A Distributed Service Architecture for end-to-end Data Intensive Analysis

Jang-uk In, Laukik Chitnis, Richard Cavanaugh, Pradeep Padala,
Sanjay Ranka, Mandar Kulkarni, Paul Avery
University of Florida
{juin, lchitnis, ppadala, ranka, md2}@cise.ufl.edu, {cavanaug, avery}@phys.ufl.edu

Abstract

Grid computing has become a popular way of providing high performance for data intensive scientific applications. Many interesting and challenging problems related to data intensive computing have been solved recently using various grid services. However, reliable and scalable software architecture for solving general-purpose distributed data intensive problems is missing. We developed a software architecture that combines existing grid services with our state-of-the-art grid scheduler, Sphinx. We deployed our prototype across the USCMS Grid3 [Grid3]. It is in the primary stages of exhibiting interactive remote data access, demonstrating interactive workflow generation and collaborative data analysis using virtual data and data provenance, as well as showing non-trivial examples of policy based scheduling of requests in a resource constrained grid environment. Here we focus on the design of infrastructure that handles tasks from the generation of abstract workflows to analysis of results. We present our experiments in scheduling various workflows using different algorithms in Sphinx.

1. Introduction

The Compact Muon Solenoid (CMS) detector [CMS04] is a multi-purpose, high-energy physics experiment at the Large Hadron Collider (LHC) that is located at the European Centre for Particle Physics (CERN) near Geneva, Switzerland. When it is completed in 2007, it will record data at a rate of hundreds of mega-bytes per second leading to data stores in the tens of peta-bytes by 2010 and exa-bytes by 2015. Event samples in such massive data stores will need to be rapidly accessed and analyzed by globally dispersed scientists in hundreds of collaborative teams.

To support this need, storage and computational resources are currently being hierarchically distributed across a global LHC Computational Grid (LCG) [LCG04]. Nevertheless, efficient scientific analysis of large datasets will require that access to those distributed resources should often occur in un-structured and interactive (but controlled) ways and point towards a distributed services architecture [Newman03]. Such a system requires that request-response latencies should be low, that finely as well as coarsely grained chaotic access to distributed datasets be possible, and that tools supporting collaborative work within a geographically distributed environment be widely available. To date, activities within the grid research communities [Globus04, Condor04, GriPhyN04, EDU04] have provided many key technologies supporting generic access to distributed grid resources. However, they have so far tended to focus on pre-planned workflows that are executed in a batch-oriented fashion and where data is accessed in pre-determined ways with less consideration given to interactive capabilities, chaotic access to data, or quality of service in a resource constrained system.

Here we report on the implementation of a prototype of distributed high-level services supporting grid-enable data analysis within the LHC physics community, and begin to investigate the associated complex behavior of such an end-to-end system. In particular, the prototype integrates several services for the distributed data analysis. The services include a user interface for data analysis, a uniform web-services interface among grid applications, a virtual data service, grid resource management and scheduling, a grid weather monitoring, and a workflow and replica management.

For test purposes, the prototype is deployed across Grid3 and is in the stages of exhibiting interactive remote data access, demonstrating interactive workflow generation and collaborative data analysis using virtual data and data provenance, as well as showing non-trivial examples of policy based scheduling of requests in a resource constrained grid environment.

In section 2, we describe the architecture of the prototype, and each service component in the system is also discussed in the section. In section 3 we present the results that we obtained from experiments of the prototype on a testbed. We conclude the work in section 4.

2. Architecture

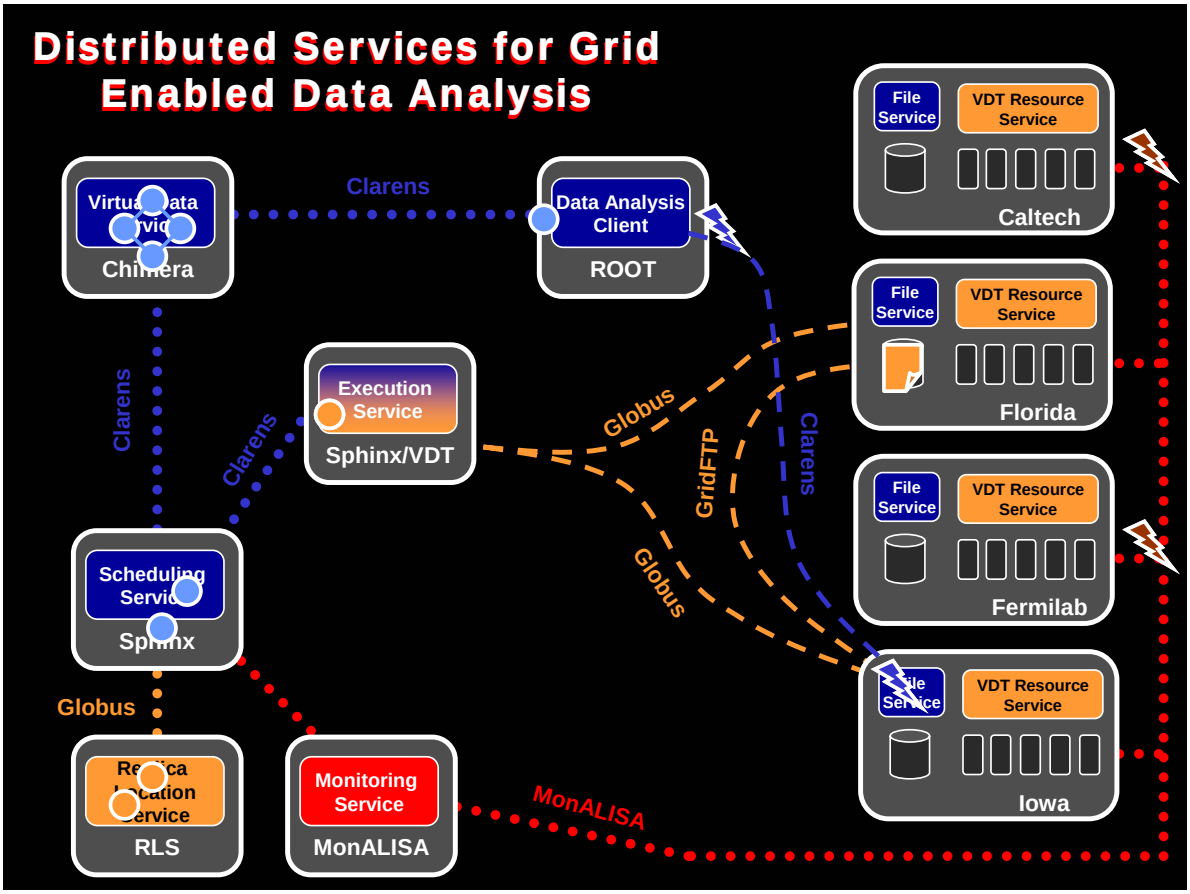

Figure 1: Web-based data analysis system architecture. The system consists of several grid-based services that make distributed data analysis possible. Each of the services is discussed in section 2.2.

A general architecture of the proposed distributed data analysis system is shown in Figure 1. The system supports a graphical user interface (GUI) through which a user can interact with the system to register, generate and explore data. The typical data analysis framework, ROOT provides appropriate GUI and event handling functionalities. Several existing or being developed services incorporates to accomplish the data analysis service. Each service is connected to the others using Clarens that is a web-based communication protocol with grid enabled security infrastructure.

A user starts data analysis by registering new virtual data into the data analysis system. Chimera virtual data system keeps description and generation procedure of the registered data. Users can send requests to generate the virtual data as well as to review the data information in Chimera. After receiving data generation request the virtual data service interacts with a grid resource scheduling service. The scheduling service allocates grid resources to the request. Chimera passes generation procedure of the requested data in the form of abstract Directed Acyclic Graph (DAG). The abstract DAG describes the procedure by specifying the dependency of input and output data among subtasks. Only logical file names are given to the data since their physical locations are not decided yet.

A grid resource scheduling service makes resource allocation decision to data generation request from a user. Specifically, the service decides physical locations and paths for logical file names in an abstract DAG passed from Chimera virtual data system. The data analysis system uses Sphinx scheduling middleware to perform the service. Sphinx works closely with the grid resource monitoring service MonALISA and replica location service RLS to make the scheduling efficient in a dynamically changed grid environment.

Once resource allocation decision for data generation is made, a grid enabled execution service submits tasks to resources in a grid test bed. The execution service observes job dependency in a DAG when it makes the submission. After receiving the execution requests, resources in a grid test bed execute the tasks based on their local scheduling decision. The data analysis system uses Virtual Data Toolkit (VDT) client and server packages to accomplish the architecture.

After finishing the data generation physical file locations and paths of generated data are registered into a replica location service (RLS). A user submits a query for the data locations to RLS, and finally she/he can explore the data. ROOT Clarens client module supports efficient methods for connecting to remote servers and analyzing data on the sites.

2.1 Services

We describe service components that are used to provide distributed system for grid enabling high-energy physics data analysis in this subsection. Each of the services is a key component making the analysis possible.

ROOT: An object-oriented data analysis framework

ROOT [Rademakers98] system provides a set of object-oriented frameworks to analyze large amount of data. ROOT manages hierarchically structured framework architecture. Each of the frameworks supports different services for data analysis. ROOT graphical user interface classes and event generating and handling classes are used to generate interfaces through which user can interact with the distributed services. ROOT is also used to analyze remotely located data with the Clarens client functionalities.

In the service demonstration we use ROOT graphical user interface to analysis remotely located data with the Clarens client functionality described in the next section. The interface allows scientists making queries to register new virtual data derivation into a virtual data catalog, to submit requests generating physical data files and to explore remotely located data files for the analysis purpose.

In Table 1 we show the various services that work in the background to provide the functions in the graphical user interface. We discuss each of the services in more detail from the next section.

Functionalities	Services
Registering virtual data derivation	Chimera virtual data system
Generating physical data	Sphinx grid scheduling service MonALISA grid resource monitoring service VDT grid resource management service
Exploring remote data files	ROOT data analysis service Clarens grid-enabled web service

Table 1: Distributed data analysis functions and services

Clarens: Grid-enabled web services framework

The communication backbone of the distributed service demonstration is Clarens web services framework [Clarens]. Clarens provides a host of grid computing services. Clarens client sends request to the host via the lightweight XML-RPC mechanism using Grid Security Infrastructure (GSI) for authentication [Sandholm].

Clarens supports secure communication between service components in the distributed service. The modules exchange requests and data through Clarens client and server modules. Clarens provides a client module for ROOT data analysis application. The ROOT client module supports an infrastructure to log into a remote Clarens server and execute RPC calls for remote data analysis. Other service components also use Clarens to exchange requests for data registration, generation and exploration. Clarens service modules are implemented in Java Servlets.

Chimera virtual data system

Chimera [Foster03] creates and manages a virtual data catalog that represents data derivation procedures and derived data. The virtual data catalog contains description of a set of virtual data ("transformations") and track all the data produced by data generation ("derivations"). Chimera produces a "recipe" to generate a logical file in the form of an abstract program execution graph.

Our distributed services use Chimera to register data derivation procedures into virtual data catalog, and to generate abstract execution graph for data generation corresponding to user request. Chimera, as a virtual data server, interacts with user in a client side through Clarens communication service.

Sphinx: Grid scheduling service

Sphinx [In03] is a novel grid-scheduling framework for planning requests of high-end computational, storage and network resources that are dynamic with respect to activity and availability. Sphinx administrates grid resource usage policies, and schedules complex and data intensive scientific applications providing a specified quality of service.

In the distributed service demonstration an abstract execution graph generated by Chimera virtual data system is passed to Sphinx scheduling system. Sphinx makes resource allocation decision for the workflow across the grid test-bed. The decision is based on policy and grid weather information as monitored by a grid monitoring system.

MonALISA: Grid resource monitoring service

MonALISA [Newman03] is a distributed monitoring service system using JINI/JAVA and WSDL/SOAP technologies. It provides monitoring information from large and distributed systems to a set of loosely coupled "higher level services" in a flexible, self describing way.

MonALISA provides Sphinx scheduling service with critical grid resource status information in the service demonstration. The information is specific to resource properties such as CPU, bandwidth, queue lengths, storage, etc. The scheduling system can make resource allocation decision based on the monitoring information and resource usage policy information in a dynamically changed grid weather environment.

The MonALISA monitoring service is deployed at each of the participating sites. A MonALISA repository is maintained at the grid-scheduling site, which had the snapshot of the required parameters (monitored resources). The scheduling engine queries this repository to get the latest grid weather (monitored information) which plays an important role in the scheduling decision-making process. MonALISA's web client is also used as a 'window' to the grid to view the load distribution of the jobs across the grid sites. It works in conjunction with the tracker module of SPHINX to display statistics such as site-wise and user-wise job distribution. SPHINX interacts with MonALISA by accessing MonALISA's database providing monitored parameters

Virtual data toolkit (VDT): Grid resource management service

Virtual data toolkit [VDT] is a set of software that supports the needs of the research groups and experiments. It consists of server and client, and each part includes Condor [Thain03], Globus [Foster02], and Chimera and other software to submit requests to resource grid sites or to provide resource power to serve remote requests.

Based on the resource allocation decision made by Sphinx grid resource scheduling service VDT client submits requests to remote grid resources. The remote resources have VDT server installed to execute incoming requests. In the distributed service demonstration several grid resource across USCMS grid test-bed are used as VDT servers, and two VDT clients are available to make request submission based on the resource allocation decision.

3. Experimental Results

3.1 Experimental setup

In the experiments presented here, we use four grid sites from the US-CMS Grid Test-bed. A job is submitted to a site gatekeeper, and each site supports a scheduler (Condor) for local load balancing of the compute cluster within that site. A canonical DAG is prepared using the Chimera Virtual Data Language consisting of two execution nodes in which each node requires three external input files and generates one output file. A workflow of 120 such DAGs is stored in the Chimera Virtual Data Catalog and then submitted to the SPHINX scheduling server for each experiment.

3.2 Benchmark planning algorithms

In order to demonstrate the functionality of workflow management, a set of simple, benchmark planning and replication algorithms has been implemented into the scheduling framework. Several strategies currently exist in the Grid community and one may categorize them according to: matchmaking [Ramen 98], knowledge-based approach utilizing AI technologies [Blythe 03], data availability based strategies [Ranganathan 02] and economy based file access optimization [Carman 02]. Indeed, many of these strategies have been implemented in various Grid projects. The tests presented here do not yet attempt to implement such strategies, but rather attempt to show that SPHINX is able to provide researchers with an environment in which one may plug-in different strategy modules and compare the performance of existing and future scheduling strategies.

As benchmark strategies, we choose to select job execution sites according to the following two algorithms. The first is a simple round robin method, which is based on a First-In-First-Out (FIFO) strategy, and does not exploit the benefits of just-in-time scheduling. Such a strategy takes advantage of possible parallelism provided by multi-processor computing across different grid sites, but does not take advantage of important information such as grid topology, job tracking, grid weather, or data replication services. As a second strategy, we augment the round robin strategy with upper limits on the number of jobs that are scheduled at an execution site at any particular time. The upper limits at different sites are set according to the number of available compute processors available at each site. Together, the upper limits incorporate system-tuning information based on the topology of the grid and therefore act as a throttle for work submission, allowing for a richer, non-trivial use of just-in-time scheduling.

3.3 Observation

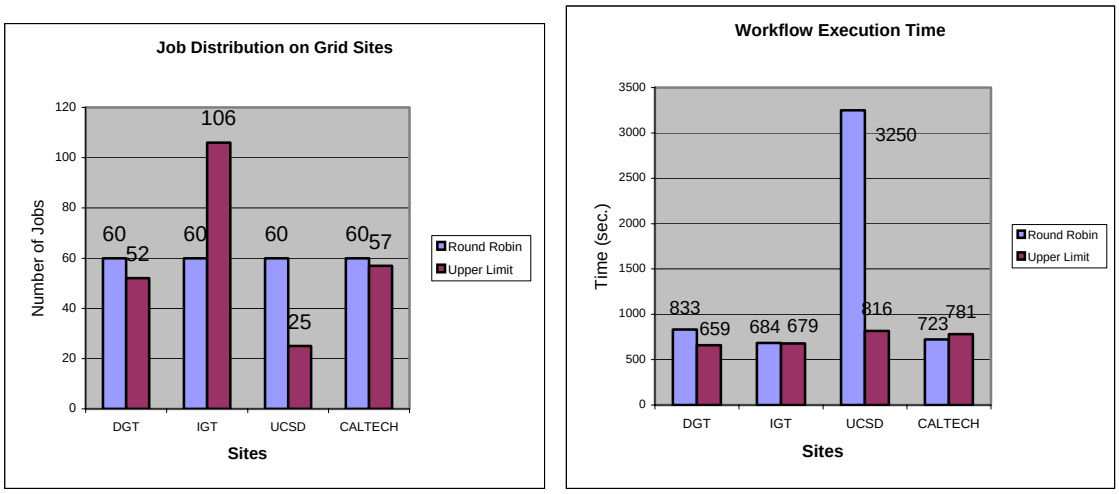

Figure 3: Job distribution and workflow execution time on grid sites according to the planning algorithms

Even though the number of assigned jobs on IGT is increased with large number, the workflow execution time on the site keeps stable. It means that IGT is capable to accept the addition al jobs without overload. On the other hand, the workflow execution time on UCSD is decreased dramatically with the decreased number of assigned jobs. It explains that extra overload on UCSD are taken by IGT leading to evenly distributed workload on the four sites. As a result, the workflow execution times are similar closely on all the sites with the upper limit-planning algorithm. By considering workload information on each grid site a planner can make more efficient planning of jobs to the grid sites.

Figure 4 shows planning and executing time of submitted jobs. From the first graph, we can detect that child job queuing and planning takes large amount of total DAG execution time on the both planning algorithms. The child job queuing and planning depend mostly on the parent job execution according to data dependency. A smart planning algorithm can reduce the queuing time by assigning the parent job to a computing resource efficiently resulting to reducing total DAG execution time. For example from our experiment, the upper limit method provides less queuing time than the round robin does. We can also detect that the ratio of the queuing and planning time of the child job over DAG execution time is greater with the upper limit method than with the round robin. We understand that the case happens because the job planning time on SPHINX is not affected much by the planning algorithms even though the queuing time and the execution time is changed according to the algorithms. From the first and second figures, the parent job planning time is consistent, while job execution time is changed a lot according to the planning algorithms.

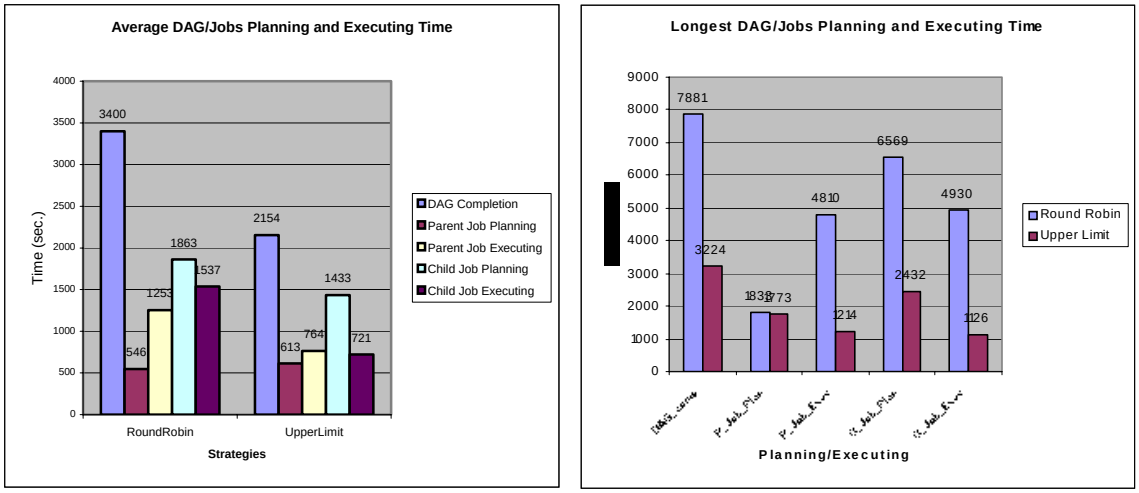

Figure 4: Average DAG and Jobs planning and executing time, and the longest planning and executing time

Comparing the longest planning and executing time with the average time for the both algorithms, we realize that time variation with the upper limit is much smaller than with the round robin. As we can find from Figure 3, with the round robin, UCSD is under the high overload, while IGT is not. From these facts, we can see that most jobs assigned to UCSD take the longest queuing and executing time, while jobs in IGT take short time, and it makes the large variation between the average and longest time. With the upper limit, the planner distributes jobs to the sites evenly according to the computing capabilities. It makes job planning and executing time on the entire sites consistent, and the variance becomes small.

4. Conclusion and Future Directions

Using the distributed services architecture provides the advantage of knowledge-based decision making. It alleviates the grid-user of scheduling the jobs himself on the grid. Job tracking and re-submission facilities add to the system fault-tolerance at a job-level. Analysis of results is expedited with easy remote file access.

More experiments are in progress which compares the different scheduling strategies and the effectiveness of monitoring and feedback information. We are also in the process of verification of adherence by Sphinx to policies and QoS requirements put down by the users.

5. Acknowledgements

- California Institute of Technology
  - Julian Bunn, Iosif Legrand, Harvey Newman, Suresh Singh, Conrad Steenberg, Michael Thomas, Frank Van Lingen, Yang Xia
- University of Florida
  - Dimitri Bourilkov, Craig Prescott
- Fermi National Accelerator Laboratory
  - Anzar Afaq, Greg Graham

6. References

[Basney 99] Basney, J., Livny, M. Deploying a High Throughput Computing Cluster, High Performance Cluster Computing, Rajkumar Buyya, Editor, Vol. 1, Chapter 5, Prentice Hall PTR, May 1999.

[Blythe 03b] Blythe, J., Deelman, E., Gil, Y., et al, The Role of Planning in Grid Computing, To appear in Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS), June 9-13, 2003, Trento, Italy.

[Carman 02] Carman, M., Zini, F., Serafini, L. Towards an Economy-Based Optimisation of File Access and Replication on a Data Grid, Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)

[CMS04] The compact muon solenoid, http://www.uscms.org, 2004

[Condor04] http://www.cs.wisc.edu/condor, 2004

[Deelman 02] Deelman, E., Blythe, J., Gil, Y., Kesselman, C. Pegasus: Planning for Execution in Grids. Technical Report GriPhyN-2002-20, Nov. 2002

[EDG04] http://www.eu-datagrid.org, 2004

[Foster99] I. Foster and C. Kesselman, Eds., The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, 1999.

[Foster03] Ian Foster, Jens Vockler, Michael Wilde, Yong Zhao, The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration, Proceedings of the 2003 CIDR Conference, January 2003

[Globus04] http://www.globus.org, 2004

[GriPhyN04] http://www.griphyn.org, 2004

[Grid3] http://www.ivdgl.org/grid3/, 2003

[In03] J. U. In, A. Arbree, P. Avery, R. Cavanaugh, S. Kategeri, and S. Ranka, "A scheduling middleware for data intensive applications on a grid, Tech. Rep. GriPhyN project technical report 2003-17, 2003.

[LCG04] http://lcg.cern.ch, 2004

[Newman03] H.B. Newman, I.C. Legrand, P. Galvez, R. Voicu, C. Cirstoiu, MonALISA: A Distributed Monitoring Service Architecture, CHEP 2003, La Jola, California, March 2003

[Rademakers98] Fons Rademakers, Rene Brun, ROOT: An Object-Oriented Data Analysis Framework, Linux Journal, Issue 51, July 1998

[Ramen 98] Ramen, R., Livny, M., Solomon, M., Matchmaking: Distributed Resource Management for High Throughput Computing, Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, July 28-31, 1998, Chicago, IL

[Ranganathan 02] Ranganathan, K., Foster, I. Decoupling Computation and data Scheduling in Distributed Data Intensive Applications. International Symposium for High Performance Distributed Computing (HPDC-11), Edinburgh, July 2002.

[Ruda 01] Ruda, M. Integrating GRID tools to build a computing resource broker: activities of DataGrid WP1. CHEP 2001, Beijing, September 2001.

[Sandholm] Thomas Sandholm, Jarek Gawor, Globus Toolkit 3 Core – Agrid Service Container Framework, http://www-unix.globus.org/toolkit/documentation.html

[Segal 00] Segal, B. Grid Computing: The European Data Project. IEEE Nuclear Science Symposium

[Steenberg03] http://clarens.sourceforge.net

[Thain03] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003. ISBN: 0-470-85319-0

[VDT04] http://www.lsc-group.phys.uwm.edu/vdt/, 2004

[Weissman 99] Weismann, J. Prophet: Automated Scheduling of SPMD Programs in Workstation Networks, Concurrency: Practice and Experience, Vol. 11, No. 6, May 1999, pp. 301-321.