# GridMiner: An Infrastructure for Data Mining on Computational Grids

Peter Brezany[1], Jürgen Hofer[1], A Min Tjoa[2], Alexander Wöhrer[1]

[1] Institute for Software Science, University of Vienna
Liechtensteinstrasse 22, A-1090 Vienna, Austria
E-mail: {brezany,hofer,woehrer}@par.univie.ac.at
[2] Institute for Software Technology, Vienna University of Techology
Favoritenstrasse 9-11/E188, A-1040 Vienna, Austria
E-mail: tjoa@ifs.tuwien.ac.at

## *Abstract*

Knowledge discovery in datasets integrated into Grids is a challenging research task. These large datasets are being collected and accumulated across a wide variety of fields, at a dramatical pace. They are often heterogeneous and geographically distributed and globally used by large user communities. There are major challenges involved in the efficient and reliable storage, fast processing, integration and extracting descriptive and predictive knowledge from this great mass of data. In this paper, we describe design principles and a service based software architecture of a novel infrastructure called the GridMiner for distributed data mining and data integration in Grid environments. This architecture is being implemented on top of the Globus 3.0 toolkit, using OGSA-DAI for access to Grid Data Sources.

**Keywords:** data mining, Grid Computing, Grid services, workflow management, distributed data integration, Grid databases, data preprocessing, meta-learning

# INTRODUCTION

Advanced analysis of data managed within Computational Grids (see Berman et.al., 2003) is a challenging problem. A (nondefinitive) list of the kinds of the most typical data that are dealt with in modern Grid applications includes (see Kunszt, 2003) files, file collections, relational databases, XML databases and semistructured data, virtual data (large sets of secondary data that are derived from primary data) and data objects. Depending on the application area, the data mentioned above may include data produced by business transactions, medical investigations, scientific simulations, along with data obtained from satellites, telescopes, microscopes, seismic or tomographic techniques, etc., and its volume is already measured in terabytes and will soon total petabytes. The cost of producing this data is sometimes very high. Without effective ways to retrieve, analyze and manipulate it, this great expense will not yield the benefits to society that we might expect. The data is often geographically distributed and its complexity is increasing, meaning that the extraction of meaningful knowledge requires more and more computing resources. The communities of users that need to access and analyze this data are often large and geographically distributed. This combination of large data volume, geographic distribution of users and resources, and computationally intensive analysis[1] results in many complex and stringent demands that, till the advent of Grid systems, were not satisfied by any existing computational and data management infrastructure.

The fundamental challenge of the current Grid research efforts is to make Grid systems widely available and easy to use for a wide range of applications. This is crucial for accelerating their transition into fully operational environments. A significant amount of research for achieving these objectives has already started both in academia as well as industry. In the first phase, this research and development has been almost exclusively driven by ``big science'', like distributed high-performance simulations, high-energy physics, material science, etc. Now the focus is going to shift to more general domains, closer to everyday life, like medical, business, and engineering applications[2] (see GRIDS: e-Science to e-Business, 2001).

However, to our best knowledge, till today, no relevant effort has been devoted to the development of Grid tools and services allowing to perform efficient knowledge discovery in large distributed databases and other datasets associated with these applications. Therefore, we have initiated a research effort to design and experimentally implement a novel infrastructure called the *GridMiner* for knowledge discovery in datasets integrated into Computational Grids.

The technology developed is being validated and tested on an advanced medical application addressing treatment of traumatic brain injury (TBI) victims (see Mauritz, 2003). TBIs typically result from accidents in which the head strikes an object. The trajectory of the TBI patient management includes the following main points: trauma event (e.g. injury at a car or sport accident), first aid, transportation to hospital, acute hospital care, and home care. All these phases are associated with collecting data into databases, which are currently autonomously managed by individual hospitals, and are, therefore, geographically distributed. Moreover, they are heterogeneous and need high data security precautions. The

[1] Moreover, certain level of QoS and security has to be guaranteed for several classes of applications.
[2] see National eScience Centre, http://umbriel.dcs.gla.ac.uk/NeSC/general

aim is to use the Grid technology for building a virtual organization, in which the cooperation of the participating hospitals and institutions is supported by the integration of the above databases. Knowledge discovered in these databases can help to significantly improve the quality of the decisions taken by the health care specialists involved in treatment of the TBI patients. In most situations, a nearly real-time response to knowledge discovery queries is urgently needed. It is obvious that this kind of applications introduces new challenges to the Grid developers.

The paper is organized as follows. The subsequent section discusses related work and outlines the principles that were followed in developing the current Grid technology, which we use as a basis for the GridMiner design. The 'GridMiner Requirements' section discusses the requirements of parallel and distributed data mining on the GridMiner. The kernel part (Sections 'Data Access and Data Mediation Services' and 'The Data Mining Layer') deals with the design of the service-oriented architecture of the GridMiner, which is the main contribution of the paper. The architecture and concepts of the services developed for Grid databases access and data mediation are described in section 'Data Access and Data Mediation Services'. The availability of such services is crucial for the design of concepts for data mining services which are presented in section 'The Data Mining Layer'. Finally we briefly conclude and outline the future work in the last section.

# BACKGROUND AND RELATED WORK

The early Grid efforts (the early to mid 1990s) started as projects to link supercomputing sites; at this time this approach was known as metacomputing. The objective was to provide computational resources to a range of high performance applications. Later this focus has shifted to more data-intensive applications (see Chervenak, 2001), where significant processing is done on very large amounts of data.

Essentially all major Grid projects are currently built on protocols and services provided by the Globus Toolkit[3] that enables applications to handle distributed heterogeneous computing resources as a single virtual machine. It provides the interoperability that is essential to achieve large-scale computation.

New-generation Grid technologies are evolving toward an Open Grid Services Architecture, OGSA (see Foster, 2002), in which a Grid provides an extensible set of services that virtual organizations can aggregate in various ways. Building on concepts and technologies from both the Grid and Web services[4] communities, OGSA defines a uniform exposed service semantics (the Grid service); defines standard mechanisms for creating, naming, and discovering transient service instances; provides location transparency and multiple protocol bindings for service instances; and supports integration with underlying native platform facilities.

OGSA also defines, in terms of Web Services Description Language (WSDL)[5] interfaces and associated conventions, mechanisms required for creating and composing sophisticated

[3] http://www.globus.org
[4] The creation of Web services standards is an industry-led initiative, with some of the emerging standards in various states of progress through the World Wide Web Consortium (W3C).
[5] http://www.w3.org/TR/wsdl12/

distributed systems, including lifetime management, change management, and notification. Service bindings can support reliable invocation, authentication, authorization, and delegation.

The development of OGSA technical specification is ongoing within the Global Grid Forum[6] within the tasks called the *Open Grid Services Infrastructure* (OGSI). The Globus Project is developing the Globus Toolkit 3.0 (GT3), which is based on OGSI mechanisms; the first experimental implementation, GT3 Alpha Grid Services, is already available.

As was already mentioned, Grid computing began with an emphasis on compute-intensive tasks, which benefit from massive parallelism for their computation needs, but are not data intensive; the data that they operate on does not scale in portion to the computation they perform. In recent years, this focus has shifted to more data-intensive applications, where significant processing is done on very large amounts of data. Therefore, within the context of OGSA activities, the Global Grid Forum Database Access and Integration Services (DAIS) Group developed a specification for a collection of OGSI-Compliant Grid database services. The first implementation of the service interfaces, OGSA-DAIS Release 2, is already available.

So far, only a little attention was devoted to knowledge discovery on the Grid. There are already many publication on parallel and distributed data mining (e.g. Grossman, 2001, Kargupta, 2000, Zaki, 2000). An attempt to design an architecture for performing data mining on the Grid was presented in Cannataro (2001) and Cannatoro (2003). The authors, M. Cannataro and D. Talia, present design of a Knowledge Grid architecture based on the non-OGSA-based version of the Globus Toolkit, and don't consider any concrete application domain. R. Moore presents the concepts of knowledge-based Grids in Moore (2001). Mahinthakumar et al. (1999) report about the first clustering algorithm implementation on the Grid. A lot of valuable data integration concepts have been developed in the project "Federated Database for Neuroscience" (see Baru, 1999, Ludaescher, 2001). The WP4[7] of the OGSA-DAI project is working on the design of a distributed query processing service for the Grid.

To our best knowledge, no research effort about development of OGSA-based data mining services has been reported. Our solutions discussed in forthcoming sections can be viewed as a result of the natural evolution of parallel and distributed data mining technology and Grid Services and Grid Database Services development, and represent the first steps towards development standards for Open Service Architecture for data mining in Grid environments.

[6] http://www.gridforum.org
[7] OGSA-DAI Workpackage 4, http://www.neresc.ac.uk/projects/ogsa-dai-dqp/dqp.html

# GRIDMINER REQUIREMENTS

The basic principles that motivate the architecture design of a service-oriented grid-aware data mining system, such as the GridMiner are listed below.

- *Open architecture.*
  In the context of the OGSA documents (see Foster, 2002 and Global Grid Forum, 2002), open is being used to communicate extensibility, vendor neutrality, and commitment to a community standardization process.

- *Data distribution, complexity, heterogeneity, and large data size.*
  GridMiner must be able to cope with very large and high dimensional data sets that are geographically distributed and stored in different types of repositories.

- *Applying different kinds of analysis strategies.*
  Because this is the first effort to service-oriented Grid-enabled data mining, a wide spectrum of data mining strategies has to be considered and accommodated in the system, and their impact on performance has to be investigated.

- *Compatibility with existing Grid infrastructure.*
  The higher levels of the GridMiner architecture use the basic Grid services for implementing wide area communication, cooperation and resource management.

- *Openness to tools and algorithms.*
  The GridMiner architecture must be open to the integration of new data mining tools and algorithms.

- *Scalability.*
  Architecture scalability is needed both in terms of number of nodes used for performing the distributed knowledge discovery tasks and in terms of performance achieved by using parallel computers to speed up the data mining task.

- *Grid, network, and location transparency.*
  Users should be able to run their data mining applications on the Grid in an easy and transparent way, without needing to know details of the Grid structure and operation, network features and physical location of data sources. This is supported by a layer of Grid data virtualization services.

- *Security and data privacy.*
  Security and privacy issues are vital features in a lot of data mining applications (see Cannataro, 2003). The Grid services offer a valid support to the GridMiner system to cope with user authentication, security and privacy of data. However, knowledge discovery specific security services, such as filtering of sensitive data, are not provided by Grid middleware, thus they are implemented as Grid data mining services.

- *OLAP Support.*
  The architecture must allow the interoperability with data warehouse and Online Analytical Processing (OLAP) services. Data mining and OLAP are two complementary

methodologies, hich, if applied in conjuction (see Han, 2000 and Tam, 1998), can provide powerful advanced data analysis solutions.

# DATA ACCESS AND DATA MEDIATION SERVICES

One of the most important GridMiner architecture design issues is the design of an appropriate data access and integration model and, consequently, specification of the services implementing this model. In this section, we deal with *Data Access services*, which implement the data access to databases and other Grid data repositories, and *Data Mediation Services*, which provide an integrated view of distributed data to the data mining services, as illustrated in Figure 1.

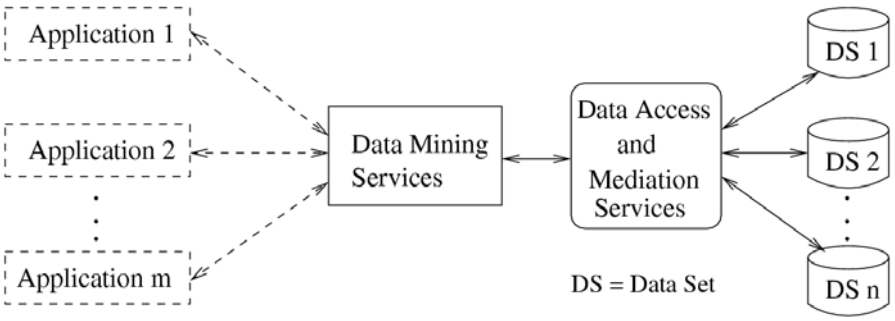


**Figure 1: Data Access, Mediation, and Data Mining Services**

## *Data Distribution Scenarios*

We are structuring our discussion along with different possibilities of data distribution.

1. *Single data source.*
   This case is quite simple, because the data is concentrated in one physical data source (e.g. a raw file, an XML or relational database). For grid-enabled access to the data, a uniform Service Interface must be implemented that provides location, platform and Grid transparency and abstracts from underlying technical details. The used technology should be open in architecture and design to support an extensible set of protocols and data sources. After binding the service to a data source it provides meta-data and mechanisms for querying data. These meta-data include a description of the physical data source structure, supported query languages, states, current workload and others. We call a component that realizes the functionality mentioned above as the Wrapper.

2. *Federated data sources, horizontal partitioning.*
   In part A of Figure 2, there are *n* sites that store parts of data of equal structure and semantics for a given problem domain. So the service has to deal with transactions that are physically distributed but stored within consistent schemes (row-wise partitioning). Let $D = \{D_1, D_2, \ldots D_n\}$ be a set of data sources having the same schema $R_{i \in (1 \ldots n)} = (a_1, a_2, \ldots a_m)$. The Service now has to be bound to *D* but provides

a global, single data source description ($R_i$). The physical partitioning of the data is transparent to higher level services which means that the service must query all the *n* data sources and combine the results to one complete result dataset. We call a component that is able to fulfill the requirements mentioned above as the Mediator.

3. *Federated data sources, vertical partitioning.*
   Our Mediator is now faced to relations that have a common attribute *id* that unambiguously identifies each transaction, but other attributes are distributed over multiple sites as pictured in part B of Figure 2. For example, there can be a table with personal data of developers $R_{pd}$ = ( *id, eduction, salary*) at one site and a second one about their favorite operation system and programming language $R_{fd}$ = ( *id, os, plang* ) at another site, and we want to predict the income class of a certain new undergraduate Linux C++ programmer. Formally written, for *n* database sites, there are *n* schemes containing *m* attributes $R_1 = (id, a_{1,1}, \ldots, a_{1,m})$, … $R_n = (id, a_{n,1}, \ldots, a_{n,m})$, and the task of the mediation system is to provide a virtual relation that consists of a subset of the superset $R^* = R_1 \times \ldots \times R_n$. The service is again bound to a set of data sources and provides a global, single data source description which is the superset $R^*$.

4. *Federated data sources, heterogenous datasources schemes.*
   The hardest task is to mediate between data sources that are heterogeneous in structure and design.This class of problems arises when attributes with the same semantics differ in name, description, application or data type, e.g. let $D_1$ be an relational database and $D_2$,$D_3$ be XML databases for our example described in part C of Figure 2. At the moment, we do not follow the approach to describe the data sources semantically and then translate over a meta-mechanisms as done in semantic Web/Grid approaches (see Levy 1996).Our goal is to provide a physical-to-logical schema mapping mechanism that allows the translation of logical resource request to their physical correspondents and back translation of the results.Formally, now the Mediator has to deal with situations like ($R_1 = (a_1, a_2)$, $R_2 = (a_1, a_3)$), and $a_2 \equiv a_3$.

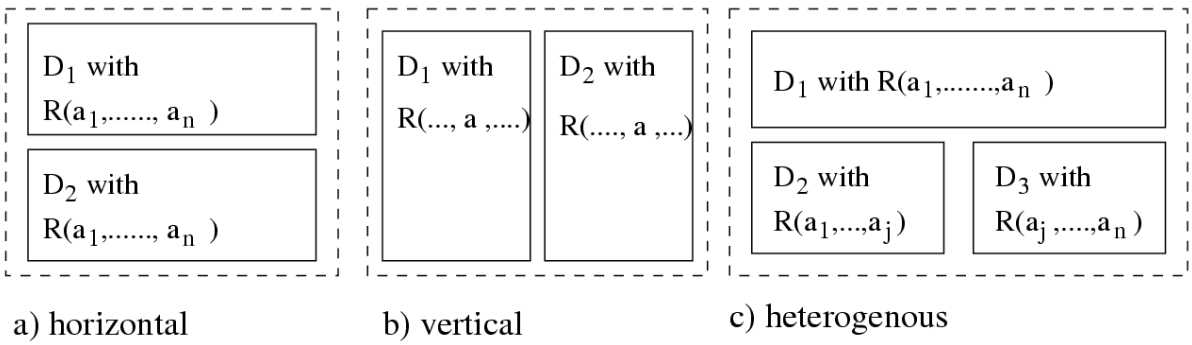


**Figure 2: Federated data sources with different types of partitioning**

## *GridMiner Mediator Configuration*

We designed the Mediator as a middleware service, which connects to the participating data sources, integrates them logically into *virtual data source* (VDS)[8], sends queries to them, and combines and deliver the results in a flexible way.

In the first prototype implementation, we use an XML-Script for describing the mediation service built-up process and characteristics of the VDS. The mapping schema given in Figure 3 holds the information to resolve a distribution such as the one given in part C of Figure 2. The table *virtual_one* needs one *join* operation (for the given vertical partitioning) and one *union* (for the horizontal partitioning) work step to be materialized correctly. As one can easily see, the provided attributes for this table in the VDS are p_id, p_name, p_dob, p_bt, p_fc.

For very complex or often used mediation tasks, this information can be stored in the service called Grid Data Service Factory (GDSF), which generates the mediation service. For highly dynamic federations, the configuration metadata can also be passed to the GDSF at runtime.

```
<MappingSchema>
 <Table name="virtual_one">
  <union>
   <join>
    <source key="pid" id="D2">
     <map dest="p_id" src="pid"/>
     <map dest="p_name" src="name"/>
     <map dest="p_dob" src="dob"/>
    </source>
    <source key="pid" id="D3">
     <map dest="p_bt" src="bloodtype"/>
     <map dest="p_fc" src="first_contact"/>
     <map dest="p_id" src="pid"/>
    </source>
   </join>
   <source key="pid" id="D1">
    <map dest="p_id" src="pid"/>
    <map dest="p_name" src="name"/>
    <map dest="p_dob" src="birthday"/>
    <map dest="p_bt" src="bt"/>
    <map dest="p_fc" src="fc"/>
   </source>
  </union>
 </Table>
 ...
</MappingSchema>
```

**Figure 3: Mapping schema for the distribution in part C of Figure 2**

[8] a data federation that presents multiple collections of data as one uniformly organized data collection

### *Implementation of the Mediation Service*

In order to avoid building a new proprietary solution and reimplementing well solved aspects of Grid Data services, we have decided to extend the free available OGSA-DAI Grid Data Service (GDS) reference implementation to provide a virtual data source (VDS).

Offering the same metadata as a normal Grid data source, it can be used and integrated in existing applications quite easily to hide the distribution and heterogeneity of the participating data sources by reformulating requests against the logical schema of the VDS.

The mediator builds the results of a query as follows. First, the *parser* checks the query, e.g. if the used attributes are availiable and if it is a valid statement[9]. Afterwards, the *query rewriter* identifies the relevant data sources with the help of the *mapping schema* given in Figure 3 and the extended *logical schemas* (see Hong et.al., 2003) stored (for each data source) in the GDSF. It develops the query execution plan for obtaining the requested data.

For our previous example this would lead to one reformulated (according to the information provided in the *select*-element of the mapping schema in Figure 3) SELECT-statement executed on the relational database and two different XQUERY/XPATH-statements performed on the XML databases. The results of the XML databases are joined via the information provided in the *join*-element of the mapping schema given in Figure 3. The last step of the mediation task is to *merge* the two result-sets together as specified by the *union*-element in the mapping schema. The delivery for a small amount of data can be done via direct XML response, for a huge amount of data more sophisticated mechanisms like GridFTP (see Allcock et.al., 2002) will be supported.

## THE DATA MINING LAYER

The following subsection ‘Components’ introduces the components that are part of the data mining layer. Afterwards the system architecture is illustrated by three different usage scenarios that emphasize different aspects of the GridMiner: A centralized data mining tasks that is performed on one node (a "grid-enabled" classical data mining application) in the first one. Then a distributed version that uses parallel data mining algorithms (e.g. SPRINT, see Shafer, 1996, ScalParC, see Joshi, 1998 for classification) and finally a more complex workflow, derived by a typical knowledge discovery process, that is orchestrated by a workflow engine. A prototype implementation (see Section ‘Implementation Prototype’) covering the first scenario is already available and is being evaluated.

[9] We are supporting a subset of SQL92 as query language for its easy usability.

## Components

The data mining layer consists of the following components:

- *GridMiner Service Factory (GMSF).*
  The GMSF is a specialized, persistant service for GridMiner-related services that is responsible for creating instances of transient GridMiner services. On each host, where a service should be created, a GMSF instance must be available. The component that wants to create a new service instance delegates the creation to the factory, which creates the service instance and returns a Grid Service Handle (GSH) to the new instance.

- *GridMiner Service Registry (GMSR).*
  The GMSR is a specialized, persistant service that is derived from the standard OGSA registry service. Although it is possible to have multiple registries only one central registry for all GridMiner-related services is assumed within this document. The registry is created at startup time of the OGSA container. GridMiner Service Factories that are created at container startup register themself automatically within the registry. A client application can then browse or search the registry for interesting services. The Registry returns GSHs as an answer to queries.

- *GridMiner DataMining Service (GMDMS).*
  The GridMiner DataMining Service is a central, transient GridService that provides an extensible set of data mining and advanced data analysis algorithms and features. It is created by the GMSF and instructed like the other services in terms of a specification expressed in the GM-JSL notation (see Section ‘GridMiner Job Specification Language’). The service can simply encapsulate an available data mining toolkit, what we call the centralized scenario, or it may create and control multiple, distributed service instances on other nodes that execute together a parallel or distributed data mining algorithm. These 'worker instances' usually have a high degree on communication, hence should use an efficient communication protocol.

- *GridMiner PreProcessing Service (GMPPS).*
  The GridMiner Preprocessing Service encapsulates functionality that is applied before the main data mining step. Such preprocessing activities include data cleaning, integration, handling missing data and statistical noise, aggregations, subset selection and many more. Obviously the amount of possible activities is quite big and the activities are very different - so this service has to be easily extensible and modular in design. The Service is able to operate on top of multiple different data sources, most important file systems, relational databases and Grid-Databases (e.g. OGSA-DAI) and of course the Mediator described above.

- *GridMiner Presentation Service (GMPRS).*
  The Presentation Service is a transient, thin component that receives a machine-readable form of a data mining model as input and translates, renders or processes the model into different output formats. Such formats may include user-friendly representations of decision-trees, charts, association rules or interactive systems a user can use afterwards.

– *GridMiner Orchestration Service (GMOrchS).*
The Orchestration Service is an optional component that aggregates a sequence of data mining related activities to a job. The user can formulate his tasks as a business process in an appropriate, formal language and the GMOrchS acts a workflow engine that executes the steps sequentially or in parallel. Remember that each service can also be contacted directly by the user application. The GMOrchS is a transient service created by a GMSF and cannot be shared among multiple users. It is destroyed after finishing the job execution. It's main purpose is to ease the handling of complex, long-running jobs with many components involved.

## *Centralized Data Mining*

Figure 4 illustrates the first scenario. In this case the user wants to perform some data preprocessing first and analyze the resulting dataset via data mining algorithms in a centralized fashion (on one host) afterwards. A Grid data source is used by both service instances that virtualizes the physical location and layout of the dataset. The control flow is the following: The client application browses the registry for available services. In later versions of the GridMiner, the registry should provide more detailed metadata and it should be possible to process more complex queries. The registry answers with URIs, called Grid Service Handles (GSH), to the factories of the target services. The client then uses the GMPPS factory to create a new instance of the preprocessing service.

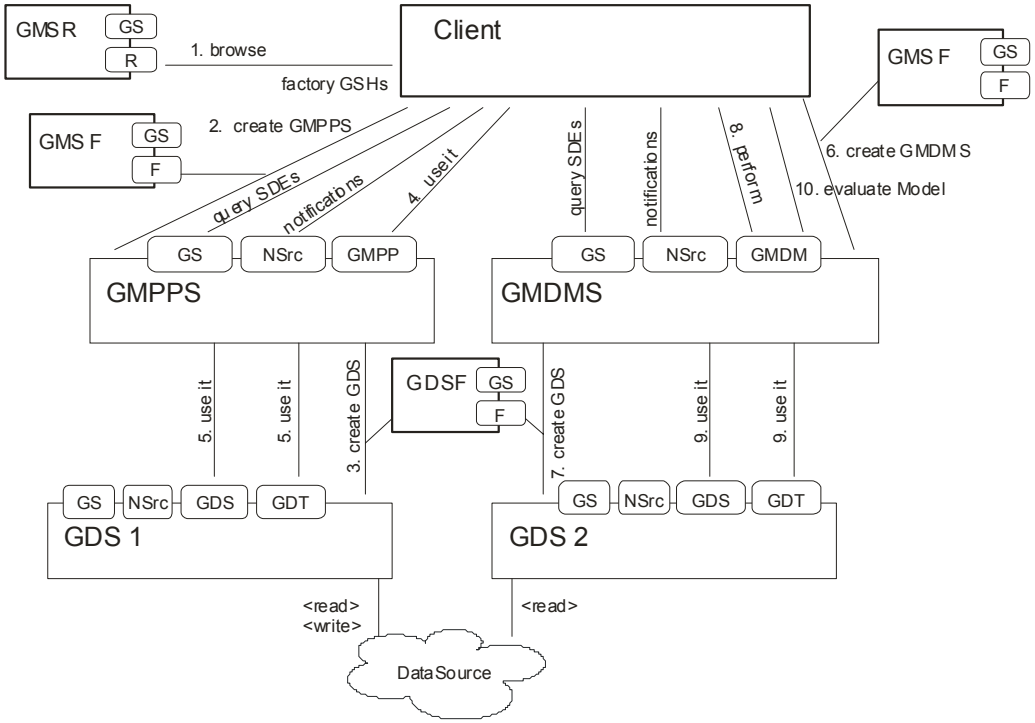


**Figure 4: Direct Usage of Preprocessing and DataMining Service.**

As defined by OSGA (see Foster, 2002), each Grid Service contains a set of service data elements (SDE), a collection of XML elements (comparable to fields within classes in object-oriented languages) that are accessible via the *FindServiceData* operation in the GridService port. GMPPS and the other services use these SDEs to provide instance-independent data (e.g. supported algorithms, available hard- and software) as well as instance-dependet data (e.g. status, progress, errors, results). The OSGA notification mechanism can be used to stay informed of the events the user is interested in after the client application has registered itself at the service.

In our example depicted in Figure 4, the client then instructs the GMPPS instance what it has to do by sending a description of the activity (in Web/Grid service scenarios this description is being sent as SOAP message according to the services WSDL interface definition). As we will see later a set of such individual activities can be assembled to a business process that is executed by the orchestration component.

Here the GMPPS uses a GridService that virtualizes one or more data sources, e.g. a mediation component as described in the sections above. The preprocessing service instance performs the specified tasks and writes the results back to the data source. Because data preprocessing may be long-running the service usage is asynchronous and the client application is being notified or in other contexts iteratively queries the SDEs for the instance's status.

After successful completion a data mining service instance is created via it's factory. The GMDMS also uses a underlying Grid data service for accessing the dataset and performs its data mining algorithm in this example on a single host.The service provides the resulting model within a generic service data element. After a final evaluation of the model generated in the previous step, the non-persistant service instances are destroyed.

## *Distributed Data Mining*

In this model, two or more Grid nodes simultaneously participate in the data mining task. The algorithms applied are either extensions of those developed for distributed-memory (shared-nothing) parallel architectures or new methods specially developed for geographically distributed systems, which inherently have high communication latency. Figure 5 shows a scenario involving a service instance created by the user application that creates four 'worker nodes' that perform a parallel data mining algorithm on their local data sets. The control instance GMDMS *0* acts as an intermediary virtualizing the parallelism. It creates, controls and destroys the worker nodes GMDMS *1* to GMDMS *n*.

Different communication models and protocols may be used for the p2p communication between the worker nodes. The main OGSA protocols SOAP over HTTP provide universality and platform-independence at the price of high latency and communication costs. Other protocols like Grid-MPI (see Foster and Karonis), JXTA[10], RMI, etc may be used instead for efficiency reasons.

[10] http://www.jxta.org

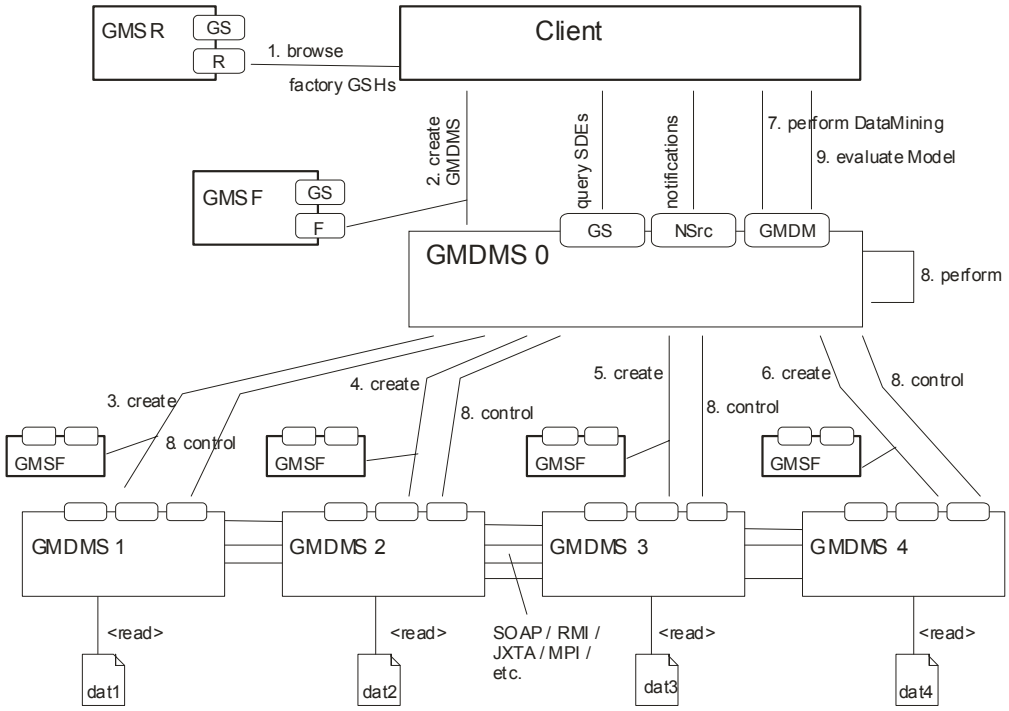


**Figure 5: Parallel Data Mining. A master service coordinates four distributed worker services executing a parallel data mining algorithm.**

*Meta-Learning* (see Prodromidis, Stolfo et.al. and Stolfo, 1997) has been suggested as related but different approach for handling massive amounts of data. It partitions the data into subsets, then independently applying a learning algorithm on each of the subsets, and afterwards combining the results of this distributed learning into one global model. An advantage of this approach is the minimal communication necessary and that there can even be used different algorithms to calculate the local results. This form of data and computation distribution is quite natural in Grid environments and, therefore, we also plan to develop data mining services based on this approach.

## *Orchestrated Knowledge Discovery*

*Business processes* consist of starting and ending points, a set of related activities and routing rules that build together a sequence of tasks that spans a functional unit. A *workflow* can be thought of as the implementation of the details, rules, activities and participants of a business process and a formal workflow description is needed for automated or semi-automated execution by an workflow engine.

What do data mining applications and business processes have in common? We think they are somewhat complementary. Although data mining is the central phase of knowledge discovery of databases it also involves data preparation (this includes data cleaning, data integration, data transformation, data filtering etc.), evaluation of data mining results, and presentation of the patterns found to the user.

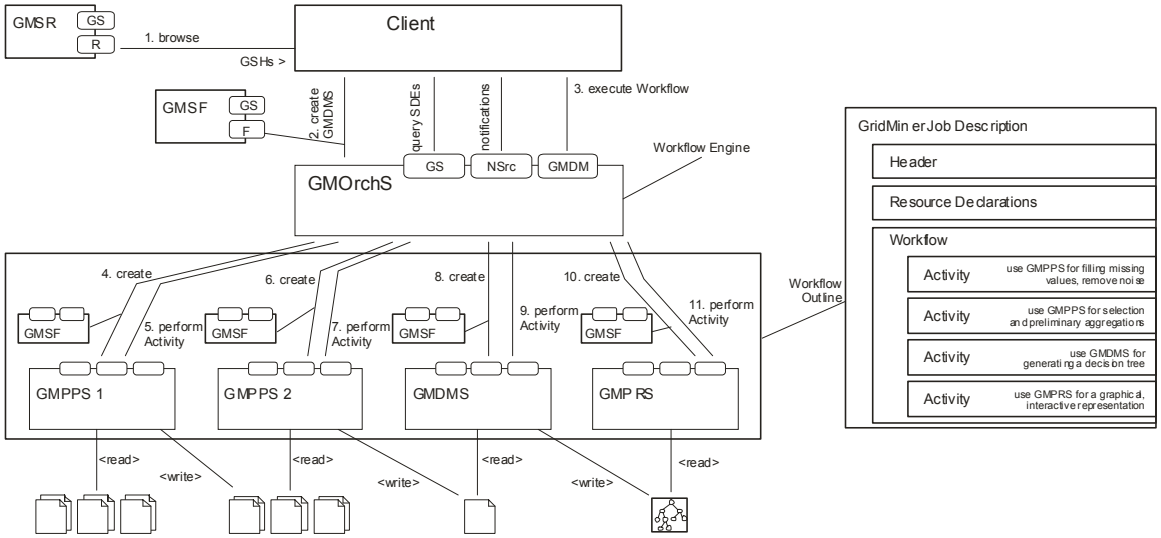


**Figure 6: In this scenario the GridMiner Orchestration Service is used as central component that coordinates a complex workflow involving several subsystems. The GMOrchS acts as workflow engine that executes a formal description of a business process.**

These individual activities can easily be grouped together to form a *knowledge discovery workflow*. We are elaborating a powerful, extensible mechanism that allows the user to specify the complete workflow in advance. We decided to use a specification language based on an XML-Schema-Grammar because there are already powerful parsers available, and it provides a human readable and editable format and good compatibility with the Grid Services, which are conformed to SOAP and OGSA.

Figure 6 illustrates such an scenario that consists of four activities (two preprocessing, a data mining and a finally a presentation activity) executed in sequence under control of our planned GridMiner Orchestration Service. The used workflow description is shown at the right side and explained in more detail in Section 'GridMiner Job Description Language'. The advantage for the user is complexity and distribution transparency by communicating with only one partner system, the GMOrchS.

In some contexts the a priori full job description approach may be more useful than sending a sequence of messages or RPCs.

- As web/grid service based systems usually have high communication cost, coarse grained service interfaces are encouraged. Our approach reduces the number of necessary messages and their content (parameter) complexity.
- Participating resources can be initialized, reserved, QoS requirements can be verified and performance predictions calculated.
- The system can easily generate an execution plan based on the input (job description, QoS) and policies/settings/states on the target host and attach it to the job queue.
- The requested job can be analyzed and checked for consistency and feasibility. This helps to reduce unnecessary load.
- To reduce learning efforts, a graphical user interface can be build that helps the user to create the job decription file.

## GridMiner Job Specification Language

Figure 7 outlines an instance of the GridMiner Job Specification Lanuge[11] (GM-JSL) for a knowledge discovery process using a dataset from the TBI application we mention in Section 'Introduction'. The type *Header* contains meta information including job name, detailed description, version info etc. More interesting are the *Resources* and the *Workflow* sections. In the *Resources* section, various kinds of resources like databases, files, filesets and nodetypes may be specified. In Figure 7, requirements (CPU architecture and frequency, operations system, memory) on computing elements are declared other elements may later refer to as host type via it's logical type-name *'linuxbox'*. Another important element in this section is the *dataset* element that declares a logical dataset (e.g. from a relational database or, like in this example, from a file).

The *Workflow* part defines a sequence of activities that are processed step by step. In our example in Figure 1 a typical workflow for our centralized data mining scenario is outlined. The job starts with two preprocessing steps, performs a decision tree mining with the C4.5 (see Quinlan, 1993) algorithm, and renders the resulting model into a graphical decision tree that is presented to the user. The first preprocessing tasks reads raw text files for cleaning and removing statistical noise. Then the second preprocessing activitiy integrates multiple datasets into the final dataset and performs some aggregations. The data mining service instance is created after completion of the previous steps and executes the data mining algorithm. The resulting model is written to another file, that is used as input by the presentation service that renders the decision tree and transports it to the user.

## Implementation Prototype

We implemented our prototype of the Data Mining Service as OSGA conformous GridService (see Foster, 2002) based on the Globus Toolkit 3 release[12]. The service runs in the Globus 3 Framework and is hosted within Jakarta-Tomcat[13] as a container. The client application is a standalone Java application also using Globus 3 Framework and Apache Axis[14].

During the first stage of our work, we focus on the data mining service architecture and issues of the hosting environment. Traditional data mining algorithms (e.g., classification and association rule mining) typically require that the internal data structures are memory resident. These algorithms have been sufficiently studied and also implemented within commercial data mining tools. Therefore, instead of reimplementing known algorithms, we make use of a freely available data mining system called Weka[15].

---

[11] In accordance with the literature where the term *job* is sometimes used to refer to instances of workflow descriptions.
[12] The Globus Toolkit, http://www.globus.org
[13] Jakarta Tomcat, http://jakarta.apache.org/tomcat
[14] Apache Axis, http://xml.apache.org/axis
[15] Weka, http://www.cs.waikato.ac.nz/ml/weka

```
<gmjsl:Job xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:gmjsl="http://jsl.dms.gridminer.org"
           xsi:schemaLocation="http://jsl.dms.gridminer.org dmjsl.xsd">
    <gmjsl:Header>
      <gmjsl:JobName> Glasgow Outcome Vienna 2</gmjsl:JobName>
      <gmjsl:Description> Decision Tree for TB1 dataset 'Vienna 2' </gmjsl:Description>
      <gmjsl:version> 0.9</gmjsl:version>
      ...
    </gmjsl:Header>
    <gmjsl:Resources>
      <gmjsl:HostType hostTypeID="linuxbox">
        <gmjsl:Param key="TYPE" value="Intel"/> <gmjsl:Param key="OS" value="Linux" />
        <gmjsl:Param key="CPU" value="2Ghz" /> <gmjsl:Param key="RAM" value="1GB" />
      </gmjsl:HostType>
      <gmjsl:File id="raw01" > <gmjsl:Include pattern="raw01.dat"/> </gmjsl:File>
      <gmjsl:File id="dat01" > <gmjsl:Include pattern="dat01.dat"/> </gmjsl:File>
      ...
      <gmjsl:DataSet id="tbiDS" >
        <gmjsl:Source ref="dat01"/>
      </gmjsl:DataSet>
    </gmjsl:Resources>
    <gmjsl:Workflow>
      <gmjsl:Activity xsi:type="gmjsl:Preprocessing" nodeType="linuxbox"
                      input="raw01" output="raw02" >
        <gmjsl:Action> clean and remove noise</gmjsl:Action">
        ...
      </gmjsl:Activity>
      <gmjsl:Activity xsi:type="gmjsl:Preprocessing" nodeType="linuxbox"
                      input="raw02" output="dat01" >
        <gmjsl:Action> aggregate and integrate</gmjsl:Action">
      </gmjsl:Activity>
      <gmjsl:Activity xsi:type="gmjsl:CentralizedDataMining" >
        <gmjsl:Action> perform C4.5</gmjsl:Action">
        <gmjsl:DataSet> dat01</gmjsl:DataSet">
      </gmjsl:Activity>
      <gmjsl:Activity xsi:type="gmjsl:PresentModel" >
        <gmjsl:Action> render decision tree</gmjsl:Action">
      </gmjsl:Activity>
    </gmjsl:Workflow>
</gmjsl:Job>
```

**Figure 7: Outline of a GM-JSL instance**

# CONCLUSIONS AND FUTURE WORK

In this paper we have described the research effort, which focuses on the application and extension of the Grid technology to knowledge discovery in Grid databases, an important, but non-traditional Grid application domain.

The integration of two comprehensive research fields like data mining on the one hand and the younger and still evolving Grid technologies on the other hand is as ambitious as extensive. On the way towards an open, service-based data mining system - that can interact with components and data sources in various concrete Grid infrastructures, that is open in architecture and design, and that builds upon a powerful mediation service - we have already taken the important steps.

In this research effort, have defined requirements, usage scenarios, a system architecture, and implemented a centralized prototype. Our overall system architecture is structured into the lower level mediation and the higher level mining services, and relies on the OGSA. A prototype implementation of the centralized Data Mining Service, has been developed to prove the feasibility of the described concept, and will also serve for future performance comparisons with the distributed versions that will be developed in the next months. These distributed versions will extend and implement approaches developed within current distributed data mining research efforts and investigate their behavior on our test-application (TBI) in our Grid testbed environment. Due to the requirements specified in Section 'GridMiner Requirements' another future research direction is the coupling of the proposed data mining services architecture with the Grid OLAP services architecture, which is being developed in another research task of our project.

# REFERENCES

W. ALLCOCK et al. Gridftp protocol specification. GGF GridFTP Working Group Document, September 2002.

C. BARU et al. XML-based information mediation with MIX. SIGMOD '99, 1999.

M. CANNATARO and D. TALIA. Knowledge grid: An architecture for distributed knowledge discovery. Communications of the ACM, January 2003.

M. CANNATARO, D. TALIA, and P. TRUNFIO. Knowledge grid: high performance knowledge discovery services on the grid. In Second Grid International Workshop, Denver, CO, USA, pages 38--50, November 2001.

A. CHERVENAK, I. FOSTER, C. KESSELMAN, C. SALISBURY, and S. TUECKE. The Data Grid:towards an architecture for the distributed management and analysis of large scientific datasets. To be published in the Journal of Network and Computer Applications, 2001.

National e Science Centre. http://umbriel.dcs.gla.ac.uk/NeSC/general/.

G. MAHINTHAKUMAR et al. Multivariate geographic clustering in a metacomputing environment using globus. In Supercomputing'99, Orlando, USA, November 1999.

F. BERMAN and A. J. G. HEY and G. FOX. Grid Computing: Making The Global Infrastructure a Reality. John Wiley & Sons, 2003.

Global Grid Forum. Grid Service Specification, Nov 2002.

I. FOSTER, C. KESSELMAN, J. NICK, and S. TUECKE. The physiology of the grid: An open grid services architecture for distributed systems integration, January 2002.

Ian FOSTER and Nicholas T. KARONIS. A grid-enabled mpi: Message passing in heterogeneous distributed computing systems.

GRIDS: e-Science to e-Business. ERCIM News, April 2001.

R. L. GROSSMAN and others (Eds.). Data Mining for Scientific and Engineering Applications (Massive Computing. Kluver Academic Publ., 2001.

J. HAN. Data Mining. Concepts and Techniques. Morgan Kaufmann, 2000.

Neil HONG, Amy KRAUSE, Susan MALAIKA, Gavin MCCANCE, Simon LAWS, James MAGOWAN,Norman W. PATON, and Greg RICCARDI. Grid Database Service Specification, Feb 2003. Global Grid Forum 7.

JOSHI, KARYPIS, and KUMAR. ScalparC: A new scalable and efficient parallel classification algorithm for mining large datasets. In IPPS: 11th International Parallel Processing Symposium. IEEE Computer Society Press, 1998.

H. KARGUPTA and P. CHAN (Eds.). Advances in Distributed and Parallel Knowledge Discovery. AAAI Press, 2000.

P. Z. KUNSZT and L. P. GUY. The open grid services architecture, and data grids. In F. BERMAN, A. J. G. HEY, and G. FOX, editors, Grid Computing: Making The Global Infrastructure a Reality, pages 65--100. John Wiley & Sons, 2003.

Alon Y. LEVY, Anand RAJARAMAN, and Joann J. ORDILLE. Querying heterogeneous information sources using source descriptions. In Proceedings of the Twenty-second International Conference on Very Large Databases, pages 251--262, Bombay, India, 1996. VLDB Endowment, Saratoga, Calif.

B. LUDAESCHER, A. GUPTA, and M. E. MARTONE. Model-based mediation with domain maps. 17th Intl. Conference on Data Engineering (ICDE), Heidelberg, Germany, IEEE Computer Society, April 2001.

W. MAURITZ, M. RUSNAK, and I. JANCIAK. Implementing scientific evidence-based guidelines: Case study of severe traumatic brain injuries. Clinical Research and Regulatory Affairs, 20(1):81--88, January 2003.

R. MOORE. Knowledge-Based Grids. Technical Report TR-2001-02, San Diego Supercomputer Center, January 2001.

Andreas L. PRODROMIDIS, Salvatore J. STOLFO, Shelley TSELEPIS, Terrance TRUTA, Jeffrey SHERWIN, and David KALINA. Distributed data mining: The jam system architecture.

J.R. QUINLAN. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

John C. SHAFER, Rakesh AGRAWAL, and Manish MEHTA. SPRINT: A scalable parallel classifier for data mining. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, Proc. 22nd Int. Conf. Very Large Databases, VLDB, pages 544--555. Morgan Kaufmann, 3--6 1996.

S. J. STOLFO, A. L. PRODROMIDIS, S. TSELEPIS, W. LEE, D. W. FAN, and P. K. CHAN. JAM: Java agents for meta-learning over distributed databases. In Proc. of the International KDD 97 Conference (1997), pages 74--81, 1997.

Y. J. TAM. Datacube: Its implementation and application in OLAP mining. MSc.Thesis, Simon Fraser University, Canada, September 1998.

M. J. ZAKI and C. T. HO (Eds.). Large-Scale Parallel Data Mining. Springer Verlag, LNCS 1759, 2000.