Management of Grid Jobs and Information within SAMGrid

A. Baranovski, G. Garzoglio, A. Kreymer, L. Lueking,
S. Stonjek, I. Terekhov, F. Wuerthwein
FNAL, Batavia, IL 60510, USA
A. Roy, T. Tannenbaum
University of Wisconsin, Madison, WI 53706 , USA
P. Mhashilkar, V. Murthi
UTA, Arlington, TX 76019, USA
R. Walker
Imperial College, London, UK
F. Ratnikov
Rutgers University, Piscataway NJ 08854, USA
T. Rockwell
Michigan State University, East Lansing, MI 48824, USA

We describe some of the key aspects of the SAMGrid system, used by the D0 and CDF experiments at Fermilab. Having sustained success of the data handling part of SAMGrid, we have developed new services for job and information services. Our job management is rooted in Condor-G and uses enhancements that are general applicability for HEP grids. Our information system is based on a uniform framework for configuration management based on XML data representation and processing.

1. Introduction

Grid [1] has emerged as a modern trend in computing, aiming to support the sharing and coordinated use of diverse resources by Virtual Organizations (VO's) in order to solve their common problems[2]. It was originally driven by by scientific, and especially High-Energy Physics (HEP) communities. HEP experiments are a classic example of large, globally distributed VO's whose participants are scattered over many institutions and collaborate on studies of experimental data, primarily on data processing and analysis.

Our background is specifically in the development of large-scale, globally distributed systems for HEP experiments. We apply grid technologies to our systems and develop higher-level, community-specific grid services (generally defined in [3]), currently for the two collider experiments at Fermilab, D0 and CDF. These two experiments are actually the largest currently running HEP experiments, each having over half a thousand users and planning to analyze repeatedly petabyte scale data.

The success of the distributed computing for the experiments depends on many factors. In the HEP computing, which remains a principal application domain for the Grid as a whole, jobs are *data-intensive* and therefore *data handling* is one of the most important factors. For HEP experiments such as D0 and CDF, data handling is the center of the metacomputing grid system [4]. The SAM data handling system [5] was originally developed for the D0 collaboration and is currently also used by CDF. The system is described in detail elsewhere (see, for example, [6, 7] and references therein). Here, we only note some of the advanced features of the system – the ability to coordinate multiple concurrent accesses to Storage Systems [8] and global data routing and replication [9].

Given the ability to distribute data on demand globally, we face the similar challenges of distributing the processing of the data. Generally, for this purpose we need global job scheduling and information management, which is a term we prefer over "monitoring" as we strive to include configuration management, resource description, and logging.

In recent years, we have been working on the SAMGrid project [10], which addresses the grid needs of the experiments; our current focus is in the Jobs and Information Management (JIM), which is to complement the SAM grid data handling system with services for job submission, brokering and execution as well as distributed monitoring. Together, SAM and JIM form SAMGrid, a "VO-specific" grid system.

In this paper, we present some key ideas from our system's design. For job management per se, we collaborate with the Condor team to enhance the Condor-G middleware so as to enable scheduling of data-intensive jobs with flexible resource description. For information, we focus on describing the sites' resources in the tree-like structures of XML, with subsequent projections onto the Condor Classified Advertisements (ClassAd) framework, monitoring with Globus MDS and other tools.

The rest of the paper is organized as follows. We discuss the relevant job scheduling design issues and Condor enhancements in Section 2. In Section 3, we describe configuration management and monitoring. In Section 4, we present the status of the project and interfaces with the experiments' computing environ-

ment and fabric; we conclude in Section 5.

2. Job Scheduling and Brokering

A key area in Grid computing is job management, which typically includes planning of job's dependencies, selection of the execution cluster(s) for the job, scheduling of the job at the cluster(s) and ensuring reliable submission and execution. We base our solution on the Condor-G framework [11], a powerful Grid middleware commonly used for distributed computing. Thanks to the Particle Physics Data Grid (PPDG) collaboration [12], we have been able to work with the Condor team to enhance the Condor-G framework and then implement higher-level functions on top of it. In this Section, we first summarize the general Condor-G enhancements and then proceed to actually describing how we schedule data-intensive jobs for D0 and CDF.

2.1. The Grid Enhancements Condor

We have designed three principal enhancements for Condor-G. These have all been successfully implemented by the Condor team:

- Original Condor-G required users to either specify which grid site would run a job, or to use Condor-G's GlideIn technology. We have enabled Condor-G to use a matchmaking service to automatically select sites for users.
- We have extended the ClassAd language, used by the matchmaking framework to describe resources, to include externally supplied functions to be evaluated at match time. This allows the matchmaker to base its decision not only on explicitly advertised properties but also on opaque logic that is not statically expressible in a ClassAd. Other uses include incorporation of information that is prohibitively expensive to publish in a ClassAd, such as local storage contents or lists of site-authorized Grid users.
- We removed the restriction that the job submission client had to be on the same machine as the queuing system and enabled the client to securely communicate with the queue across a network, thus creating a multi-tiered job submission architecture.

Fundamentally, these changes are sufficient to form a multi-user, multi-site job scheduling system for generic jobs. Thus, a novelty of our design is that we use the standard Grid technologies to create a highly reusable framework to the job scheduling, as opposed to writing our own Resource Broker, which would be specific to our experiments.

In the remainder of this Section we described higher-level features for the job management, particularly important for data-intensive applications.

2.2. Combination of the Advertised and Queried Information in the MMS

Classic matchmaking service (MMS) gathers information about the resources in the form of published ClassAds. This allows for a general and flexible framework for resource management (e.g. jobs and resource matching), see [13]. There is one important limitation in that scheme, however, which has to do with the fact that the entities (jobs and resources) have to be able to express all their relevant properties upfront and *irrespective of the other party*.

Recall that our primary goal was to enable co-scheduling of jobs and data. In data-intensive computing, jobs are associated with long lists of data items (such as files) to be processed by the job. Similarly, resources are associated with long lists of data items located, in the network sense, near them. For example, jobs requesting thousands of files and sites having hundreds of thousands of files are not uncommon in production in the SAM system. Therefore, it would not be scalable to explicitly publish all the properties of jobs and resources in the ClassAds.

Furthermore, in order to rank jobs at a resource (or resources for the job), we wish to include additional information that couldn't be expressed in the ClassAds at the time of publishing, i.e., before the match. Rather, we can analyze such an information during the match, *in the context of the job request*. For example, a site may prefer a job based on similar *already scheduled* data handling requests. [1] Another example of useful additional information, not specific to data-intensive computing, is the pre-authorization of the job's owner with the participating cite, by means of e.g. looking up the user's grid subject in the site's **gridmapfile**. Such a pre-authorization is not a replacement of security, but rather a means of protecting the matchmaker from some blunders that otherwise tend to occur in practice.

The original MMS scheme allowed for such additional information incorporation only in the *claiming* phase, i.e., after the match when the job's scheduler actually contacts the machine. In the SAMGrid design, we augment information processing by the MMS with the ability *to query* the resources with a job in the context. This is pictured in Figure 1 by arrows extending from the resource selector to the resources,

[1] Unlike the information about data already placed at sites, the information about scheduled data requests, and their estimated time of completion, is not described by any popular concept like replica catalog.

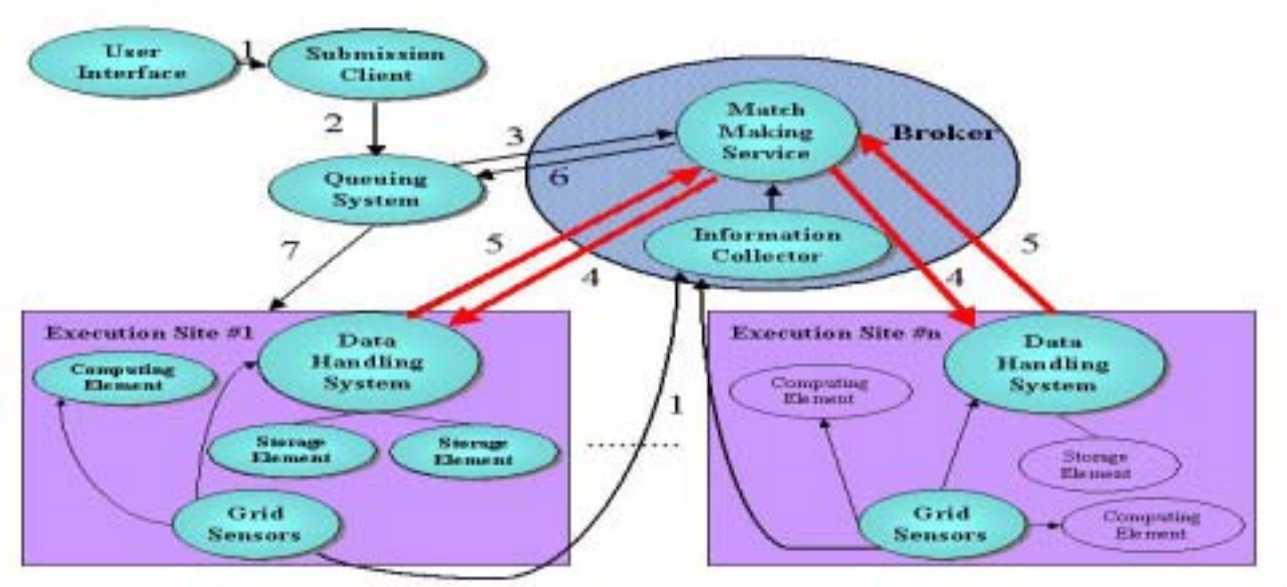

Figure 1: The job management architecture in SAMGrid. 1,2 – Jobs are submitted while resources are advertised, 3 – MMS matches jobs with resources, 4,5 – ranking functions retrieve additional information from the data handling system, 6,7 – resource is selected and the job is scheduled.

specifically to the local data handling agents, in the course of matching. It is implemented by means of externally supplied ranking function whose evaluation involves remote call invocation at the resources' sites. Specifically, the resource ClassAds in our design contain pointers to additional information providers (data handling servers called Stations):

```
Station_ID = foo
```

and the body of the MMS ranking function called from the job classAd

```
Rank = fun(job_dataset,
           OTHER.Station_ID)
```

includes logic similar to this pseudo-code:

```
station = resolve(Station_ID,...)
return station->
 get_preference(job_dataset,...)
```

In the next subsection, we discuss how we believe this will improve the co-scheduling of jobs with the data.

2.3. Interfacing with the SAM Data Handling System

The co-scheduling of jobs and data has always been critical for the SAM system, where at least a subset of HEP analysis jobs (as of the time of writing, the dominating class) have their latencies dominated by data access. Please note that the SAM system already implemented the advanced feature of retrieving multi-file datasets asynchronously with respect to the user jobs [14, 15] – this was done initially at the cluster level rather than at the grid level.

Generally with the data-intensive jobs, we attempt to minimize the time to retrieve any missing data and the time to store output data, as these times propagate into the job's overall latency. As we try to minimize the grid job latency, we ensure that the design of our system, Figure 1 is such that the data handling latencies will be taken into account in the process of job matching. This is a principal point of the present paper, i.e., while we do not yet possess sufficient real statistics that would justify certain design decisions, we stress that our system design enables the various strategies and supports considerations listed below.

In the minimally intelligent implementation, we prefer sites that contain most of the the job's data. Our design does not rely on a replica catalogue because in the general case, we need *local* metrics computed by and available from the data handling system:

- The network speeds for connections to the sources of any missing data;
- The depths of the queues of data requests for both input and output;
- The network speeds for connections to the nearest destination of the output files.[2]

[2]In the SAM system the concept of data routing is implemented such that the first transfer of an output file is seldom done directly to the final destination.

It is important that network speeds be provided by a high-level service in the data handling rather than by a low-level network sensor, for reasons similar to those why having e.g. a 56Kbps connection to the ISP does not necessarily enable one to actually download files from the Internet with that speed.

3. The Management of Configuration and Information in JIM

Naturally, being able to submit jobs and schedule them more or less efficiently is necessary but not sufficient for a Grid system. One has to understand how resources can be described for such decision making, as well as provide a framework for monitoring of participating clusters and user jobs.

We consider these and other aspects of *information management* to be closely related to issues of Grid configuration. In the JIM project, we have developed a uniform configuration management framework that allows for generic grid services instantiation, which in turn gives flexibility in the design of the Grid as well as inter-operability on the Grid (see below).

In this Section, we briefly introduce the main configuration framework and then project it onto the various SAMGrid services having to do with information.

3.1. The Core Configuration Mechanism

Our main proposal is that grid sites be configured using a site-oriented schema, which describes both resources and services, and that grid instantiation at the sites be derived from these site configurations. We are not proposing any particular site schema at this time, although we hope for the Grid community as a whole to arrive at a common schema in the future which will allow reasonable variations such that various grids are still instantiatable.

Figure 2 shows configuration derivation in the course of instantiation of a grid at a site. The site configuration is created using a meta-configurator similar to one we propose below.

3.1.1. The Core Meta-Configurator and the Family of Configurators

In our framework, we create site and *all* other configurations by a universal tool which we call a *meta-configurator*, or configurator of configurators. The idea is to separate the process of querying the user for values of attributes from the *schema* that describes what those attributes are, how they should be queried, how to guess the default values, and how to derive values of attributes from those of other attributes. Any concrete configurator uses a concrete schema to ask the relevant questions to the end user (site administrator) in order to produce that site's configuration. Any particular schema is in turn derived from a meta-schema. Thus, the end configuration can be represented as:

$$C = c(S_d, I_u) = c(c(S_0, I_d), I_u),$$

where C is a particular configuration, c is the configuration operation, S_d is a particular schema reflecting certain design, S_0 is the meta-schema, I_d and I_u are the inputs of the designer and the user, respectively.

In our framework, configurations and schemas are structures of the same type, which we choose to be trees of nodes each containing a set of distinct attributes. Our choice has been influenced by the successes of the XML technologies and, naturally, we use XML for representing these objects.

To exemplify, assume that in our *present* design, a grid **site** consists of one or more **clusters** each having a name and an architecture (homogenous), as well as exactly one **gatekeeper** for Grid access. Example configuration is:

```
<?xml version='1.0'?>
<site name='FNAL'
      schema_version='v0_3'>
  <cluster name='samadams'
           architecture='Linux'>
    <gatekeeper ...>
  </cluster>
</site>
```

This configuration was produced by the following schema

```
<?xml version='1.0'?>
<site cardinalityMin='1'
      cardinalityMax='1'
      name='inquire-default,FNAL' >
   <cluster  cardinalityMin='1'
     name='set,CLUSTERNAME,inquire'
     architecture=
      'inquire-default,exec,uname'/>
</site>
```

in an interactive session with the site administrator as follows:

```
What is the name of the site ? [FNAL]:
 <return>
What is the name of cluster
  at the site 'FNAL'?  samadams
What is the architecture
  of cluster 'samadams' [Linux]?
...
```

When the schema changes or a new cluster is created at the site, the administrator merely needs to re-run the tool and answer the simple questions again.

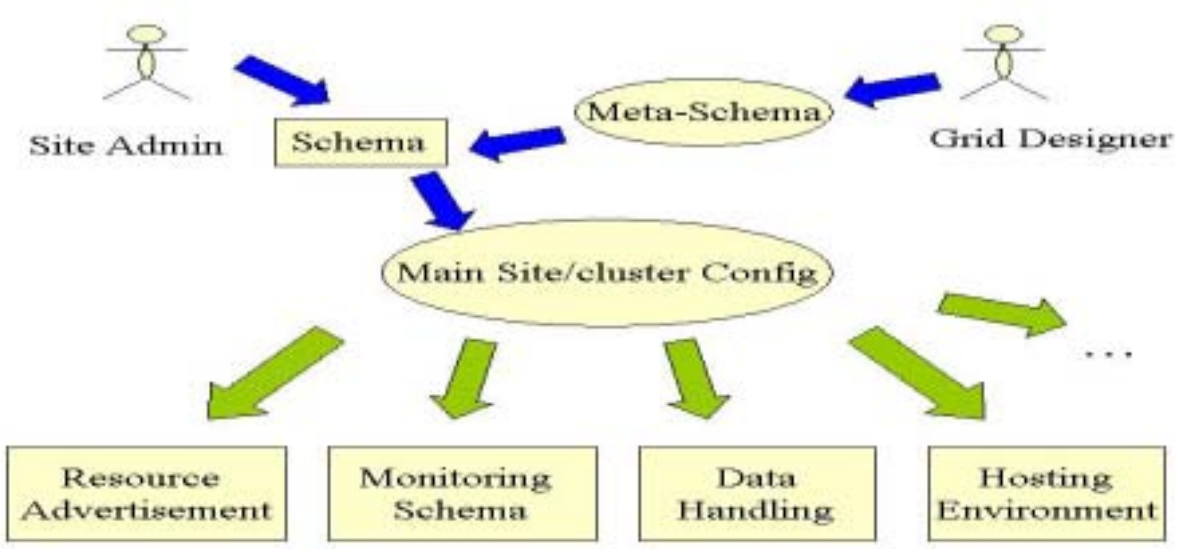

Figure 2: Configuration creation and derivation in our framework. Service collections are typical of the SAMGrid project.

3.2. Resource Advertisement for Condor-G

In the JIM project, we have designed the grid job management as follows. We advertise the participating grid clusters to an information collector and grid jobs are matched [13] with clusters (resources) based on certain criteria primarily having to do with the data availability at the sites. We have implemented this job management using Condor-G [11] with extensions that we have designed together with the Condor team [10].

For the job management to work as described in Section 2, we need to advertise the clusters together with the **gatekeepers** as the means for Condor to actually schedule and execute the grid job at the remote site. Thus, our *present* design requires that each advertisement contain a cluster, a gatekeeper, a SAM station (for jobs actually intending to process data) and a few other attributes that we omit here. Our advertisement software then selects from the configuration tree all patterns containing these attributes and then applies a ClassAd generation algorithm to each pattern.

The selection of the subtrees that are ClassAd candidates is based on the XQuery language. Our queries are generic enough as to allow for *design evolution*, i.e. to be resilient to some modifications in the schema. When new attributes are added to an element in the schema, or when the very structure of the tree changes due to insertion of a new element, our advertisement service will continue to advertise these clusters with or without the new information (depending on how the advertiser itself is configured) but the important factor is that this site will continue to be available to our grid.

For example, assume that one cluster at the site from subsection 3.1.1 now has a new grid gatekeeper mechanism from Globus Toolkit 3, in addition to the old one:

```
<?xml version='1.0'?>
<site name='FNAL'
      schema_version='v0_3'>
  <cluster name='samadams'
           architecture='Linux'>
    <grid_accesses>
       <gatekeeper ...>
       <gatekeeper-gtk3 ...>
    </grid_accesses>
    ...
```

Assume further that our particular grid is not yet capable of taking advantage of the new middleware and we continue to be interested in the old **gatekeeper** from each cluster. Our pattern was such that a **gatekeeper** is a descendant of the **cluster** so we continue to generate meaningful ClassAds and match jobs with this site's cluster(s).

3.3. Monitoring Using Globus MDS

In addition to advertising (pushing) of resource information for the purpose of job matching, we deploy Globus MDS-2 for pull-based retrieval of information about the clusters and activities (jobs and more, such as data access requests) associated with them. This

allows us to enable web-based monitoring, primarily by humans, for performance and troubleshooting [10]. We introduce (or redefine in the context of our project) concepts of **cluster**, **station** etc, and map them onto the LDAP attributes in the OID space assigned to our project (the FNAL organization, to be exact) by the IANA[16]. We also create additional branches for the MDS information tree as to represent our concepts and their relations.

We derive the values of the *dn*'s on the information tree from the site configuration. In this framework, it is truly straightforward to use XSLT (or a straight XML-parsing library) to select the names and other attributes of the relevant pieces of configuration. For example, if the site has two clusters defined in the configuration file, our software will automatically instantiate two branches for the information tree. Note that the resulting tree may of course be distributed within the site as we decide e.g. to run an MDS server at each cluster, which is a separate degree of freedom.

3.4. Multiple Grid Instantiation and Inter-Operability

We have been mentioning that there are in fact several other grid projects developing high-level Grid solutions; some of the most noteworthy include the European Datagrid [17], the Crossgrid[18], and The NorduGrid [19]. Inter-Operability of grids (or of solutions on The Grid if you prefer) is a well-recognized issue in the community. The High Energy and Nuclear Physics InterGrid [20] and Grid Inter-Operability [21] projects are some of the most prominent efforts in this area. As we have pointed out in the Introduction, we believe that inter-operability must include *the ability to instantiate and maintain multiple grid service suites at sites*.

A good example of inter-operability in this sense is given by various cooperating Web browsers which all understand the user's bookmarks, mail preferences etc.. Of course, each browser may give a different look and feel to its "bookmarks" menu, and otherwise treat them in entirely different ways, yet most browsers tend to save the bookmarks in the common HTML format, which has *de facto* become the standard for bookmarks. Our framework, proposed and described in this Section, is a concrete means to facilitate this aspect of inter-operability. Multiple grid solutions can be instantiated using a grid-neutral, site-oriented configuration in an XML-based format.

We can go one step further and envisage that the various grids instantiated at a site have additional, separate configuration spaces that can easily be conglomerated into a *grid instantiation database*. In practice, this will allow the administrators e.g., to list all the Globus gatekeepers with one simple query.

4. Integration and Project Status

To provide a complete computing solution for the experiments, one must integrate grid-level services with those on the fabric. Ideally, grid-level scheduling complements, rather than interferes with, that of local batch systems. Likewise, grid-level monitoring should provide services that are additional (orthogonal) to those developed at the fabric's facilities (i.e., monitoring of clusters' batch systems, storage systems etc.).

Our experiments have customized local environments. CDF has been successfully using Cluster Analysis Facility (CAF), see [22]. D0 has been using MCRunJob [23], a workflow manager which is also part of the CMS computing insfrastructure. An important part of the SAMGrid project is to integrate its job and information services with these environments.

For job management, we have implemented GRAM-compliant job managers which pass control from Grid-GRAM to each of these two systems (which in turn are on top of the various batch systems). Likewise, for the purposes of (job) monitoring, these systems supply information about their jobs to the XML databases which we deploy on the boundary between the Grid and the Fabric. (For resource monitoring, these advertise their various properties using the frameworks described above).

We delivered a complete, integrated prototype of SAMGrid in the Fall of 2002. Our initial testbed linked 11 sites (5 D0 and 6 CDF) and the basic services of grid job submission, brokering and monitoring. Our near future plans include further work on the Grid-Fabric interface and more features for troubleshooting and error recovery.

5. Summary

We have presented the two key components of the SAMGrid, a SAM-based datagrid being used by the Run II experiments at FNAL. To the data handling capabilities of SAM, we add grid job scheduling and brokering, as well as information processing and monitoring. We use the standard Condor-G middleware so as to maximize the reusability of our design. As to the information management, we have developed a unified framework for configuration management in XML, from where we explore resource advertisement, monitoring and other directions such as service instantiation. We are deploying SAMGrid at the time of writing this paper and learning from the new experiences.

6. Acknowledgements

This work is sponsored in part by DOE contract No. DE-AC02-76CH03000. Our collaboration takes place as part of the DOC Particle Physics Data Grid (PPDG), [12] Collaboratory SciDAC project. We thank the many participants from the Fermilab Computing Division, the D0 and CDF Experiments, and the members of the Condor team for fruitful discussions as well as the development of our software and other software that we rely upon.

References

[1] I. Foster and C. Kesselman (eds.) *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufman, 1999.

[2] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *International Journal of High Performance Computing Applications*, 15(3), 200-222, 2001.

[3] I. Foster, C. Kesselman, J. Nick, S. Tuecke, Grid Services for Distributed System Integration, *IEEE Computer* 35(6), 2002.

[4] I. Terekhov, "Meta-Computing at D0", plenary talk at ACAT-2002, see [10], in Proceedings.

[5] The SAM project home page, http://d0db.fnal.gov/sam.

[6] V. White *et al.*, "D0 Data handling", plenary talk at The International Symposium on Computing in High Energy Physics (CHEP) 2001, September 2001, Beijing China, in Proceedings. L. Carpenter *et al.*, "SAM Overview and Operational Experience at the Dzero Experiment", ibidem. L. Lueking *et al.*, 'Resource Management in SAM and the D0 Particle Physics Data Grid", ibidem.

[7] V. White *et al.* "SAM and the Particle Physics Data Grid", see [6]

[8] I. Terekhov *et al.*, "Distributed Data Access and Resource Management in the D0 SAM System" in Proceedings of 10-th International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, July 2001, San-Fransisco, CA

[9] V. White and I. Terekhov, "SAM for D0 - a fully distributed data access system", talk given at VII International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT-2000), October, 2000, Batavia, IL.

[10] G. Garzoglio, "The SAM-GRID Project: Architecture and Plan", in Proceedings of *The VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT-2002)*, June 2002, Moscow, Russia.

[11] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, "Condor-G: A Computation Management Agent for Multi-institutional Grids", in the same proceedings as [8].

[12] The Particle Physics Data Grid, http://www.ppdg.net.

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

[14] I. Terekhov and V. White, "Distributed Data Access in the Sequential Access Model in the D0 Run II data handling at Fermilab", in Proceedings of The 9-th International Symposium on High Performance Distributed Computing (HPDC-9), August 2000, Pittsburgh, PA

[15] I. Terekhov for the D0 collaboration, Distributed Processing and Analysis of Physics Data in the D0 SAM System at Fermilab, FERMILAB-TM-2156.

[16] Internet Assigned Numbers Authority, http://www.iana.org.

[17] P. Kunszt, "Status of the EU DataGrid Project", in the Proceedings of ACAT-2002 The project home page is at http://www.eu-datagrid.org.

[18] M. Kunze, "The CrossGrid Project", in the Proceedings of ACAT-2002.

[19] A. Konstantinov, "The NorduGrid Project: Using Globus Toolkit for Building Grid Infrastructure", in the Proceedings of ACAT-2002.

[20] High Energy and Nuclear Physics InterGrid, http://www.hisb.org/

[21] Grid Inter-Operability, http://www.grid-interoperability.org/.

[22] M. Neubauer, "Computing at CDF", in these Proceedings.

[23] G. Graham, I. Bertram and D. Evans, "McRunJob: A High Energy Physics Workflow Planner for Grid Production Processing", in These Proceedings