

Digging Deep into the Data Mine with DataMiningGrid

V. Stankovski^{1,*}, M. Swain², V. Kravtsov³, T. Niessen⁴, D. Wegener⁴, M. Röhm⁵, J. Trnkoczy¹,
M. May⁴, J. Franke⁵, A. Schuster³ and W. Dubitzky²

¹ *University of Ljubljana, FGG-KGI, SI-1000 Ljubljana, Slovenia*

² *University of Ulster, Cromore Road, Coleraine BT52 1SA, Northern Ireland*

³ *Technion - Israel Institute of Technology, Technion City, 32000 Haifa, Israel*

⁴ *Fraunhofer Institute for Intelligent Analysis and Information Systems, D-53754 Sankt Augustin, Germany*

⁵ *DaimlerChrysler, Research Center, D-89013 Ulm, Germany*

*Communicating author

University of Ljubljana, FGG-KGI, Jamova cesta 2, SI-1000 Ljubljana, Slovenia

Phone: +386-41-200-565, Fax: +386-1-425-0681; E-mail: vlado@stankovski.net

Abstract

The growing computerization in modern knowledge and technology sectors is generating huge volumes of electronically stored data. *Data mining* technology is often employed to make sense of these data. However, as modern data mining applications increase in complexity, so do their demands for resources. Grid computing is one of several emerging networked computing paradigms promising to meet the requirements of heterogeneous, large-scale and distributed data mining applications. Despite this promise, there are still too many issues to be resolved before grid technology is commonly applied to large-scale data mining tasks. To address some of these issues, we developed the DataMiningGrid system, which principally differs from similar systems by its ability to integrate a diverse set of programs and application scenarios within a single framework. The system's key features include high performance and scalability, sophisticated support for relevant standards, different user types, and flexible extensibility. The software is available as open source.

Keywords: Distributed architectures, Distributed applications, Data mining, Middleware/business logic

Introduction

Data mining is a methodology which facilitates the automated extraction of potentially useful information from large volumes of data. Increasing data volumes and the geographic distribution of applications in many modern knowledge sectors are fueling the need for novel data mining solutions. Recent trends that leverage network-based infrastructures include distributed data mining in peer-to-peer networks, mobile and embedded devices, sensor networks, and both parallel and privacy-preserving data mining.

Data mining in *grid* [1] computing environments represents a specific incarnation of distributed data mining that is motivated by resource sharing via local and wide area networks [2]. Increased performance, scalability, access and resource exploitation are the key drivers behind such endeavors. However, enabling large-scale data mining applications for the grid is hampered by several factors. To begin with, grid computing itself is relatively new and relevant standards and technologies are still evolving. Then there exists a plethora of data mining technologies and a staggering number of largely varying data mining application scenarios. Finally, data mining users range from highly domain-oriented end users to technology-aware specialists. To the former user group transparency and ease-of-use is paramount, whereas the latter group needs to be in control of detailed data mining and grid technology aspects.

In the DataMiningGrid project we have aimed to address the requirements of modern data mining application scenarios, in particular those which involve sophisticated resource sharing. A detailed technical account of the DataMiningGrid system is given here [14]. In this article we give a higher-level introduction to the system, and describe its main requirements, architecture and features. We also present system validation results and discuss related technologies.

1 Use Cases and Requirements

We analyzed use case scenarios from a wide range of application areas. These included ecological modeling, computational biology and bioinformatics, customer relationship and quality management in the automotive industry, performance monitoring and fault diagnosis in network computing systems, and analysis of distributed data in large-scale medical studies. A key challenge we faced was the need to integrate the requirements of all these complex application scenarios within a single framework.

Grid-enabling data mining applications should not require modification of their source code. We had to grid-enable numerous data mining applications, ranging from the well-known toolkit Weka, which runs under JVM to proprietary machine learning, text-mining and ontology-learning applications that only execute on specific platforms and use special libraries. Bearing in mind the diversity of the application scenarios, the system could not be restricted to specific data mining applications, tools, techniques or algorithms and had to support various types of data sources, including database management systems and data sets stored in file systems.

Efficiency, novel use and improved resource exploitation. Our grid-based data mining environment had to offer one or more of the following benefits: increased performance for data mining applications (speed-up, throughput), high scalability to serve more users and more demanding applications, possibilities for creation of novel workflow based data mining applications (e.g. by combining several data mining tasks to cover pre-processing, processing and post-processing stages), and improved exploitation of existing hardware and software resources.

Usability. To ensure that the system is easy to use, the grid's intricate technological details had to be hidden from the domain-oriented users, while at the same time we had to provide ways for the technically knowledgeable users to define, configure and parameterize details of the grid-enabled data mining application.

Resource brokering. Resource brokering is a central feature of grid systems [3]. The grid-based data mining system should support and optimize (1) matching available resources to job requests, (2) globally scheduling the execution of matched jobs, and, (3) managing and monitoring job executions, including data staging. In particular the data mining aspects of the application should guide the resource-job managing process.

2 DataMiningGrid System and Features

The collected requirements imply various complex system components, such as a workflow editor and manager, a resource manager, grid-based data management, security mechanisms, execution management system, and so on. In order to address these requirements, the DataMiningGrid system is designed according to three principles: *services-*

oriented architecture (SOA), standardization and open technology. SOA promotes the sharing of geographically dispersed business functions in a flexible way. We exploited SOA's main advantages, such as loose coupling, ease and flexibility of reuse, scalability, interoperability, and service abstraction from underlying technology [4]. To ensure the DataMiningGrid system's seamless evolution, two important distributed computing standards are also supported: the Open Grid Service Architecture (OGSA) and the Web Services Resource Framework (WSRF).

Figure 1 depicts the DataMiningGrid system architecture in four layers. Generally, components in higher layers make use of components organized in lower layers. The layer at the bottom represents *software and hardware resources*, the *Globus Toolkit 4* layer depicts some of the system's core grid middleware components, the *high-level services* layer shows components providing central DataMiningGrid services, and the *client components* layer depicts the DataMiningGrid applications' client side components.

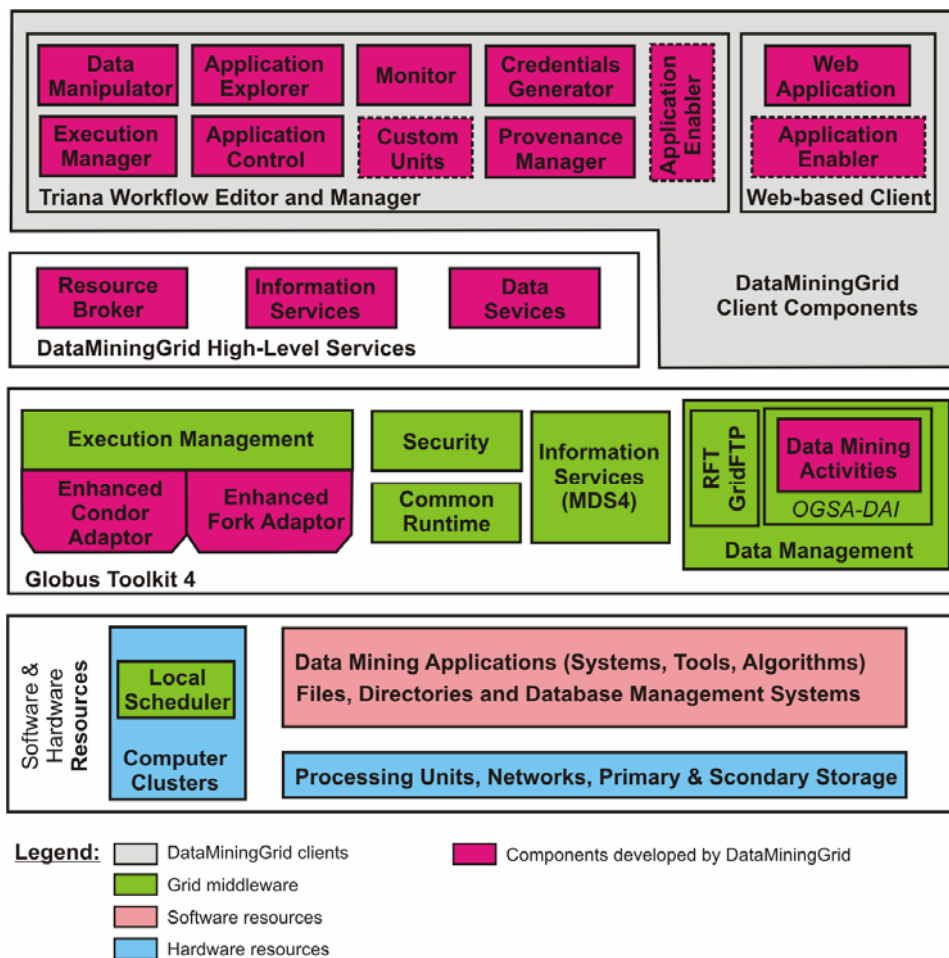


Figure 1. The DataMiningGrid system architecture.

A metadata schema definition, which we call an Application Description Schema (ADS) is used to manage user interaction with the components of the system architecture, to grid-enable existing data mining applications, to register and search for software on the grid, to match jobs with suitable computational resources and to dynamically create user interfaces. Instances of the ADS are XML documents at various levels of specification, which are passed between the system components.

A fully specified instance of the ADS contains:

General information about the application i.e. a narrative description, program vendor, version etc.

Implementation aspects of the application, such as operating system (Windows or Linux), programming language, executable and library file(s) and their storage location, and requirements, such as CPU architecture, memory and disk space. By specifying the detailed requirements of an application it is possible to ensure that it will be only executed in the correct environment.

Parameters that define the data mining application's execution behavior. At runtime these parameters are used to specify different command line options, and can be used to create multiple simultaneous executions that, for example, perform parameter sweep over different data inputs (files or directories) or application options (including numeric and string values).

Data mining aspects of the application, which are used to facilitate its fast discovery on the grid. It includes the domain-specific problem addressed, the data mining task, method, algorithm, software, and the Cross Industry Standard Process (CRISP) for the application's data mining phase.

The ADS is expressive enough to accommodate data mining applications from a wide range of platforms, technologies, application domains and sectors and has been tested with eight different application domains.

Software and hardware resources layer

Software resources include data resources, such as database and file systems, and data mining applications, while typical hardware resources include processing units, primary and secondary storage devices, and computer clusters.

Globus Toolkit 4 layer

The Globus Toolkit 4 (GT4) layer of the DataMiningGrid architecture provides core grid middleware functionality [5]. GT4 is an open-source toolkit for building grid systems provided by the Globus Alliance, which meets the requirements of the OGSA, implements the WSRF and provides the following critical components.

The *Information Services: Monitoring & Discovery System 4 (MDS4)* of the GT4 is essentially a system for storing and searching dynamically changing distributed XML documents describing grid software and hardware resources and their status.

The *data management* components of the GT4 include grid file transfer functionality (GridFTP, Reliable File Transfer service (RTF)) and data services (OGSA-DAI [6]). The GT4 data access and integration tools (OGSA-DAI component) provide grid-enabled access to files, relational databases, and XML databases. In addition to GT4 data management functions, we use the Java Commodity Grid Kit [7] to allow application client machines, which do not have a GridFTP server installation, to manipulate and transfer files to and from grid storage servers.

The DataMiningGrid makes extensive use of GT4's *execution management* tools to handle the initiation, monitoring, management, and coordination of remote computations. GT4 is used as front end to either single machines or computational clusters, however, it does not *per se* implement a global scheduling functionality. GT4 provides a Web service version of the Grid Resource Allocation and Management (WS-GRAM) interface, which is responsible for either the job's execution on the local single machine, or for forwarding and controlling execution on the local cluster manager, such as Condor, LSF etc. In order to overcome the relatively limited abilities of WS-GRAM to interact with the local cluster managers, the DataMiningGrid enhanced WS-GRAM's execution adaptors by adding data transfer functionality to deal with recursive directory structures, and by enabling the execution of Java applications including the proper handling of JVM parameters, class path information and so on.

DataMiningGrid high-level services

In the DataMiningGrid system, a *job* refers to one DataMiningGrid-enabled data mining application (executables and associated libraries) that needs data files (data from databases must be converted to files before job submission) and appropriate computing resources in order to be executed, while *multi-jobs* are collections of such jobs (e.g. parameter sweeps).

The *Resource Broker Service* was built on the basis of the GridBus Resource Broker [8], which was modified to provide adherence to the WSRF standard. Hereafter, we refer to our enhanced GridBus Resource Broker Service

simply as the Resource Broker, for the sake of brevity. It takes as input an instance of the ADS, which specifies a requested job or multi-job for execution. The Resource Broker determines the list of available hardware resources, from all administrative domains. From that list, the Resource Broker determines the best matching resources according to the user's requests and data mining application requirements. Then, it performs job submission to the selected sites. In particular, the submission process includes data and data mining application staging and setting up the environmental variables. Afterwards, the Resource Broker is responsible for monitoring the jobs, and when a job completes it performs data stage-out and clean-up tasks.

The *DataMiningGrid Information Services* are designed to support the discovery, characterization and monitoring of various resources and services. A registry is created via the underlying MDS4 and is used to keep records of grid-enabled software resources (i.e. ADS instances, which describe data mining applications) and to provide information on all other resources (e.g., available clusters, storage and CPU capacity, operating systems and load of resources). The registry can be searched for software resources according to ADS attributes, such as application name, vendor, version, functional description etc. The Resource Broker requires the registry to plan, allocate and carry out job execution.

The *DataMiningGrid Data Services* involve extensions to OGSA-DAI [6] in order to provide data mining specific functionality for data sets configured as OGSA-DAI data resources, such as XML and relational databases. These are accessed by a set of clients developed with the OGSA-DAI API [6] and can be used for integrating distributed databases, performing data transformations, calculating data summaries and formatting data 'on-the-fly' to support different file formats. File resources are not usually manipulated using OGSA-DAI, but via other DataMiningGrid client components.

DataMiningGrid client components

We designed the DataMiningGrid system with three main user groups in mind: administrators that manage the hardware and middleware components, developers that build new grid-based applications, and end-users that select, set-up parameters and launch applications for execution. For the later two user groups, we have envisioned two application client types, a *workflow editor* for more complex and interactive applications and hard-wired *Web clients*, which provide much simpler interactions. Web clients are intended to be easy to use, and are designed specifically for domain-oriented end users with limited interest or knowledge of the underlying data mining and grid

technology. The workflow editor is based on Triana [9] and is designed to facilitate fine-grained user control of data mining applications, since it supports flexible workflow composition, parameter settings, input and output data-flows etc. It also provides components for remotely browsing and viewing files, and transferring them to and from the grid storage servers. Type-checking is applied to connections between workflow components and to any parameters which are manually entered by users.

Using the DataMiningGrid system

The DataMiningGrid's architecture is best understood by viewing it in action, from the perspective of different types of users. Here we outline all the necessary steps required to run a simple DataMiningGrid workflow application.

Let us consider an example in which it is necessary to grid-enable an existing data mining application. To do this the application developer specifies the properties and requirements of the application in a uniform way by partially filling a new instance of the ADS. The actual software (executable and libraries) is uploaded to a grid storage server, while a partially filled ADS instance is created and registered with the DataMiningGrid Information Services. Following this, the data mining application can instantly be discovered, configured and run in the grid environment by anyone who has the necessary permissions.

Before using the DataMiningGrid workflow components, an end-user must have installed Triana [9], the DataMiningGrid components and a valid certificate. The end-users part of the DataMiningGrid components consists of Java libraries, which contain the client software that is used to manipulate data resources, select data mining applications and orchestrate all the services needed to execute applications in the grid.

Now the end-user uses the workflow editor to compose a simple workflow.

The first step in the workflow involves selecting the data, which can be performed in three ways: first, via an OGSA-DAI client that accesses the Data Services, secondly by selecting local data which will then be uploaded to a default GridFTP server, and finally by specifying files or directories that are already present on a GridFTP server, such as the results of a preceding run.

In the next step, the end-user selects a data mining application, using a workflow component, which uses the Information Services, to query the application registry for the previously grid-enabled data mining application, and its ADS instance is selected.

Now the user interface in the workflow editor is dynamically configured according to the specifications of the ADS instance. As described earlier, this interface provides slots for the end-user to instantiate fields in the ADS, i.e. for specifying mutable options, such as algorithm parameters, data inputs, and so on. Once the end-user specifies the execution options the ADS instance is passed to the Resource Broker for execution. The end-user now has little involvement in the process until the jobs have completed, except to optionally monitor their execution status.

The Resource Broker uses the Information Services to determine the available computational resources. It matches the requirements of the ADS instance to descriptions of available resources and selects suitable resources for execution. The Resource Broker uses RFT to transfer the executable (e.g. a Java Jar file or a C binary and related libraries) and the uploaded data to each selected grid site, which has a GT4 WS-GRAM installation. WS-GRAM may then use its enhanced Condor Adaptor to submit the jobs to its local Condor cluster, where it is finally executed. A multi-job may be executed using a number of different WS-GRAM installations and Condor clusters, which may span across different administrative domains. The results of the job will be retrieved from the WS-GRAM installation and transferred with RFT to a predefined storage server.

Finally, the end-user browses and downloads the result files using suitable workflow components.

It is worthwhile mentioning that the Resource Broker is able to ‘ship’ data mining applications to targeted machines on the grid. Whether applications are ‘shipped’ around the grid is determined manually or automatically. This feature greatly adds to the flexibility of the system since no pre-installation of applications is required, and supports an important use-case scenario: shipping algorithms to data as opposed to shipping data to algorithms. This is extremely useful in many application scenarios we have investigated, for example, if data cannot be transferred because of its inherently distributed nature, its large volume, or for privacy issues.

3 Evaluation

We set up a comprehensive test bed spanning three European countries: United Kingdom, Germany and Slovenia. Each site provided at least one server with a GT4 installation (two in Germany) and heterogeneous Condor compute clusters, comprising up to 90, 5 and 40 available nodes, respectively, typically with CPU speeds of 1.4 to 3GHz, RAM of 521MB to 2GB, and containing both Windows and Linux machines, and in Ulster a 64 node SGI Altix machine with CPU speed of 900 MHz and 128GB shared RAM. One machine in Germany centrally hosted the

Resource Broker and the Information Services. We validated the system by performing comprehensive data mining studies based on our main use case scenarios. These applications involved:

Topology discovery for gene regulatory networks, which is a CPU intensive application based on a genetic algorithm that is implemented in Java. It was capable of using every available compute node in the testbed and so considerable gains in performance were achieved.

Analysis of protein folding simulations, on the other hand, is a data intensive application (data were in the range of 10GB to 2TB), and involved data mining distributed data warehouses of molecular simulations [14]. Data was processed *in situ* by shipping algorithms implemented in different languages to the data resources. Removing the need to download large data volumes was important for this application, as was the need to quickly grid-enable new analysis algorithms.

In the medical application scenario, distributed databases were integrated and queried as if they were a single resource by using Data Services. Then, cross-validation and classification studies were performed using an algorithm from the Weka data mining system. This resulted in a significant increase in productivity, as medical specialists were now able to easily perform data analyses.

The scenario of customer relationship and quality management in the automotive industry involved distributed analysis of millions of documents by shipping pre-processing algorithms to the document repositories. Proprietary programs for document classification and ontology learning were grid-enabled and combined in complex workflow applications.

The ecosystem modeling scenario involved a compute intensive application based on an equation discovery machine-learning program written in C and Python, and restricted to run under Linux. Due to these requirements. Only 18 of the available nodes in the test bed could execute this application, yet the ecologists who were not familiar with grid technology reported that their applications returned results on average more than 6 times faster. They also appreciated the ability to construct complex workflows as this allowed them to easily execute complex and error-prone applications and re-use such applications, which were previously done manually, each and every time.

Here we give a more detailed explanation of a text-mining application concerned with classification. Typical text mining tasks applied to large and fast-evolving text corpora include text classification facilitating, for example, document redirection to suitable subject matter experts, discovering emerging new subjects, and extracting

keywords. In this text classification case study, we performed measurements of cross-validation run-times on newswire data available from the Deutsche Presse-Agentur (dpa).

The experiments were performed on 1 000 documents of the dpa collection of October 2004 (the source XML file is 39 MB in size) using grid-enabled Java applications. Each 10 fold cross-validation run first included a document pre-processing stage, for example to remove stop words, digits and punctuation, and to convert the document collection to a binary format. In each cross-validation experiment, parameter sweeps were performed on the following: the importance weight for corpus terms, the term significance threshold and the text category to be used for classification. Jobs with different settings for these three parameters run in parallel and each job splits the data into 10 folds and performs training and classification 10 times.

In order to gather information about the performance we ran the experiments identically on a single machine, sequentially, and compared these results with parallel runs on the three different GRAM machines with Condor pools in the test bed. This led to the following results: Speed-up depended on the input data location within each site because the algorithm is data intensive and required constant file reading. For a central file server with NFS and Gigabit Ethernet, we measured a linear speed-up only for up to 5 machines. For all the machines in the test bed, linear speed-up could be maintained if input data were mirrored on local disks within each site by specifying the proper Condor setting. In case of local data, parallel executions had a quasi-linear speed-up of run-time due to grid overheads concerned with cluster configurations, file transfer and scheduling

4 Related Works

Recently, various systems and approaches to grid-based data mining have been reported in the literature. Some of those, that are particularly relevant to the DataMiningGrid system, are briefly reviewed here.

GridMiner [10] is designed to support data mining and *online-analytical processing* in distributed computing environments. It provides sophisticated distributed data warehousing functionality. GridMiner implements some common data mining algorithms, including parallel versions, and it also supports various text mining tasks. One of the major differences between GridMiner and DataMiningGrid is that the latter complies with the recent trend towards WSRF.

Knowledge Grid (K-Grid) [11] is a service-oriented system providing grid-based data mining tools and services. The K-Grid system can be used for wide ranging data mining and related tasks such as data management and knowledge representation. Originally, K-Grid was not designed to support OGSA and WSRF. Current developments are geared towards adding this support. Also, as of the time of writing, K-Grid is not available as open source. Whereas the development of the DataMiningGrid was driven by the requirements of a diverse set of applications, a more conceptual view of the knowledge-discovery process has driven the design of K-Grid.

Anteater [12] is a service-oriented architecture for data mining which relies on Web services to achieve extensibility and interoperability and is freely available. However, unlike the other systems, it does not support grid standards such as WSRF or OGSA. Anteater requires data mining applications to be converted into a *filter-stream* structure. While this feature greatly increases scalability, the overhead involved is likely to limit the number of applications that will actually be ported to this platform.

The DataMiningGrid differs in comparison to these other systems, due to:

User-friendliness: It supports various different user interfaces, according to technological capabilities of its users, and the installation process has been kept as simple as possible.

Extensibility: the Application Description Schema allows existing data mining applications to be quickly grid-enabled, and this is further supported by the workflow editor, which dynamically configures its user interface according to the parameters in the instantiated schema.

Parameter sweeps: the Application Description Schema allows the definition of multi-jobs that iterate over application parameters, or input files or directories.

Diverse application support: A wide variety of applications, from different domains, have been implemented and tested. No modification to the data mining application source code has been necessary.

Standardization: the WSRF standard is relatively new, and not yet supported by the Gridminer and Anteater systems.

Open-source: the Gridminer and K-grid system are not freely available.

Conclusions

It seems obvious that emerging large-scale data mining applications shall rely increasingly on distributed computing environments. To tackle these and other issues we developed the DataMiningGrid system and its main features

include high performance, scalability, flexibility, ease of use, conceptual simplicity, compliance with emerging grid and data mining standards, and the use of mainstream grid and open technology. The developed DataMiningGrid software is freely available under the Apache Open Source License V2.0 via SourceForge.net, including supporting documentation.

Acknowledgments

This work was supported by the European Commission FP6 grant DataMiningGrid, Contract No. 004475.

Biographies

The past and current research interests of the authors are as follows.

Vlado Stankovski - application of data mining techniques to engineering and medical problems.

Dr Martin Swain - biophysics, systems biology, data management and grid computing.

Valentin Kravtsov - grid technology and distributed and parallel computing.

Thomas Niessen - grid computing, data mining and databases.

Dennis Wegener - data mining and grid computing.

Matthias Röhm - distributed text-mining.

Jernej Trnkoczy - federated digital libraries.

Dr Michael May is Head of the Knowledge Discovery Department at the Fraunhofer Institute for Intelligent Analysis and Information Systems. His research interests are in data mining and knowledge discovery.

Dr Jürgen Franke is Head of the text-mining team at DaimlerChrysler. His research interests are in text-mining and character recognition.

Professor Assaf Schuster is Head of the Distributed Systems Laboratory at the Technion. His research interests are in distributed and parallel computing.

Professor Werner Dubitzky holds a Chair in Bioinformatics and is Head of the Systems Biology Research Group at the University of Ulster. His research interests include bioinformatics, systems biology, data and text mining, artificial intelligence and grid technology.

Relevant Uniform Resource Locators

URL	Remark
http://www.DataMiningGrid.org	The DataMiningGrid Project Web site
http://www.crisp-dm.org	CRoss Industry Standard Process for Data Mining
http://www.globus.org/ogsa/	Open Grid Services Architecture
http://www.oasis-open.org/	OASIS Web site: Hosts the OASIS Web Services Resource Framework (WSRF) TC

References

- [1] Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, 2001, pp. 200--222.
- [2] Kumar, M.M. Kantardzic, and S. Madden, "Guest Editors' Introduction: Distributed Data Mining--Framework and Implementations," *IEEE Internet Computing*, vol. 10, no. 4, 2006, pp. 15--17.
- [3] J. Nabrzyski, J.M. Schopf, and J. Węglarz (editors), "Grid Resource Management: State of the Art and Future Trends," *Kluwer Academic Publishers*, Boston 2004.
- [4] P. Plaszczak and Jr. R. Wellner, "Grid Computing: The Savvy Manager's Guide," *Morgan Kaufmann*, Amsterdam 2006.
- [5] Sotomayor and L. Childers, "Globus Toolkit 4: Programming Java Services," *Morgan Kaufmann*, Amsterdam 2006.
- [6] M. Antonioletti et al., "The design and implementation of Grid database services in OGSA-DAI," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2-4, 2005, pp. 357--376.
- [7] G. Von Laszewski et al., "A Java commodity grid kit," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, 2001, pp. 645--662.
- [8] S. Venugopal, R. Buyya and L. Winton, "A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids," *Concurrency and Computation: Practice and Experience*, vol.18, no 6, May 2006, pp. 685-69.
- [9] G. Churches et al., "Programming scientific and distributed workflow with Triana services", *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, 2005, pp. 1021--1037.

- [10] P. Brezany, I. Janciak, and A.M. Tjoa, "GridMiner: A Fundamental Infrastructure for Building Intelligent Grid Systems," *Proc. 2005 IEEE/WIC/ACM Int'l Conf. Web Intelligence (WI'05)*, 2005, pp. 150--156.
- [11] Congiusta, D. Talia, and P. Trunfio, "Distributed data mining services leveraging WSRF," *Future Generation Computer Systems*, vol. 23, no. 1, 2007, pp. 34--41.
- [12] D. Guedes, W.Jr. Meira and R. Ferreira, "Anteater: A Service-Oriented Architecture for High-Performance Data Mining", *IEEE Internet Computing*, 2006, pp. 36--43.
- [13] C. Silva et al., " P-found: The Protein Folding and Unfolding Simulation Repository" Proc. 2006 IEEE Symp. Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'06), pp.101-108, Toronto.
- [14] Stankovski et al., "Grid-enabling data mining applications with the DataMiningGrid: An architectural perspective" *Future Generation Computing Systems*, In press.