BUILDING SCALABLE DATA MINING GRID APPLICATIONS
An Application Description Schema and Associated Grid Services

Vlado Stankovski
Faculty of Civil and Geodetic Engineering, University of Ljubljana, Jamova cesta 2, Ljubljana, Slovenia

Dennis Wegener
Fraunhofer Institute for Intelligent Analysis and Information Systems, Sankt Augustin, Germany

Keywords: Grid, distributed applications, data mining, middleware.

Abstract: Grid-enabling existing stand-alone data mining programs, data and other resources, such as computational servers, is motivated by the possibility for their sharing via local and wide area networks. Expected benefits are improved effectiveness, efficiency, wider access and better use of existing resources. In this paper, the problem of how to grid enable a variety of existing data mining programs, is investigated. The presented solution is a simple procedure, which was developed under the DataMiningGrid project. The actual data mining program, which is a batch-style executable, is uploaded on a grid server and an XML document that describes the program is prepared and registered with the underlying grid information services. The XML document conforms to an Application Description Schema, and is used to facilitate discovery and execution of the program in the grid environment. Over 20 stand-alone data mining programs have already been grid enabled by using the DataMiningGrid system. By using Triana, a workflow editor and manager which represents the end-user interface to the grid infrastructure, it is possible to combine grid enabled data mining programs and data into complex data mining applications. Grid-enabled resource sharing may facilitate novel, scalable, distributed data mining applications, which have not been possible before.

1 INTRODUCTION

Data mining in grid computing environments is motivated by resource sharing via local and wide area networks (Stankovski et al 2008a, 2008b). Increased performance, scalability, access and resource exploitation are the expected key benefits. Furthermore novel distributed data mining applications may facilitate the automated extraction of potentially useful information from increasingly large, geographically distributed data volumes.

However, grid-enabling large-scale data mining applications is difficult to achieve due to a number of factors. Grid computing itself is a novel field of research and relevant standards and technologies are still evolving (Foster, Kesselman, Tuecke, 2001; Plaszczak and Wellner, 2006; Sotomayor and Childers, 2006; Antonioletti et al., 2005). Moreover, there exists a plethora of data mining technologies and a staggering number of largely varying data mining application scenarios (Kumar, Kantardzic and Madden, 2006; Guedes, Meira and Ferreira, 2006; Stankovski and Dubitzky, 2007; Conguista, Talia and Trunfio, 2007). 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 presented elsewhere (Stankovski et al, 2008a, 2008b) as well as the actual applications (e.g. Trnkoczy and Stankovski, 2008). We have designed and implemented a workflow-oriented, scalable, high performance computing system that supports emerging grid interoperability standards and technology. The system itself is freely available under the Apache Open Source License V2.0 via

Stankovski V. and Wegener D. (2008).
BUILDING SCALABLE DATA MINING GRID APPLICATIONS - An Application Description Schema and Associated Grid Services.
In *Proceedings of the Third International Conference on Software and Data Technologies - PL/DPS/KE*, pages 221-228
DOI: 10.5220/0001891302210228

SourceForge.net, including all supporting documentation.

In the present study, we investigate the problem of how existing, stand-alone data mining applications can be grid-enabled and subsequently executed on a grid service infrastructure.

Our final goal is to enable users from various disciplines to build and utilize complex, scalable data mining applications. To that end, an effective mechanism, which was developed under the DataMiningGrid project, is presented in this paper.

2 GRID RESOURCES

In a grid environment, it is possible to exploit numerous, potentially unlimited resources, such as data, data mining applications, CPUs, storage, networks and clusters.

Given the nature of data mining applications, a variety of computational resources that may be shared were identified:

- **Data.** The data to be mined, which may exist in the form of relational databases, data files (documents in various formats) and directories consisting of collections of documents;
- **Programs.** Data mining programs providing the implementation of data mining algorithms used to mine data. Seen from the grid resource viewpoint, a data mining program, application or algorithm is *an executable* with associated input data, output data and parameter settings. The executable can be anything starting from a Java, C, Python or a BashShell program.
- **Computational Machines.** Computational machines providing raw computing power to run the data mining program and process the data. Important parameters about computational machines are speed, occupancy, memory which can be used during processing, architecture and so on;
- **Storage.** Data storage devices to physically store the input and output data of data mining applications. For storage devices it is mainly important to have the ability to reserve space in advance, and to have safe and fast mechanisms for storing and retrieving data. Should storage be used for storing relational databases, than the necessary server-side software is also essential to be implemented on the actual site;
- **Streaming Devices.** Sensors and other devices streaming data in a network are special kind of resources. These, however, are currently not (directly) supported by the DataMiningGrid technology; and
- **Networks.** Optimization of network parameters should also be considered for time-critical applications.

It is important to realize that certain resources from the list above can easily be moved in the network (notably data, programs and associated libraries) and existing transfer protocols could be used for that purpose (e.g. ftp, GridFTP or RFT), while other resources can not be moved in the network (e.g., computational machines). All resources, however, have different parameters that should be considered when developing a distributed, grid-based data mining application.

We have investigated several large-scale data mining scenarios in which it is impractical or impossible to move the data in the network due to variety of reasons, e.g., the large amount of data or security restrictions. In such cases, it must be made possible to move the data mining programs to the data rather then the data to the data mining programs.

3 GRID SERVICES

Grid middleware services represent virtualization of grid resources. In other words, grid resources can only be accessed by using grid services, while local resources have to be grid-enabled before they are actually shared.

3.1 Virtualization

In the past years, considerable research and development effort has been put towards the development of middleware services and tools targeting some of the access and composition obstacles to large-scale resource sharing and exploitation. The key benefit of grid services is that they provide an effective and popular way of abstracting the complexity of distributed data and computational resources and also represent a variety of utility services. DataMiningGrid represents a platform that enables the sharing of grid resources between applications which facilitates reuse, embedding, modification or extension of an application's content to enable a far more rapid development cycle than what was previously possible with conventionality programming methodologies.

In particular, users are not concerned either with the technical details associated with constructing grid services, or the technical details of the underlying grid and Web service infrastructures, but are interested in exploring their data. To that end, DataMiningGrid provides workflow-editing software components that are integrated with the existing Triana workflow editor and manager. By using the workflow editing components it is possible to develop applications which are tailored to the specific needs of the end-users. Moreover, interoperability between services within a distributed system is enabled by using the OGF's WSRF standard, so as to liberate users to use the grid services regardless of their own systems.

3.2 Workflows

Workflows are effectively declaratively defined as coordinated "plumbing" between services, executed as steps. Triana is an advanced system for editing and managing workflows, which is used as a user interface to Web services (Churches at al., 2005) and recently also as a user interface to grid services (Stankovski et al., 2008a). It may be used to link efficiently and effectively data resources, analytical tools and computing processes together in a form of various domain-specific graphs representing information flows. It is complete and self-defined, and represents a far more effective mean of sharing knowledge, processing, communication, storage or content than the more elementary building blocks that represent the individual services.

As more data is produced by users every day and more utility services are provided online, coupled with 'time-to-market' pressures, the complexity and dynamicity of information flows in the past years have been increasing dramatically. It is therefore necessary in many scientific and business communities, and will be even more so in the near future, to discover, extract and share the knowledge contained in complex workflows.

3.3 Grid Middleware and Services

Different grid middleware solutions exist and continue to be developed over the past years. Here, we provide a short overview of ready-to-use grid, WSRF-compliant services, which were used to build a grid service infrastructure (a test-bed). These include the Globus Toolkit 4, Condor and DataMiningGrid high-level services.

3.3.1 Globus Toolkit 4 Middleware and Services

One of the first grid middleware toolkits implementing the Web Services Resource Framework (WSRF) v. 1.2 specification, a specification promoted by the Organization for the Advancement of Structured Information Standards (OASIS), is the Globus Toolkit 4 (GT4) (Foster, Kesselman and Tuecke, 2001). GT4 provides a range of grid services that can be directly used to build a distributed grid environment. These include data management, job execution management, community authorization services etc. All these services can be used to build custom grid applications, and are elaborated in detail elsewhere (Sotomayor and Childers, 2006). Besides these ready-to-use services, the GT4 provides an Application Programming Interface (API) that allows for development of proprietary WSRF-compliant services. Due to the reasons listed above the GT4 was used as grid middleware in this study.

Following is a short overview of relevant ready-to-use grid services from the GT4 toolkit.

- **The Web Service - Grid Resource Allocation and Management** (WS-GRAM) provides all basic mechanisms required for execution management, i.e., initiation, monitoring, management, scheduling, and coordination of remote computations.
- **Data Management Services**, such as GridFTP and Reliable File Transfer (RFT). These data services are mainly used for transfer and management of distributed, file based data, including program executables and their software libraries. GridFTP is used, e.g., to transfer executables and required libraries to the selected computational server in the grid.
- **Information Services** are used to discover, characterize and monitor resources, services and computation. The GT4's Monitoring and Discovery System 4 (MDS4) provides information about the available grid resources and their status. It has the ability to collect and store information from multiple, distributed information sources. This information is used to monitor (e.g., to track usage) and discover (e.g., to assign computing jobs and other tasks) the current state of services and resources in a grid system. The DataMiningGrid high-level services (in particular the Resource Broker and Information Services) are using the MDS4 service.

In our test bed, the following GT4 services are used extensively: WS-GRAM, GridFTP and MDS4.

3.3.2 Condor

Scheduling of grid jobs in local computing clusters is achieved by using the Condor middleware. Condor is specialized workload management software for submitting compute-intensive jobs to local computational clusters, which has been described in detail elsewhere (Thain, Tannenbaum and Livny, 2005). In our application, the GT4 submits a subset of parallel jobs to appropriate Condor clusters, and it is up to the Condor software to place them into a local queue, choose when and where in the local cluster to run the jobs, monitor the progress of the jobs, and ultimately inform GT4 services upon their completion.

3.3.3 DataMiningGrid High-Level Services

In addition to the core grid services provided by GT4, other high-level WSRF compliant, ready-to-use grid services have recently been developed under the DataMiningGrid project. Here, we provide a brief overview of the Resource Broker and the Information Integrator Service. These services fully support the parallel execution of a variety of data mining (batch-style) programs in the grid environment.

- **The Resource Broker Service** is responsible for the execution of data mining programs anywhere in the grid environment. It provides a matching between the request for data mining program execution, which is also called a *job* in grid terminology, and the available computational and data resources in the grid. It takes as input the computational requirements of the job (CPU power, memory, disk space etc.) and data requirements of the job (data size, data transfer speed, data location etc.) and selects the most appropriate execution machine for that particular job. The job is passed on to the WS-GRAM service and executed either on an underlying Condor cluster or by using the GT4's Fork mechanism. The Resource Broker service is capable of job delegation to resources spanning over multiple administrative domains. The execution machines are automatically selected so that the inherent complexity of the underlying infrastructure is hidden from the users. The Resource Broker service performs the orchestration of automatic data and application transfers between the grid nodes, using the GridFTP component of GT4 for the transfers. The Resource Broker is designed to execute *multi-jobs*. Multi-jobs are collections of single jobs that are bound for parallel execution. In DataMiningGrid, a multi-job usually consists of a single data mining program, which is instantiated with different input parameters and/or different input data sets. The individual jobs are then executed in parallel on various computational servers in the grid environment. Each job, therefore, represents one execution of a data mining program (i.e., an executable) with specific input parameters and data inputs. The Resource Broker makes extensive use of the associated Information Integrator service.
- **The Information Integrator Service** provided by DataMiningGrid operates in connection to the MDS4 service provided by GT4. The Information Integrator service is designed to feed into other grid components and services, including services for discovery, replication, scheduling, troubleshooting, application adaptation, and so on. Its key role is to create and maintain a register of grid-enabled data mining programs. By doing so, it facilitates the discovery of grid-enabled programs on the grid, and their later use through the Resource Broker service.

4 AN APPLICATION DESCRIPTION SCHEMA

A system whose main function is to facilitate the sharing of grid resources within a grid environment and supporting the development of distributed, scalable data mining applications has to take into account the unique constraints and requirements of data mining programs with respect to the data management, their execution requirements, and so on.

In order to cope with the complexity of the dynamically changing grid environment, an Application Description Schema (ADS), which is a novel metadata model in form of an XML schema, was developed. The ADS defines properties necessary to describe data mining programs and other grid resources that may be shared in distributed grid applications in a uniform way. For example, the XML schema provides properties to describe the data mining program and all its input data, output data, parameter settings etc, so it can be used to describe any program (i.e., executable) in general.

The ADS consists of two parts, which are:

- **A common part**, which can be used to describe any program, i.e., not necessarily data mining programs; and
- **A data mining (domain-specific) part** containing additional information relating to data mining programs, e.g., the program's application domain(s), the name of the atomic algorithm, the problem solving technique etc.

The common part of the ADS is subdivided into:

- **General** part that contains definitions of properties of the program, such as a unique ID, the program's name, vendor etc. This information is used to build a grid wide registry of available programs;
- **Execution** part contains definitions of properties related to the execution of the program. This includes the application type (Java, C, Bash Shell or Python), the location of the executable in the grid, list of libraries required for execution etc.
- **Application** part provides definitions of properties needed to use the program. This includes options, data inputs and outputs, parameter lists and loops, requirements, environment variables etc.

The ADS adds a great range of functionality to the system. It is used through the whole execution process in the grid environment, beginning from the registration of data mining and other programs through program discovery, selection, parameter, and input data specification to the actual execution of the program on the grid.

5 GRID ENABLING DATA MINING PROGRAMS

5.1 Batch-Style Programs

A data mining program is grid enabled when it is made available in the grid environment so that the grid users can actually share it and make use of it, hence, the program may be considered a grid resource. To cope with interoperability and other high-level aspects it is necessary to provide an extensive description of each data mining program by means of the ADS.

From an infrastructural viewpoint, all executables and their associated libraries represent files. In DataMiningGrid, any data mining program that can be invoked from command-line (and is implemented to run without a graphical user interface), can be grid enabled. A predominant variety of general and domain specific programs can be adapted to be invoked from a command-line. For example, input data, output data, and parameter settings can be presented to a program via the command-line, using a specific format, e.g. a flag followed by the associated value (‘`flag <space> value`’, e.g., ‘`-number 77`’). Additionally the program may have some system requirements like minimum free disk space, minimum memory or required operating system, which have to be specified in order to later run the program on appropriate computational machine(s) in the grid.

Execution of data mining algorithms in the grid is based always on a valid description of the prerequisites the algorithm needs. The ADS was developed to describe the algorithms including all their parameters, input/output data, necessary environment prerequisites etc.

5.2 Grid-Enabler (Web) Application

By using the ADS schema, the developers can create a very detailed description of their program, which guarantees that it will run successfully and on the other hand provides information for its discovery on the grid. Providing the description will always rely on the developer of the grid application, someone who may not be acquainted with the underlying grid system. The presented solution is a very simple procedure: the actual data mining program (i.e., executable) is uploaded on a grid server and an ADS instance that describes the program is prepared and registered with the underlying Information Integrator service.

To speed-up the grid-enabling process, the DataMiningGrid project developed a Web application which consists of several form-based jsp web pages, leading the user through the whole process of creating and uploading his data mining program. The following Web pages are provided:

- General information
- Execution information
- Input data specification
- Output data specification
- Requirements specification
- Executable and libraries upload

5.3 Life-Cycle of the ADS Instance

The ADS instance file contains all invariant properties of the respective data mining program (e.g., system architecture, location of the executable and libraries, programming language). These attributes cannot be altered by users of the system, but are typically specified by the developer of the program during the process of publishing the program on the grid. The ADS instance also includes default values for all options, but the exact values are not set.

When querying for a data mining program, the client side components (implemented as Triana units) use the ADS instance in order to dynamically create a GUI, which conforms to the description of that particular data mining program. For example, for each option a form field is generated, where the user can specify the values for that option. At this stage the user provides the exact values for the applications parameters (during runtime, e.g., application parameter values, data input, additional requirements) of the program.

A fully specified ADS instance represents a multi-job description and is submitted to the Resource Broker for parallel execution in the grid.

The Resource Broker uses the information contained in the ADS instance to aggregate appropriate resources. Particularly useful are the following information:

- **Static Resource Requirements.** regarding system architecture and operating system. Applications implemented in a hardware-dependent language (e.g., C) typically run only on the system architecture and operating system they have been compiled for (e.g., PowerPC or Intel Itanium running Linux). For this reason, the Resource Broker has to select execution machines that offer the same system architecture and operating system as required by the application.
- **Modifiable Resource Requirements.** memory and disk space. While data mining applications may require a minimal amount of memory and disk space at start-up time, memory and disk space demands typically rise with the amount of data being processed and with the solution space being explored. Therefore, end users are allowed to specify these requirements in accordance with the data volume to be processed and their knowledge of the application's behaviour. The Resource Broker will take into account these user-defined requirements and match them to those machines and resources that meet them.
- **Modifiable Requirements.** identity of machines. In some cases end users may generally wish to limit the list of possible execution machines based on personal preferences, for instance, when processing sensitive data. To support this requirement, it is possible for the user to specify the IPs of such machines in the job description. Such a list causes the Resource Broker to match only those resources and machines listed and to ignore all other machines independent of their capabilities.
- **The Total Number of Jobs.** Instead of specifying single values for each option and data input that the selected application requires, it is also possible to declare a list of distinct values (e.g., true, false) or a loop (e.g., from 0.50 to 10.00 with step 0.25). These represent rules for variable instantiations, which are translated into a number of jobs with different parameters by the Resource Broker. This is referred to as a multi-job. As a result, the Broker will prefer computational resources that are capable of executing the whole list of jobs at once in order to minimize data transfer. Typically, such resources are either clusters or high-performance machines offering many distinct processors. As an example, if the user specifies two input files (a.txt, b.txt) for the same data input and two loops running from 1 to 10 with step 1 as parameters for two options, the Resource Broker will translate this into 200 (2 x 10 x 10) distinct jobs. If no singe resource capable of executing them at once is available, the Broker will distribute these jobs over those resources that provide the highest capability.

In addition, the Resource Broker evaluates further information from the job description that becomes important at the multi-job submission stage. This information is briefly described below:

- **Instructions.** on where the program executables are stored, including all required libraries, and how to start the selected program. These are required for transferring executables and associated libraries to execution machines across the grid, which is part of the stage-in process. By staging-in programs together with the input data dynamically at run-time, the system is capable of executing these applications on any suitable machine in the grid without prior installation of the respective data mining program.
- **All Data Inputs and Data Outputs.** that have to be transferred prior the execution.
- **All Option Values (Data Mining Program Parameters).** that have to be passed to the

program at start-up. As the Resource Broker is capable of scheduling executables that are started in batch-mode from a command line, it passes all option values as flag-value pairs. Here, each flag is fixed and represents a single option. The values, however, may change for each call if a multi-job is specified.

6 GRID ENVIRONMENT

From this point forward the data mining program is ready to be used in the grid execution environment.

6.1 Test Bed

The DataMiningGrid test bed was developed on the bases of the grid middleware and ready-to-use grid services discussed in the previous sections. It is a grid service infrastructure, with services running at various sites across different administrative domains in three European countries (Ireland, Slovenia and Germany).

The test bed provides a number of capabilities, the most important being the following:

- **The ability to execute a variety of batch-style programs**, **at any appropriate computational server in the grid.** Over 20 grid-enabled programs are currently stored in executable repositories on various grid servers in the test bed. These programs may be combined in complex-workflows, containing several multi-jobs executed sequentially;
- **Meta-scheduling,** i.e., dynamic and automatic allocation of optimal computational servers in the grid environment is achieved through the use of the Resource Broker, the Information Integrator service and MDS4.
- **Program and data movement across different administrative domains** is achieved through the use of the GridFTP and RFT services.

In addition to these, grid environments based on GT4 and DataMiningGrid high-level services have a number of other capabilities, such as a Grid Security Infrastructure, which is described in Stankovski et al. (2008a, 2008b). Over 20 grid-enabled data mining programs were already combined into large scale data mining applications. To demonstrate the flexibility and extensibility of the developed software, we have developed applications for various research and industrial sectors, such as bioinformatics, industry (text-mining), medicine, open publishing and digital libraries, civil engineering. For example, grid-enabled Federated Digital Libraries have been described by Trnkoczy, Turk, and Stankovski (2006), and Trnkoczy and Stankovski (2008).

7 CONCLUSIONS

It seems obvious that emerging large-scale data mining applications shall rely increasingly on distributed computing environments.

Many data mining programs require a repeated execution of the same process with different parameters that usually control the behaviour of the implemented data mining algorithm or different input data sets. This is typically required in optimization or sensitivity analysis tasks. Hence, properties like performance, scalability, usability and security are critical for this kind of applications. The DataMiningGrid system was developed to address these requirements. More details about the results of this project can be found at its web site (DataMiningGrid). The developed software is distinguished by its flexibility, ease of use, conceptual simplicity, compliance with emerging grid and data mining standards, and the use of mainstream grid and open technology.

The DataMiningGrid project developed a coherent framework, which offers data miners, who are usually not grid experts, the ability to easily grid enable existing, stand-alone data mining programs, construct complex data mining tasks, which are represented by complex Triana workflows and execute these workflows in a grid environment. A case study on the extensibility of the DataMiningGrid platform is given in the literature (Wegener and May, 2007).

The developed DataMiningGrid software is freely available under the Apache Open Source License V2.0 via SourceForge.net, including all supporting documentation.

Despite its promise, however, there are still a lot of issues to be resolved before grid technology is commonly applied to large-scale data mining tasks. (Stankovski and Dubitzky, 2007).

ACKNOWLEDGEMENTS

This work was supported by the European Commission FP6 grant DataMiningGrid, http://www.dataminingrid.org, contract no. 4475. The collaboration of all project partners is

acknowledged. They have jointly participated in developing the system.

REFERENCES

Antonioletti, M., et al., 2005. “The design and implementation of Grid database services in OGSA-DAI,” “Concurrency and Computation: Practice and Experience,” vol. 17, no. 2-4, pp. 357--376.

Churches, G., et al., 2005. “Programming scientific and distributed workflow with Triana services”, Concurrency and Computation: Practice and Experience, vol. 18, no. 10, pp. 1021--1037.

Congiusta, D., Talia, D., and Trunfio, P. 2007. “Distributed data mining services leveraging WSRF,” Future Generation Computer Systems, vol. 23, no. 1, pp. 34--41.

DataMiningGrid. 2006. Data Mining in Grid Computing Environments, EU contract no. 4475, http://www.dataminninggrid.org

Foster, I., Kesselman, C. and Tuecke, S., 2001. “The Anatomy of the Grid: Enabling Scalable Virtual Organizations,” International Journal of High Performance Computing Applications, vol. 15, no. 3, pp. 200--222.

Guedes, D., Meira, W.Jr., and Ferreira, R., 2006. “Anteater: A Service-Oriented Architecture for High-Performance Data Mining”, IEEE Internet Computing, pp. 36--43.

Kumar, M., Kantardzic, M. and Madden, S., 2006. “Guest Editors' Introduction: Distributed Data Mining--Framework and Implementations,” IEEE Internet Computing, vol. 10, no. 4, pp. 15--17.

Nabrzyski, J., Schopf, M., and Węglarz, J., 2004. (editors), “Grid Resource Management: State of the Art and Future Trends,” Kluwer Academic Publishers, Boston.

Plaszczak, P. and Wellner, Jr. R., 2006. “Grid Computing: The Savvy Manager’s Guide,” Moragan Kaufmann, Amsterdam.

Sotomayor, B., and Childers, L., 2006. “Globus Toolkit 4: Programming Java Services,” Moragan Kaufmann, Amsterdam.

Stankovski et al., “Grid-enabling data mining applications with DataMiningGrid: An architectural perspective”, Future Generation Computing Systems, vol. 24, no. 4, pp. 259--279.

Stankovski et al., 2008b. “Digging Deep in the Data Mine with DataMiningGrid”, IEEE Internet Computing, in press.

Stankovski, V. and Dubitzky, W. 2007. “Special Section: Data Mining in Grid Computing Environments”, Future Generation Computer Systems, vol. 23, no. 1, pp.

Thain, D., Tannenbaum, T. and Livny, M., 2005. Distributed computing in practice: The Condor Experience, Concurrency-Practice and Experience, vol. 17, pp. 323—356.

Trnkoczy, J. and Stankovski, V. 2008. “Improving the performance of Federated Digital Library services” Future Generation Computer Systems, in press, doi:10.1016/j.future.2008.04.007.

Trnkoczy, J., Turk, Ž. and Stankovski, V., 2006. “A Grid-based Architecture for Personalized Federation of Digital Libraries,” Library Collections, Acquisitions, and Technical Services, vol. 30, pp. 139--153.

Venugopal, S., Buyya, R., and Winton, L., 2006. “A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids,” Concurrency and Computation: Practice and Experience, vol.18, no 6, pp. 685-69.

Wegener, D. and May, M. 2007. “Extensibility of Grid-Enabled Data Mining Platforms: A Case Study” In Proc. of the 5th International Workshop on Data Mining Standards, Services and Platforms, pp 13-22, San Jose, California, USA, August, 2007. ISBN 978-1-59593-838-1.