

# Design of Distributed Data Mining Applications on the KNOWLEDGE GRID

Mario Cannataro  
ICAR-CNR  
Via P. Bucci 41c  
87036 Rende, Italy  
cannataro@acm.org

Domenico Talia  
DEIS  
University of Calabria  
Via P. Bucci 41c  
87036 Rende, Italy  
talia@deis.unical.it

Paolo Trunfio  
DEIS  
University of Calabria  
Via P. Bucci 41c  
87036 Rende, Italy  
trunfio@deis.unical.it

## Abstract

*Many industrial, scientific, and commercial applications need to analyze large data sets maintained over geographically distributed sites. The geographic distribution and the large amount of data involved often oblige designers to use distributed and parallel systems. The Grid can play a significant role in providing an effective computational support for distributed data mining and knowledge discovery applications. This paper introduces a software system for geographically distributed high-performance knowledge discovery applications called KNOWLEDGE GRID, describes the main system components, and discusses how to design and implement distributed data mining applications using these components.*

## 1. Introduction

In many scientific and business areas, massive data collections of terabyte and petabyte scale need to be analyzed. Moreover, in several cases data sets must be shared by large communities of users that pool their resources from different sites of a single organization or from a large number of institutions. Grid computing has been proposed as a novel computational model, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation. Today Grids can be used as effective infrastructures for distributed high-performance computing and data processing [1].

Together with the Grid shift towards industry and business applications, a parallel shift toward the implementation of data Grids has been registered. Data Grids are designed to allow for the storing of large data sets in repositories and also for moving them about with the same ease that small files can be moved. They represent an extension and enhancement of computational Grids driven by the need to

handle large data sets without constant, repeated authentication. Their main goal is to support the implementation of distributed data-intensive applications. Significant examples are the EU DataGrid, the Particle Physics Data Grid, the Japanese Grid DataFarm, and the Globus Data Grid [2] projects.

Data Grid middleware is crucial for the management of data on Grids, however in several scientific and business applications is also vital to have tools and environments that support the process of analysis, inference and discovery over the available data. The evolution of data Grids is represented by *knowledge grids* which offer high-level tools and techniques for the distributed mining and extraction of knowledge from data repositories available on the Grid. The development of such an infrastructure is the main goal of our work focused on the design and implementation of an environment for geographically distributed high-performance knowledge discovery applications called KNOWLEDGE GRID. The KNOWLEDGE GRID can be used to perform data mining on very large data sets available over grids to make scientific discoveries, improve industrial processes and organization models, and uncover business valuable information.

The outline of the paper is as follows. Section 2 describes the KNOWLEDGE GRID general architecture and the features of the main components. Section 3 discusses the basic steps to design, build, and execute a distributed data mining application using the tools offered by the KNOWLEDGE GRID. Section 4 concludes the paper.

## 2. The Knowledge Grid Architecture

The KNOWLEDGE GRID architecture [3] is defined on top of Grid toolkits and services, i.e. it uses basic Grid services to build specific knowledge extraction services. Following the Integrated Grid Architecture approach, these services can be developed in different ways using the available Grid toolkits and services. The current implementa-

tion is based on the Globus toolkit [4]. Like Globus, the KNOWLEDGE GRID offers global services based on the cooperation and combination of local services. We designed the KNOWLEDGE GRID architecture so that more specialized data mining tools are compatible with lower-level Grid mechanisms and also with the Data Grid services. This approach benefits from "standard" Grid services which are more and more utilized and offers an open parallel and distributed knowledge discovery architecture that can be configured on top of Grid middleware in a simple way.

## 2.1. Knowledge Grid services

The KNOWLEDGE GRID services are organized in two hierarchic levels:

- the Core K-Grid layer and
- the High level K-Grid layer.

The former refers to services directly implemented on the top of generic Grid services, the latter is used to describe, develop and execute distributed knowledge discovery computations over the KNOWLEDGE GRID. The KNOWLEDGE GRID layers are depicted in figure 1. The figure shows layers as implemented on the top of Globus services. The KNOWLEDGE GRID data and metadata repositories are also shown. In the following the term *K-Grid node* will denote a Globus node implementing the KNOWLEDGE GRID services.

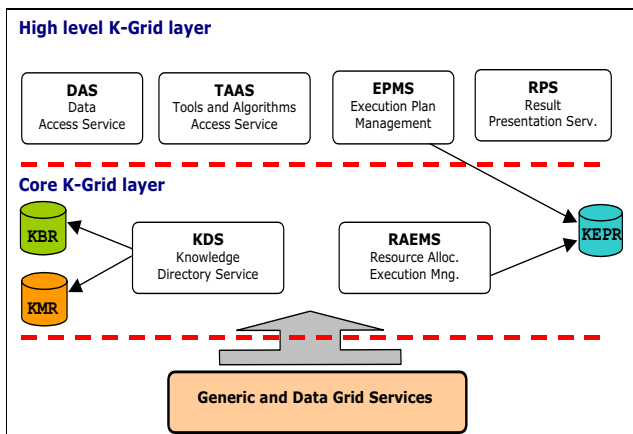


Figure 1. The Knowledge Grid architecture

**Core K-Grid Layer.** This layer offers the basic services for the definition, composition and execution of a distributed knowledge discovery computation over the Grid. Its main goals are the management of all metadata describing features of data sources, third party data mining tools,

data management, and data visualization tools and algorithms. Moreover, this layer coordinates the application execution by attempting to fulfill the application requirements and the available Grid resources. This layer comprises two main services:

- Knowledge Directory Service (KDS)

This service extends the basic Globus MDS service and it is responsible for maintaining the metadata describing all the data and tools used in the KNOWLEDGE GRID. They comprise:

- Repositories of data to be mined (data sources);
- Tools and algorithms used to: i) extract, filter and manipulate data (data management tools); ii) analyze (mine) data; iii) visualize, store and manipulate mining results;
- Distributed knowledge discovery execution plans. An execution plan is an abstract description of a KNOWLEDGE GRID application, that is a graph describing the interaction and data flow between data sources, data mining (DM) tools, visualization tools, and result storing.
- Knowledge obtained as result of the mining process, i.e. learned models and discovered patterns.

The metadata information are represented by XML (eXtensible Markup Language) documents and are stored in a Knowledge Metadata Repository (KMR). Whereas it could be infeasible to maintain the data to be mined in an ad hoc repository, it could be useful to maintain a repository of the discovered knowledge. This information (see below) is stored in a Knowledge Base Repository (KBR); metadata describing them are managed by the KDS.

The KDS is then used not only to search and access raw data, but also to find previously discovered knowledge that can be used to compare the output of a given mining computation when data change, or to apply data mining tools in an incremental way. Data management, analysis and visualization tools are usually pre-existent to the KNOWLEDGE GRID (i.e. they reside over file systems or code libraries). Another important repository is the Knowledge Execution Plan Repository (KEPR) that holds the execution plans of data mining processes.

- Resource Allocation and Execution Management Service (RAEMS).

This service is used to find the best mapping between an execution plan and available resources, with the goal of satisfying the application requirements (computing power, storage, memory, database, network

bandwidth and latency) and Grid constraints. The mapping has to be effectively obtained (co-)allocating resources. After the execution plan activation, this layer manages and coordinates the application execution. Other than using the KDS and the Globus MDS services, this layer is directly based on the Globus GRAM services. Resource requests of each single data mining program are expressed using the Resource Specification Language (RSL). The analysis and processing of the execution plan will generate global resource requests that in turn are translated into local RSL requests for local GRAMs.

**High level K-Grid Layer.** The High level K-Grid layer includes services used to compose, validate, and execute a parallel and distributed knowledge discovery computation. Moreover, the layer offers services to store and analyze the discovered knowledge. Main services are:

- Data Access Service (DAS)

This service is responsible for the search, selection (Data search services), extraction, transformation and delivery (Data extraction services) of data to be mined. The search and selection services are based on the core KDS service. On the basis of the user requirements and constraints, the Data Access service automates (or assists the user in) the searching and finding of data sources to be analyzed by the DM tools. The extraction, transformation and delivery of data to be mined are based on the Globus GASS services and it makes use of KDS.

- Tools and Algorithms Access Service (TAAS)

This service is responsible for the search, selection, downloading of data mining tools and algorithms. As before, the metadata regarding their availability, location, and configuration are stored into the KMR and managed by the KDS, whereas the tools and algorithms are stored into the local storage facility of each K-Grid node. A node wishing to "export" data mining tools to other users has to "publish" them using the KDS services, which store the metadata in the local KMR.

- Execution Plan Management Service (EPMS)

An execution plan is represented by a graph describing the interaction and data flows between data sources, extraction tools, DM tools, visualization tools, and storing of knowledge results in the KBR. In simplest cases, a user can directly describe the execution plan, using a visual composition tool where the programs are connected to the data sources. However, due to the variety of results produced by the DAS and TAAS,

different execution plans can be produced, in terms of data and tools locations, strategies to move or stage intermediate results and so on. Thus, the Execution Plan Management Service is a semi-automatic tool that takes data and programs selected by the user, and generates a set of different, possible execution plans that meet user, data and algorithms requirements and constraints. Execution plans are stored in the Knowledge Execution Plan Repository (KEPR).

- Results Presentation Service (RPS)

Result visualization is a significant step in the data mining process that can help users in the model interpretation. This service specifies how to generate, present and visualize the knowledge models extracted (e.g., association rules, clustering models, classifications), and offers the API to store them in different formats in the Knowledge Base Repository. The result metadata are stored into the KMR to be managed by the KDS.

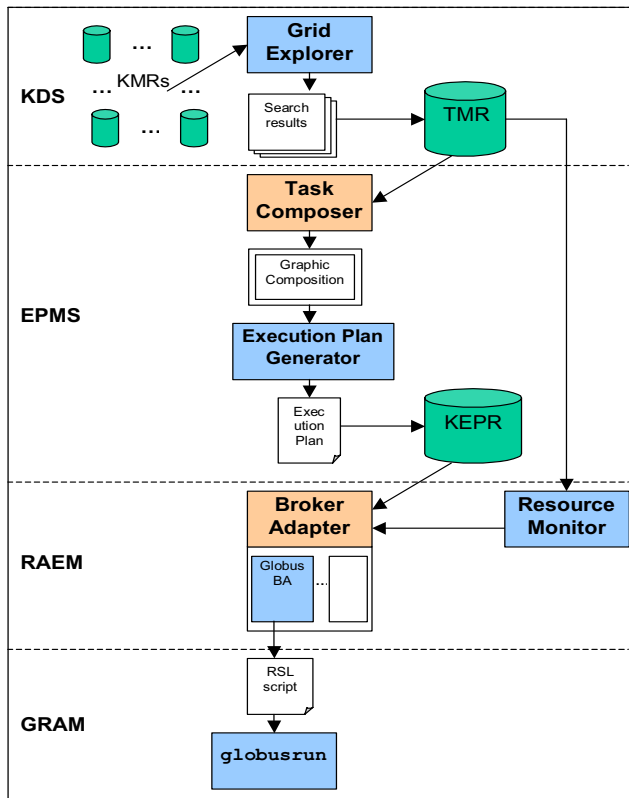
### 3. Design and Execution of Distributed Data Mining Applications

The design and execution of computations on the KNOWLEDGE GRID is performed according to the scheme showed in figure 2. The *Grid Explorer (GE)* is used to discover resources of interest over the KNOWLEDGE GRID. The search is accomplished in two steps:

1. the metadata repositories are first located, searching LDAP directories for *K-Grid* entries, then
2. the relevant XML metadata are downloaded from the specified KMR (Knowledge Metadata Repository) and stored into the local file system.

The collected XML metadata are analyzed to find more specific information by using different search parameters and selection filters. The useful metadata (i.e., those satisfying the searching and filtering criteria) are then stored into the *Task Metadata Repository (TMR)*, a local space which contains information about resources (nodes, data sources and algorithms) selected to perform a computation. The TMR is organized as a set of directories: each one is named with the fully qualified hostname of a Grid node, and contains metadata files about resources of that node.

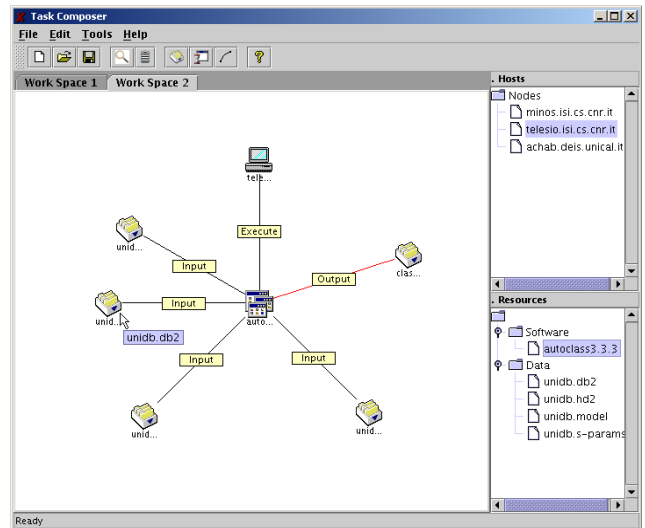
The *Task Composer (TC)* (see figure 3) assists a user in the design of a data mining computation. It allows a user to browse the TMR directory and select the hosts to be involved in the computation. Selected hosts are displayed into the *Hosts* panel, and a user can explore resources of each one by clicking on its label. Those resources are displayed by categories into the *Resources* panel. The



**Figure 2. Design and execution steps of a data mining application**

user designs the overall computation as composed of one or more *Workspaces*. The user can drag the objects (hosts, software, data, etc.) presented in the *Hosts* and *Resources* panels into the active *Workspace*, then she/he can link those objects in order to indicate the interaction between them. Links can represent different actions, such as data transfer, programs execution, and input and output relations. The *data transfer* link is used to move resources among different locations of the Grid. The *execute link* is used to run an application on a Grid host, the input and output links are used to respectively indicate input and output elements of a program. For each link type is possible to set related parameters (e.g., protocol and destination path of the data transfer, job-manager of the execution, etc.).

The TC verifies the execution plan consistency by allowing users to create links only if they represent actions that can be really executed. The computation can be designed by using a set of different *Workspaces*. *Workspaces* are executed sequentially whereas the tasks specified in a given *Workspace* are (when possible) executed in parallel. Therefore, the resources generated by a certain *Workspace* are made available to be used in the next *Workspaces*. For



**Figure 3. Task Composer interface**

instance, if in the *Workspace 1* a file  $F$  is transferred to a node  $N$ , a new metadata document is created for  $F$  and stored in the directory  $N$  of the TMR. The document specifying the new location of the  $F$  file is marked as temporary until the data transfer is performed. However, if the *Workspace 1* is opened in the same session, the  $F$  file is displayed as already available under the resources of  $N$ .

Starting from the graphic description of the computation produced by the TC, the *Execution Plan Generator (EPG)* will produce the corresponding XML document representing the execution plan of the computation. The EPG can be invoked by the user through a dedicated section of the TC, after the design of the application is done. When invoked, the EPG accomplishes its goal by parsing the tasks specified in the *Workspace* sequence and taking in account the properties of the involved resources and the parameters of the links. The execution plan describes the data mining computation at a high level, neither containing physical information about resources (which are identified by metadata references), nor about status and current availability of such resources. In fact, specific information about the involved resources will be included in the next phase, when the execution plan is translated into a particular broker language. Figure 4 shows a portion of an execution plan.

The execution plan gives a list of tasks and task links, which are specified using respectively the XML tags `Task` and `TaskLink`. The `label` attribute for `Task` element identifies each basic task in the execution plan, and it is used in linking various basic tasks to form the overall task flow. Each `Task` element contains a task-specific sub-element, which indicates the parameters of the particular represented task. For instance, the task identified by the `ws1_dt4` label contains a `DataTransfer` element, indicating that

```

<ExecutionPlan>
...
<Task ep:label="ws1_dt4">
  <DataTransfer>
    <Source ep:href="minos../unidb_db2.xml" ep:title="unidb.db2 on
minos.isi.cs.cnr.it"/>
    <Destination ep:href="telesio../unidb_db2.xml" ep:title="unidb.db2 on
telesio.isi.cs.cnr.it"/>
    ...
  </DataTransfer>
</Task>
...
<Task ep:label="ws2_c1">
  <Computation>
    <Program ep:href="telesio../autoclass3-3.xml" ep:title="autoclass on
telesio.isi.cs.cnr.it"/>
    <Input ep:href="telesio../unidb_db2.xml" ep:title="unidb.db2 on
telesio.isi.cs.cnr.it"/>
    ...
    <Output ep:href="telesio../classes.xml" ep:title="Classes on
telesio.isi.cs.cnr.it"/>
  </Computation>
</Task>
...
<TaskLink ep:from="ws1_dt4" ep:to="ws2_c1"/>
...
</ExecutionPlan>

```

**Figure 4. Extract of a sample execution plan.**

it is a data transfer task. The `DataTransfer` element specifies `Source` and `Destination` of the data transfer. The `href` attributes of such elements specify the location of metadata about source and destination objects. In this example, metadata about source of data transfer in the `ws1_dt4` task are provided by the `unidb_db2.xml` file stored in the directory named `minos.isi.cs.cnr.it` of the TMR, whereas metadata about destination are provided by the `unidb_db2.xml` file stored in the directory named `telesio.isi.cs.cnr.it` of the same TMR. The first of such XML documents provides metadata about the `unidb.db2` data set stored on `minos.isi.cs.cnr.it`, whereas the second one provides metadata about the `unidb.db2` data set when, after the data transfer, is stored on `telesio.isi.cs.cnr.it`. The `TaskLink` elements represent relations among tasks of the execution plan. For instance, the showed `TaskLink` indicates that the task flow proceeds from the task `ws1_dt4` to the task `ws2_c1`, as specified by its `from` and `to` attributes.

The *Broker Adapter (BA)* maps a generic execution plan, represented by an XML document, into a specific broker script, which can be directly executed by that broker. We implemented the Globus-BA (G-BA), a module that generates Globus RSL scripts. The TC also integrates a user interface to the G-BA module, allowing to generate the RSL script from the current design session, and submit it to `globusrun` program for its execution.

The mapping process performed by the BA is based on the *Resource Monitor (RM)* tool that provides physical information of resources referenced in the execution plan. Static information (e.g., CPU speed, memory size, etc.) is provided by the RM directly accessing metadata stored in the TMR, while dynamic information (e.g., current CPU

load and network latency) is provided by means of specific testing tools, for instance based on standard Unix or TCP/IP utilities.

All the design and development functionalities of the KNOWLEDGE GRID discussed in this section have been implemented in a toolset named *VEGA* (a Visual Environment for Grid Applications) that allows a user to build a data mining computation starting from a set of useful remote resources such as computational nodes, sources of data, and data mining algorithms. VEGA offers the functionalities of the Grid Explorer, the Task Composer, the Execution Plan Generator, and the Resource Monitor through a graphical interface (see figure 3) which a user can use to compose and execute a knowledge discovery computation in a simple way.

## 4. Conclusion

The Grid infrastructure is growing up very quickly and is going to be more and more complete and complex both in the number of tools and in the variety of supported applications. Along this direction the Grid services are shifting from generic computation-oriented services to high-level information management and knowledge discovery services. The KNOWLEDGE GRID system we discussed here is a significant component of this trend. It integrates and completes the data Grid services by supporting distributed data analysis and knowledge discovery and management services that will enlarge the application scenario and the community of Grid computing users.

**Acknowledgements.** This work has been partially funded by the project "MIUR Fondo Speciale SP3: GRID COMPUTING: Tecnologie abilitanti e applicazioni per eScience".

## References

- [1] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Intl. J. Supercomputer Applications*, 15(3), 2001.
- [2] A. Chervenak, I. Foster, C. Kesselman, C. Salisburry, S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. *J. of Network and Computer Applications*, 23:187–200, 2001.
- [3] M. Cannataro, D. Talia, P. Trunfio. KNOWLEDGE GRID: High Performance Knowledge Discovery Services on the Grid. In *Proc. GRID 2001*, LNCS 2242, pp. 38-50, 2001.
- [4] I. Foster, C. Kesselman. Globus: a Metacomputing Infrastructure Toolkit. *Intl. J. Supercomputer Applications*, 11:115–128, 1997.