in Triana. One of the initial strategic decisions made by the Partners was to pursue a service-oriented architecture, which would wrap all main elements of the system as WSRF-compliant services. In this deliverable, we described (a) how clients for such WSRF-compliant services are wrapped as Triana units, (b) the communication between the units, and (c) the integration of all the stand-alone parts of the system into one common framework. This can be summarized as follows: The DataMiningGrid system is offering a wide range of solutions for data mining experts (system users). Its main advantage is to facilitate easy, scaleable deployment of batch-oriented data mining tasks utilizing the resources available in grid computing environments. In addition to the batch-job execution capabilities, the system offers users the ability to invoke WSRF-compliant Web services. This eventually results in a highly flexible system that, on one hand, facilitates elaborate data mining workflows for expert users and, on the other hand, simple workflows and simple Web-based user interfaces for end users who are not data mining experts.

Another important feature of the DataMiningGrid system is the integration of provenance\(^3\) information. Provenance information is indispensable for users who wish to use any results of previously performed data mining processes. Provenance information that contains all the data regarding previously executed jobs (e.g., log files, run time, used parameters values, execution machine, etc.). This data will be collected by the DataMiningGrid Resource Broker during the execution of jobs and stored in a dedicated XML database. This information can

\(^3\) Generally, provenance refers to the origin or source from which something comes. The term is often used in the sense of place and time of manufacture, production or discovery. Comparative techniques, expert opinion, written and verbal records and the results of tests are often used to help establish provenance. In significant distributed computing projects, such as are supported by a grid, or by Web services, many data and computational resources are produced and consumed. For example, a DataMiningGrid-supported project analyzing environmental data might involve equipment, various data sets, data mining tools, pre-processing steps and tools, environmental experts and data mining experts with different skill levels, etc. There may be many circumstances where there is a need to validate this information trail (or provenance) of the project: a regulatory authority might require an audit trail, it might be necessary to repeat the experiments, publishers might request provenance information to be submitted with a scientific paper, and so on.
later be retrieved and processed. Furthermore, this information may later be used for another data mining study. The DataMiningGrid Provenance Database will be also accessed by OGSA-DAI activities, which are an integrated part of the workflow framework. Hence, the DataMiningGrid system will provide users with simple but highly flexible tool for handling provenance information.
References

References cited in this document and throughout the project are listed in the DataMiningGrid Project Manual, which accompanies the deliverables and reports. Some of the references can also be searched on the project’s Web site’s digital library. The Project Manual is available to the EC Services and the reviewers on the project’s BSCW document server.

Abbreviations

Acronyms and abbreviations used in this document and throughout the project are listed and described more comprehensively in the DataMiningGrid Project Manual. The Project Manual is available to the EC Services and the reviewers on the project's BSCW document server.