Towards an Open Service Architecture for Data Mining on the Grid*

Peter Brezany[1], Jürgen Hofer[1], A Min Tjoa[2], and 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 of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstrasse 9-11/E188, A-1040 Vienna, Austria
E-mail: tjoa@ifs.tuwien.ac.at

Abstract

Across a wide variety of fields, huge datasets are being collected and accumulated at a dramatical pace. The datasets addressed by individual applications are very often heterogeneous and geographically distributed, and are used for collaboration by the communities of users, which are often large and also geographically distributed. There are major challenges involved in the efficient and reliable storage, fast processing, 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 for distributed and high-performance data mining in Computational Grid environments. This architecture is designed and being implemented on top of the Globus 3.0 Alpha toolkit (it provides basic Grid services, such as authentication, information and resource management, etc.) and OGSA-DAI Grid Services (they provide basic access to Grid databases).

1 Introduction

Advanced applications are continuing to generate ever larger amounts of valuable data, but we are in danger of being unable to extract fully the latent knowledge within the data because of insufficient technology [29]. 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. Depending on the application area, the datasets 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. The volume of these datasets is already measured in terabytes and will soon total petabytes. They are often geographically distributed and their 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 dataset size, geographic distribution of users and resources, and computationally intensive analysis results in complex and stringent performance demands that, until recently, were not satisfied by any existing computational and data management infrastructure. Moreover, certain level of QoS and security has to be guaranteed for several classes of applications accessing the Grid data analysis services.

In tackling these problems, there are the beginnings of a new form of information technology known as the Computational Grid (or simply Grid) – an infrastructure that enables the integrated use of remote high-end computers, databases, scientific instruments, networks, and other resources. The fundamental challenge is to make Grid systems widely available and easy to use for a wide

*This research is being carried out as part of the research projects "Aurora" and "Modern Data Analysis on Computational Grids" supported by the Austrian Research Foundation.

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 [13]. 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 GridMiner for knowledge discovery in databases coupled to Computational Grids.

The technology developed is being validated and tested on an advanced medical application addressing treatment of traumatic brain injury (TBI) victims. 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 data collection 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 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. Section 2 outlines the principles that were followed in developing the current Grid technology, which we use as a basis for the GridMiner design. Section 3 discusses the requirements of parallel and distributed data mining on the GridMiner. The kernel part (Sections 4 and 5) 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 4. The availability of such services is crucial for the design of concepts for data mining services, which are presented in Section 5. Section 6 discusses related work, and we briefly conclude and outline the future work in Section 7.

2 The Grid

A Grid based computational infrastructure [4] couples a vide variety of geographically distributed computational resources (such as PCs, workstations, and supercomputers), storage systems, data sources, databases, libraries, computational kernels, and special purpose scientific instruments, and presents them as a unified integrated resource which can be shared by communities (*"virtual organizations"*) as they tackle common goals.

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. Today the grid infrastructure is capable of binding together more than just a few specialized supercomputing centers. It is more ubiquitous and can support diverse applications requiring large-scale computation and data. Essentially all major Grid projects are currently built on protocols and services provided by the *Globus* Toolkit (http://globus.org) 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) [12] 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[1] communities, OGSA defines a uniform exposed service semantics (the Grid service); defines standard mechanisms for creating, naming, and discovering transient service instances; provides lo-

[1]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).

cation 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)[2] interfaces and associated conventions, mechanisms required for creating and composing sophisticated 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 (http://www.gridforum.org) 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 1*, is already available.

So far, only a little attention was devoted to knowledge discovery on the Grid [5, 7, 20], and 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.

3 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 [12, 11], *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 [5]. 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 muss allow the interoperability with data warehouse and Online Analytical Processing (OLAP) services. Data mining and OLAP are two complementary methodologies, which, if applied in conjuction [15, 27], can

[2] see `http://www.w3.org/TR/wsdl12/`

provide powerful advanced data analysis solutions.

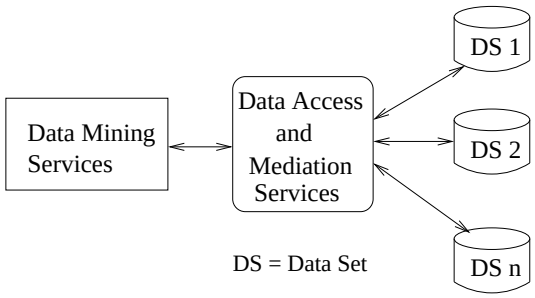


Figure 1. Data Access, Mediation, and Data Mining Services

4 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.

4.1 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 this case, 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. 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 $R_1 = (id, a_{1,1}, \ldots, a_{1,m}), \ldots R_n = (id, a_{n,1}, \ldots, a_{n,m})$, 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 data-sources 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. 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 [18]. 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 the situations like $(R_1 = (a_1, a_2), R_2 =$

```
<GDS:Activity name="Mediation" version="123">
        <GDS:Parameter name="MappingSchema" >
            <Table name="virtual_table" >
                <ConsistsOf name="source_table1" from="source_db1" />
                <ConsistsOf name="source_table2" from="source_db2" />
            </Table>
        </GDS:Parameter>
</GDS:Activity>
```

Figure 2. Outline of a Mapping Schema instance for a virtual table which is distributed over two databases having the same schema.

```
<?xml version="1.0" encoding="UTF-8" ?>
<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 Mining for Traumatic Brain Injury Patterns in DataSet 'Vienna 2'
        </GMJSL:Description>
        <GMJSL:JSLVersion> Text </GMJSL:JSLVersion> <GMJSL:JobSubmitter> Text </GMJSL:JobSubmitter>
    </GMJSL:Header>
    <GMJSL:Resources>
        <GMJSL:DataSet xsi:type="GMJSL:RDBMSDataSet" datasetID="ds1" >
            <GMJSL:name> Vienna 2 </GMJSL:name> <GMJSL:host> 127.0.0.1 </GMJSL:host>
            <GMJSL:database> tbi-vienna2 </GMJSL:database>
            <GMJSL:query type="SQL92" > SELECT * FROM a, b WHERE a.id=b.id </GMJSL:query>
        </GMJSL:DataSet>
        <GMJSL:NodeType nodeTypeID="linuxbox" >
            <GMJSL:params>
                <GMJSL:Param key="TYPE" value="Intel"/> ] <GMJSL:Param key="OS" value="Linux" />
                <GMJSL:Param key="CPU" value="2GHZ" /> <GMJSL:Param key="RAM" value="1024MB" />
            </GMJSL:params>
        </GMJSL:NodeType>
    </GMJSL:Resources>
    <GMJSL:Workflow>
        <GMJSL:Activity xsi:type="GMJSL:CreateCDMSActivity" >
            <! – ... – >
            <GMJSL:DataSetBinding¿ ds1 </GMJSL:DataSetBinding">
            <GMJSL:NodeType> linuxbox </GMJSL:NodeType>
        </GMJSL:Activity>
        <GMJSL:Activity xsi:type="GMJSL:PerformClassificationActivity" >
            <! – ... – >
            <GMJSL:Algorithm> C4.5 </GMJSL:Algorithm>
        </GMJSL:Activity>
        <GMJSL:Activity xsi:type="GMJSL:PresentModelActivity" >
            <! – ... – >
            <GMJSL:Type> Decision Tree </GMJSL:Type>
        </GMJSL:Activity>
    </GMJSL:Workflow>
</GMJSL:Job>
```

Figure 3. Outline of a GM-JSL instance. Within the `Resources`-Tag there is an example definition of a dataset from a relational database and of requirements for a linux node. A possible workflow for a C4.5 decision tree mining operation on a computing element is defined within the `Workflow` Tag.

$(a_1, a_3)) \wedge a_2 = a_3$.

4.2 GridMiner Mediator Configuration

We designed the Mediator as a middleware service, which connects to the participating data sources, integrates them logically into a *virtual data source (VDS)*[3], sends queries to them, and combines and deliver the results in a flexible way. For this purpose, a lot of information is needed to setup. On one hand, we provide the metadata information about the involved data sources[4] as required by the OGSA-DAI support software, and on the other hand we proposed a set of metadata structures, which are needed to construct a VDS. For example, Figure 2 outlines a *mapping schema*, which must be provided, to resolve logical names to their physical representation and location.

In the first prototype implementation, we use an XML-Script for describing the mediation service built-up process and characteristics of the VDS. For very complex or often used mediation tasks, this information can be stored in the service called *GridMediatorFactory*, which generates the mediation service. For highly dynamic federations, the configuration metadata can also be passed to the GridMediatorFactory at runtime.

4.3 Implementation of the Mediation Service

In order to avoid building a new proprietary solution and reimplementing well solved aspects of Grid services, we have decided to extend the free available OGSA-DAI implementation to provide the *virtual data source (VDS)* functionality. Our VDS offers the same metadata as a *normal* Grid data source, like a logical schema, and technical aspects, like the supported query language[5], and hides the distribution and heterogeneity of the participating data sources by reformulating requests against the provided logical schema with the help of the given *mapping schema*. The mediator identifies the relevant data sources and develops a query execution plan for obtaining the requested data. The plan typically breaks the original query into fragments to be delegated to individual data sources, and specifies additional processing to be performed by the mediator to further order, merge or join the data. For our example in Figure 2, this will lead to the parallel execution of two Select-Statements and following merging of the results. 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 [2] will be supported.

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

[4] logical and physical schema, access properties, etc.

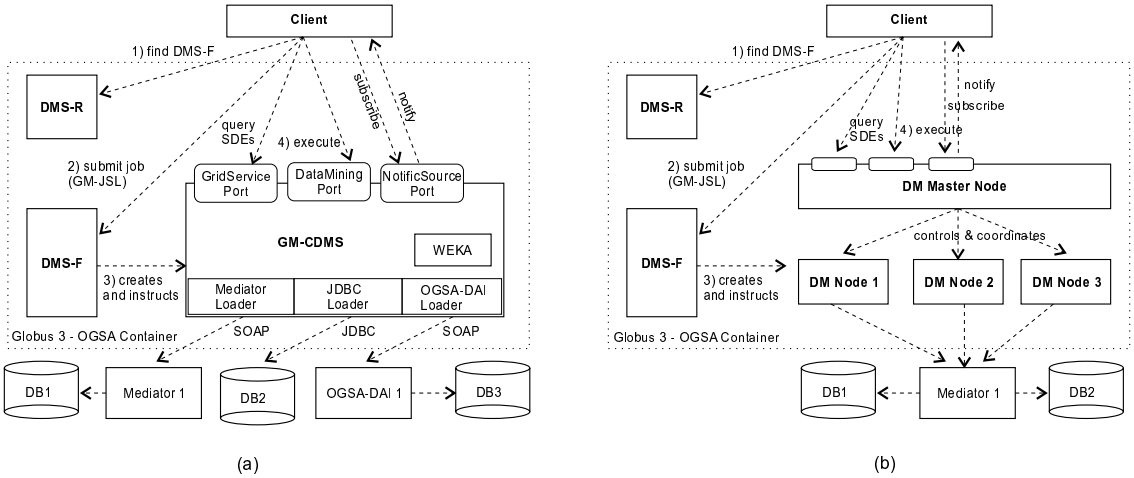
[5] a subset of the SQL language

5 Data Mining Service

We are approaching the issues addressed by our work by defining two execution models for our data mining service. First, we are *"grid-enabling"* an already available data mining toolkit for demonstration and performance comparison reasons as well as to proof feasibility of our concept. Afterwards, a distributed version is being developed that makes use of the loosely-coupled, distributed resources (hard- and software, datasets) unified within a grid.

1. Centralized data mining (CDM).
In the simplest case, data mining is performed on one node that loads the data into it's memory, executes the necessary algorithms and presents the results to the user process. The node may involve sequential or parallel computing power. The Service virtualizes the concrete implementation but allows the user to select the used algorithms and tools. It is open in design and architecture and easily extensible. Usually, in the whole knowledge discovery process, there are data cleaning, integration and selection steps preceding the data mining step. While data integration is provided by the Mediator described in the previous section, the Data Mining Service can also include several data preprocessing tasks (e.g., data cleaning, selecting the attributes of interest, etc.). We also provide a separated set of *Data Mining Support Services*, which involve tools for data preprocessing.

2. Distributed data mining (DDM).
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. One example of the last approach is *meta-learning* [22, 26], which handles massive amounts of data by partitioning it 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



Components: DMS-Registry (DMS-R), DMS-Factory (DMS-F), GridMiner Centralized Data Mining Service (GM-CDMS).

Figure 4. Static View on System Architecture: (a) a centralized layout using Weka with three different data source loaders; (b) a distributed data mining layout with three nodes controlled and coordinated by a master node.

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 are also developing data mining services based on this approach.

5.1 GridMiner Job Specification Language (GM-JSL)

Data mining is the central phase of knowledge discovery of databases, which 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.

We are developing a powerful, extensible mechanism that allows the user to specify the complete knowledge discovery 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.

The a priori full job description approach has several advantages over sending a sequence of messages or applying remote procedure calls, as briefly stated below.

1. The interface complexity of software components involved can be reduced dramatically.

2. Participating resources can be initialized, reserved, QoS requirements can be verified and performance predictions calculated.

3. An execution plan and schedule can be generated and optimized based on the job description, QoS parameters, current state, and performance predictions of components involved.

4. The requested job can be analyzed and checked for consistency and feasibility. This helps to reduce unnecessary load.

5. To reduce learning efforts, a graphical user interface can be build that helps the user to create the JSL file.

Figure 3 outlines a GM-JSL instance for a grid-enabled data mining process in a dataset of the TBI application we mention in Section 1. The type `Header` contains meta information including job name, description of the GM-JSL version. More interesting are the `Resources` and the `Workflow` sections. In the `Resources` section, it is possible to define the inputs of the data mining service in form complexTypes (an XML-Schema concept) inherited from the abstract type `DataSet`.

A dataset for relational databases is predefined but other dataset types (e.g. for flat files, XML-databases) can be easily defined as XML-Schema subclasses. It is also possible to specify requirements on the type of nodes used for execution. We define a 'linuxbox' as Intel-based personal computer, running Linux, with CPU frequency greater than 2 GHz and 1 GB capacity of RAM.

The `Workflow` part defines a sequence of activities that are processed step by step. In our example in Figure 3, we show a typical workflow for our centralized data mining scenario. A centralized data mining service (CDMS) instance is created on a machine that satisfies the requirements for `linuxbox`es. After its creation, the service is bound to DataSet `ds1` specified in the `Resources` section, and a C4.5 classification algorithm is applied. Finally, the model found is presented as a decision tree on the output channel.

5.2 Implementation of the Centralized Data Mining Service

We implemented our prototype of the Data Mining Service as GridService [6] based on the Globus Toolkit 3 Alpha release[7]. The service runs in the Globus 3 Framework hosted within Jakarta-Tomcat[8]. The client application is a standalone Java application that also makes use of the Globus 3 Framework and Apache Axis[9].

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[10]. This software package has been developed at the University of Waikato in New Zealand and primarily implements the methods discussed by Witten and Frank [30]. Since it is a pure Java program with a nice documented API it is well suited for our purpose and we use it as a data mining engine in our centralized scenario.

Figure 4(a) illustrates the structure and a possible workflow of our centralized data mining service (further on referred to as CDMS) within the GridMiner architecture. The client process contacts the community registry[11] (DMS-R) by sending a search query for available data mining factories[12] (DMS-F). These factories provide metadata and state data with the standard OGSA serviceDataElement-mechanism, hence the user could choose which one fits best if the registry returns a set of factory-GSHs (Grid Service Handles) as result. The user specifies his data mining task in terms of the GM-JSL (see section 5.1) and sends this job description to the DMS-F. The factory analyzes the input, creates the data mining service (GM-CDMS) instance on a suitable host and instructs the service what to do by forwarding the job description. The instance itself prepares for execution by connecting to data sources, loading of the necessary algorithms and creation of an execution plan. Finally the instance registers itself to the registry.

After successful service creation, the user can subscribe to the `NotificationSource` portType provided by the newly created service instance. Hence, the client is asynchronously notified of service-related events he is interested in (e.g. progress, state, errors). On invocation of the services `execute` operation, GM-CDMS starts processing the specified tasks.

The CDMS implements the OGSA standard ports `GridService` (for lifecycle management and service data element queries), `NotificationSource` that allows client applications to subscribe to topics as described above, and the service specific `DataMiningPort`. The `GridService` port provides operations for initializing the service (by the factory) and executing the submitted Job (in terms of GM-JSL).

The DataMiningPort uses Service Data Elements (SDEs) for holding lifecycle, progress and state information, the execution plan and after termination of the learning process the result is also stored with a service data element in XML notation. Clients can use the OGSA-predefined SDE query mechanism for accessing the content of service data elements.

Our current implementation prototype is able to load its data either directly through JDBC or from our Mediator prototype; a third way, from OGSA-

[6] A GridService is a Web Service that provides a set of well-defined interfaces and that follows specific conventions [12]

[7] http://www.globus.org/ogsa/releases/alpha/index.html

[8] http://jakarta.apache.org/tomcat

[9] http://xml.apache.org/axis

[10] http://www.cs.waikato.ac.nz/ml/weka

[11] a standard GridService implementing the OGSA `Registry` portType

[12] a user-defined GridService implementing the OGSA `Factory` portType

```
<GMJSL:Workflow>
    <GMJSL:Activity xsi:type="GMJSL:CreateDDMSActivity" >
        <GMJSL:CreateMasterService name="ctrl" services="s1 s2 s3" />
        <GMJSL:CreateSlaveService name="s1" dataset="ds1" nodeType="linuxbox" />
        <GMJSL:CreateSlaveService name="s2" dataset="ds2" nodeType="linuxbox" />
        <GMJSL:CreateSlaveService name="s3" dataset="ds3" nodeType="linuxbox" />
        <!- ... ->
    </GMJSL:Activity>
</GMJSL:Workflow>
```

Figure 5. Creation of master and slave instances in a distributed data mining scenario

DAI[13], is currently being developed.

5.3 Distributed Data Mining Service

The architecture of this service is depicted in Figure 4(b). A Grid data mining node is first determined to act as a coordinator, and we will call it the *Master*. The Master node is initially instructed by the user to perform a certain data mining task by means of the GM-JSL. The Master is responsible for instruction and coordination of the working nodes. The factory creates one DM Master Node and three DM working Nodes. The working nodes either access a single mediation service or access distributed data sources if specified.

Figure 5 outlines the GM-JSL language constructs, which allow to describe the workflow of distributed data mining applications on the Grid. It is possible to describe the layout of multiple service instances that parallel mine in different datasets, their coordination, communication, lifetime, and the requirements on QoS, hardware and software.

6 Related Work

There are already many publication on parallel and distributed data mining (e.g., [1, 9, 23, 24, 25, 31, 8, 14, 16, 17, 21, 26, 28]). An attempt to design an architecture for performing data mining on the Grid was presented in [6]. 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 [20]. Mahinthakumar et al. [10] 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" [3, 19].

[13] http://www.ogsadai.org.uk

7 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. 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

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *U. Fayyad (ed.), Advances in Knowledge Discovery and Data Mining, AAAI Press*, pages 307–328, Menlo Park, CA, 1996.

[2] W. Allcock et al. Gridftp protocol specification. GGF GridFTP Working Group Document, September 2002.

[3] C. Baru et al. XML-based information mediation with MIX. SIGMOD '99, 1999.

[4] R. Buyya, J. Giddy, and D. Abramson. An economy grid architecture for service-oriented grid computing. In *10th IEEE International Heterogeneous Computing Workshop (HCW 2001), In conjunction with IPDPS 2001*, San Francisco, California, USA, April 2001.

[5] M. Cannataro, D. Talia, and P. Trunfio. Design of distributed data mining applications on the knowledge grid.

[6] M. Cannataro, D. Talia, and P. Trunfio. Knowledge grid: high performance knowledge discovery services on the grid. In *Second International Workshop, Denver, CO, USA*, pages 38–50, November 2001.

[7] M. Cannataro, D. Talia, and P. Trunfio. Distributed data mining on the grid. *Future Generation Computer Systems*, 18:1101–1112, 2002.

[8] J. Chattratichat, J. Darlington, Y. Guo, S. Hedvall, M. Köler, and J. Syed. An architecture for distributed enterprise data mining. In *Proceedings of HPCN Europe 1999, Lecture Notes in Computer Science, 1593*, pages 573–582, Berlin, May 1999.

[9] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocesors. In *M. J. Zaki and C.-T. Ho (eds), Large-Scale Parallel Data Mining, Springer-Verlag, LNCS 1759*, pages 245–260, 1999.

[10] G. Mahinthakumar et al. Multivariate geographic clustering in a metacomputing environment using globus. In *Supercomputing'99*, Orlando, USA, November 1999.

[11] Global Grid Forum. *Grid Service Specification*, Nov 2002.

[12] 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.

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

[14] R. Grossman, S. Bailey, S. Kasif, D. Mon, A. Ramu, and B. Malhi. The preliminary design of Papyrus: a system for high performance, distributed data mining over clusters, meta-clusters and super-clusters. In *Proceddings of the International KDD 98 Conference*, pages 37–43, 1998.

[15] J. Han. *Data Mining. Concepts and Techniques*. Morgan Kaufmann, 2000.

[16] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective data mining: a new perspective toward distributed data mining. In *In H. Kargupta and P. Chan (eds.) Advances in Distributed and Parallel Knowledge Discovery (AAAI Press)*, 1999.

[17] H. Kimm and T.-W. Ryu. A framework for distributed knowledge discovery system over heterogeneous networks using CORBA. In *Proceedings of the KDD2000 Workshop on Distributed and Parallel Knowledge Discovery*, 2000.

[18] 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.

[19] 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.

[20] R. Moore. Knowledge-Based Grids. Technical Report TR-2001-02, San Diego Supercomputer Center, January 2001.

[21] R. Moore, C. Baru, R. Marciano, A. Rajasekar, and M. Wan. Data-intensive computing. In: I. Foster and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure. Morgan - Kaufmann, 1999.

[22] Andreas L. Prodromidis, Salvatore J. Stolfo, Shelley Tselepis, Terrance Truta, Jeffrey Sherwin, and David Kalina. Distributed data mining: The jam system architecture.

[23] T. Shintani and M. Kitsuregawa. Mining algorithms for sequential patterns in parallel: Hash based approach. In *2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, 1998.

[24] T. Shintani and M. Kitsuregawa. Parallel generalized association rule mining on large-scale PC cluster. In *M. J. Zaki and C.-T. Ho (eds), Large-Scale Parallel Data Mining, Springer-Verlag, LNCS 1759*, pages 145–160, 1999.

[25] A. Srivastava, E. H. Han, V. Kumar, and V. Singh. Parallel formulation of decision-tree classification algorithms. In *Data Mining and Knowledge Discovery: An International Journal 3*, pages 237–261, 1999.

[26] 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.

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

[28] G. Williams et al. The integrated delivery of large-scale data mining: The ACSys data mining project. In *M. J. Zaki and C.-T. Ho (eds), Large-Scale Parallel Data Mining, Springer-Verlag, LNCS 1759*, pages 24–54, 1999.

[29] R. Williams et al. Large scientific databases. Joint EU-US Workshop, Annapolis, USA, September 1999.

[30] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.

[31] M. J. Zaki. Scalable data mining for rules. PhD Thesis, University of Rochester, 1998.