

Ph.D. THESIS PROPOSAL

A globally distributed system for job, data and information handling for high energy physics

Gabriele Garzoglio

September 08, 2004

Abstract

The computing infrastructures of the modern high energy physics experiments need to address an unprecedented set of requirements. The collaborations consist of hundreds of members from dozens of institutions around the world and the computing power necessary to analyze the data produced surpasses already the capabilities of any single computing center. A software infrastructure capable of seamlessly integrating dozens of computing centers around the world, enabling computing for a large and dynamical group of users, is of fundamental importance for the production of scientific results. Such a computing infrastructure is called a computational grid.

The SAM-Grid offers a solution to these problems for CDF and DZero, two of the largest high energy physics experiments in the world, running at Fermilab. The SAM-Grid integrates standard grid middleware, such as Condor-G and the Globus Toolkit, with software developed at Fermilab, organizing the system in three major components: data handling, job handling, and information management. This research proposal presents the challenges for such a computing infrastructure and introduces the solutions already working and the ones in phase of development.

Contents

1	Introduction	3
1.1	The standard grid middleware	4
1.2	The SAM-Grid system	6
2	The information management component	10
2.1	The configuration infrastructure	11
2.1.1	The configuration process	13
2.1.2	Site and Product configuration	14
2.2	The Monitoring and Logging Infrastructures	16
3	The Job Management Component	20
3.1	The SAM-Grid Client	24
3.2	The Job Submission Service	28
3.3	The Resource Selection Service	31
4	The Execution Site	35
4.1	Local Batch System Adaptation	36
4.2	Dynamic Product Installation	37
4.3	Local Sandbox Management	39
5	Summary	40

1 Introduction

The collaborations of high energy physics experiments are traditionally large and geographically distributed. Only a few laboratories around the world offer the facilities necessary to conduct innovative programs of research in the field. Physicists from all over the world gather at these sites, building particle detectors specifically designed to best exploit the unique characteristics of these facilities. Nowadays, collaborations involving scientists from dozens of different nationalities are becoming increasingly common. In addition, in the past decades, the complexity and the cost of these detectors have increased dramatically, as new and more advanced physics questions have become the focus of the community. To confront the unprecedented budgets and the challenges posed by the research programs, the collaborations have naturally increased in size, involving typically many hundreds of scientists.

The complexity of the physics and the detectors does not only entail larger numbers of collaborators. The amount of data to be gathered, catalogued, processed and analyzed, in fact, is on the order of a Petabyte per year per experiment. Various factors contribute to the exponential growth of the data size. First, the observation of the physics of interest requires a much greater spatial resolution than in the past: the typical number of data channels of a modern detector reaches in the millions, an order of magnitude larger than a decade ago. Second, high energy physics studies phenomena statistical in nature and most of the particles studied in today's research programs have an extremely rare probability of occurring. The characteristics of such rare particles can be studied with the required level of statistical significance only if enough of them are observed during the data taking. In the modern experiments, it is usual to gather data for many years and observe a few dozen occurrences of a certain rare particle, while the total number of background particles amounts to many billions.

In this scenario, addressing the computing needs of the modern high energy physics collaborations presents many challenges, and the amount of data and large geographical scale are only the most evident. For most experiments today, the processing power required to produce innovative scientific results is already larger than what a single central computing facility can provide. In the past, the multinational institutions participating in the experiment generally financed the hosting laboratory to buy the computing hardware and take the responsibility of managing it. Today, this trend is reversing, and, typically, national funding agencies are willing to invest in computing

infrastructures as long as they are maintained locally and dedicated to more than one experiment or even discipline. This politics is popular because it promotes the development of national computing centers and expertise, and, at the same time, it streamlines the maintenance of the computing systems by promoting resource sharing. This trend leads to the formation of computing centers very diverse with respect to hardware, software systems, configurations, management policies, and availability. A global software infrastructure capable of interfacing to each individual center is therefore the key to enabling transparent access to the total pool of computing resources.

Besides the diversity of the computing fabric, another important challenge is the dynamic nature of the membership to the collaborations. Typically, scientists join and leave the physics collaboration throughout the lifetime of the research program. This group of people, temporarily working together to accomplish a common goal, is sometimes referred to as a “Virtual Organization”. A successful distributed computing infrastructure must preserve the efficiency and, at the same time, guarantee the security of the computing environment.

In summary, the challenges of such a global software infrastructure is to enable a Virtual Organization to handle Petabytes of data in a secure, accountable and transparent fashion, accessing a Petaflop-scale pool of shared and distributively owned resources. This software infrastructure is also called a “Computational Grid” [1], in analogy to the Electrical Power Grid: in the same fashion as the power grid provides electricity to the users irrespectively of where it is produced, analogously, the computational grid gives transparent access to geographically distributed computing resources.

1.1 The standard grid middleware

Astronomers and high energy physicists have started moving toward grid computing between five to ten years ago (see the collaborations of BaBar [2], Belle [3], DZero [4], CDF [5], SDSS [6], and LIGO [7]). At the same time, many groups of physicists and computer scientists are addressing the same problems for the next generation of experiments, which will start taking data in 2007 at the Large Hadron Collider (LHC) at the European Laboratory for Particle Physics (CERN) [8], Switzerland (see the collaborations of CMS [9, 10], Alice [11, 12], LHCb [13], and Atlas [14]). In this scenario, the proliferation of distinct solutions to very similar problems risks to become a big concern. Not only is this a duplication of effort in a time where the

budget for high energy physics research grows only by one to two percent per year, but this also implies in projection a cost to educate the scientists to use many different computing infrastructures. At this point, there is clearly a need for the scientific community to come together and define what services a grid must provide and by what protocols and interfaces.

Since 2001, these matters are discussed by hundreds of people gathering at the Global Grid Forum (GGF) Conferences [15]. The conferences are organized around working groups, which provide documents to define the specific problems, to propose the standards and, sometimes, provide reference implementations. The leader in this effort is the Globus Alliance [16], which provides open source technologies that are used as building blocks of many grids for scientific communities and the industry. Their Globus Toolkit [17] provides a de facto standard implementation of four major components of a grid computing system:

1. Security: the Globus Security Infrastructure (GSI) is based on a Public Key Infrastructure that uses X509 certificates. Certificates define the identity of software services and people. They are generated by users and administrators and certified (signed) by Certificate Authorities, which, ultimately, define the trust relationship in the system. All the software in the Toolkit is integrated with GSI.
2. Data Management: these components offer data movement via a GSI-enabled FTP service called GridFTP, and replica location services via the Globus Replica Catalog.
3. Information Services: the resources of a computing cluster and the jobs running therein can be monitored using the Monitoring and Discovery Service (MDS), a GSI-enabled version of an LDAP server. MDS offers also an indexing service, which provides access to information across multiple information servers.
4. Resource Management: provides a reference implementation of the Globus Resource Allocation and Management (GRAM) protocol. The GRAM server, called gatekeeper, offers simple interfaces to manage jobs running on underlying batch systems.

Various other groups also provide similar or competing software components, which are less popular than the Globus Toolkit within the high energy physics

community (see Avaki [18], Platform Computing [19], Entropia [20], Sun Grid Engine [21], United Devices [22], Parabon [23], ProcessTree [24], Popular Power [25], Mojo Nation [26], and DataSynapse [27]).

Despite its wide acceptance, the components of the Globus Toolkit are not usable “out of the box” as a complete grid solution. The GRAM protocol is not user friendly and it does not provide persistency in the job submission request or error recovery mechanisms. The Monitoring and Discovery Service is well suited for gathering information upon request, but it does not allow information to be pushed into the system, a characteristic needed to monitor event driven systems. The Data management tools do not provide mechanisms for space allocation, data caching or scratch space management. In other words, the components are too low level to provide, alone, a full solution.

Because of this, various groups have developed software that offer higher-level services, building on top of the Globus Toolkit [28, 29, 30, 31]. A leading such group is the Condor Team, the developers of the Condor batch system [32]. For more than a decade, they have been developing software to support high throughput computing, delivering large amounts of processing capacity over long periods. In the past years, they have reused some of the concepts of the Condor batch system to address the problem of job management on the grid. Condor-G [33], the GRAM enabled extension to Condor, provides reliable and robust job management via durable distributed transactions and error recovery mechanisms as well as a user-friendly system interface.

The Globus Toolkit and Condor are solutions so widely used together that are often referred to as the standard middleware. They are the central piece of various middleware distribution packages [34, 35] and are the basic components of many grid projects throughout the world [36, 37, 38, 39, 40, 41, 42, 43, 44, 45].

1.2 The SAM-Grid system

The SAM-Grid [46, 47, 48, 49, 50, 51] is a computing project started at the Fermi National Accelerator Laboratory [52] in January 2002 and it is one of the first grids deployed for the high energy physics community. The Laboratory, in Batavia Illinois, operates the particle accelerator with the highest energy in the world, the Tevatron. The accelerator is used by two of the largest particle physics experiments currently taking data: CDF (Central Detector at Fermilab) [5] and DZero [4]. The project is conducted as a collab-

orative effort between physicists and computer scientists and it is financed by the Particle Physics Data Grid (PPDG) [53], in the US, and GridPP [54], in the UK. The goal of the SAM-Grid is to enable fully distributed computing for the DZero and CDF experiments, integrating standard grid tools with in-house software, when the standards do not provide an appropriate solution. The SAM-Grid computing infrastructure is the focus of this dissertation.

The SAM-Grid architecture is composed of three major components: the data handling, the job and the information management systems. This division is mostly natural as it closely follows the organization of the standard middleware and best capitalizes on the software already developed at Fermilab for the experiments. The most notable of this in-house software is the Sequential Access via Metadata (SAM) [55, 50, 56, 57, 58, 59, 60, 61, 62], the data handling system of the experiments. Figure 1 shows an architectural diagram of the SAM-Grid.

The SAM project was started at Fermilab in 1997 to address the data handling challenge that the DZero experiment was going to face throughout the following decade. As the system grew more configurable and operationally stable, in 2001 CDF opted to adopt SAM for its data handling needs. Today the system manages a throughput of Terabytes of data throughout dozens of sites in America, Europe and Asia.

The SAM project is designed and implemented with four principal goals in mind. First, to provide reliable data storage, for data coming either directly from the detector or from data processing facilities around the world. Second, to enable data distribution among all the collaborating institutions, today on the order of 70 per experiment. Third, to thoroughly catalogue the data for content, provenance, status, location, processing history, user-defined datasets, et cetera. Fourth, to manage the distributed resources in order to optimize their usage and, ultimately, the data throughput, while enforcing the administrative policies of the experiments.

Other groups have developed systems that address some of these goals (SRB [63], GDMP [64, 65], Giggle [66], NeST [67] and references therein, Magda [68], DataCutter [69], and the Global Grid Forum [70]). As of today, SAM is still arguably the most comprehensive data handling solution for the high energy physics domain.

The flexibility and stability of the SAM system is of central importance for the job and information management infrastructure of the SAM-Grid. Both services rely heavily on the maturity of SAM, in order to provide solutions to classical grid problems, such as data pre-staging and job/data colocation (see

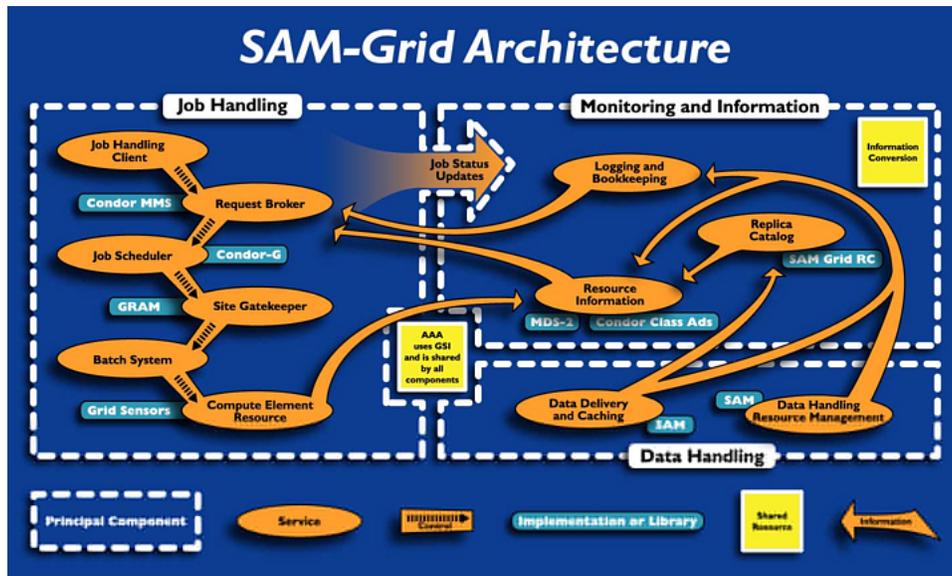


Figure 1: The SAM-Grid is divided into three major components: data handling, job handling and information management. All three components are integrated with strong security mechanisms. Each orange bobble represents an abstract aggregated service, whose implementation appears in the blue label next to it. The major challenge of the project was integrating all the services to enable globally distributed computing for the DZero and CDF experiments at Fermilab.

section 3.3). For these reasons, the SAM system is thoroughly described in the dissertation, even if the focus of this research is the job and information management components.

In high energy physics, as in other fields, a computational task can be decomposed in one or more units of computation, called jobs. The job handling component of the SAM-Grid is designed to provide robust, fault tolerant and efficient management of jobs on the grid. In addition, it provides a user interface especially designed for high energy physics applications. Various groups address this problem in the context of other grid infrastructures. The European Data Grid (EDG) [36] software has provided a solution with the development of a large, arguably monolithic code base, based on the standard middleware. LCG [37] is extending the infrastructure of EDG to address the

needs of the LHC experiments. The GriPhyN (Grid Physics Network) [42] project uses the standard middleware for the low level job management, and introduces the concept of virtual data to organize job dependencies. See also [71] and [72].

The Condor and the SAM-Grid teams decided in 2001 to address the computational requirements of large high energy physics experiments by enhancing the Condor-G framework. The advantages of such approach consist mainly in promoting the standards and ease the maintenance of the infrastructure by encapsulating the domain specific software to a series of modular, lightweight plug-ins. The challenges posed by this work are presented in chapter 3 and the solutions proposed will be discussed in the dissertation.

In order to reliably execute jobs, the job management component of every grid relies on a series of services at the resources. In addition to standard services, such as a local scheduler, the grid needs services that manage the grid jobs and the resources at the site. These services are sometimes called the fabric services. In the SAM-Grid architecture, they provide local batch system adaptation for the incoming grid jobs, dynamic product installation, intra cluster transport of the job and its output, et cetera. For most of these services, standard and mature technologies are not available to the community, thus, the necessity of spending most of our in-house development on them. A discussion on the design of the SAM-Grid fabric services can be found in this research proposal in chapter 4 and the details of the implementation in the dissertation.

In most grid infrastructures, the job management and data handling components heavily depend on the information management system to accomplish their tasks. The data handling is interfaced to the information system for a number of services. The first and foremost for high energy physics is data bookkeeping, as it enables the reproducibility of the physics results. The resource selection service, a central component of any sophisticated job dispatch infrastructure, cannot function without an information system capable of discovering resources and propagating their characteristics. The resource selection service of LCG, also called the broker [73] relies on the RGMA (Relational Grid Monitoring Architecture) [74] information service to function. The SAM-Grid resource selection service is based on the information collection mechanisms of the Condor match making service (chap. 3). Finally, the information system is used by the users to track the progress of their jobs and the status of the resources.

For the SAM-Grid, the information management is organized in three

categories: first, the static or semi-static information, which deals with the configuration of the grid products, services and resources. Second, the dynamic information, which is mostly used for monitoring. Third, the historical view of the previous information, which consists of the logging and bookkeeping services. Each category of information, when considered in the context of a grid system, presents a different set of challenges. The SAM-Grid information management infrastructure is discussed in chapter 2.

As the SAM-Grid is expanding in size and usage, we are gathering useful first hand information on the challenges of deploying and operating a grid for high energy physics. In addition to describing the system, the dissertation will present such challenges and report on the “lesson learned” for this first generation of grid systems.

The proposed dissertation will focus on the job and information management components of the SAM-Grid and will address problems that have arisen during the phase of design, implementation, and deployment of the system. We now present these problems and our developed solutions organized by component.

2 The information management component

A software environment designed to support the computing of a modern high energy physics experiment needs to address the challenges associated with handling large amounts of data, processed by jobs that run on a distributively owned, shared and dynamic resource infrastructure. In recent years, various research groups have proposed theoretical models to describe the status of such software environments. Some of these groups use simulations to study the behavior of the systems and propose different configurations to optimize metrics, such as time to completion of jobs or network bandwidth [75]. While in a theoretical model the information associated with the system can be viewed as the system status over time in some abstract parameter space, in a physical system, such as the SAM-Grid, it is hard to deal with concepts like global status or list a complete set of parameters that would define such status. Rather, information management can benefit from a classification of the information in three categories, treated in practice with different software tools:

1. the information that is static or semi-static in nature, such as the global parameters of the grid or the setup of services and resources at a site. Changing this information generally involves a human in-

tervention, typically by a system or service administrator. We refer to this category of information with the generic name of grid configuration and we talk more about it in section 2.1;

2. the information associated with the behavior of the entities in the grid, such as resources, services, jobs, et cetera. This information is dynamic in nature and it is captured in various degrees and representations by the monitoring infrastructure of the grid (section 2.2);
3. the historical view of the previous two categories, with the appropriate level of synthesis: this category is useful mainly for bookkeeping, an activity of fundamental importance for the reproducibility of scientific results, and for statistical studies of the grid, interesting for the comparison between theoretical models and physical systems. We discuss this last category of information in section 2.2.

2.1 The configuration infrastructure

The SAM-Grid configuration system provides facilities to manage only those parameters that define the behavior of its services. It does not attempt to offer tools to administer the whole grid infrastructure, such as operating system components (e.g. libraries and devices) or standard system applications (e.g. batch and file systems). This latter problem is addressed on the Grid by a few emerging software infrastructures [76, 77, 78] that extend the classical software distribution mechanisms for local clusters. It should be noted that these tools address configuration at the level of the operating system and of a few standard applications. The configuration of high-level software services is in general more complex in nature and not necessarily configurable automatically. We expose the problem in more detail in the rest of this section.

There are various challenges that a grid configuration infrastructure must address, first of all the large number of parameters. This factor has two main impacts. The first is the high likelihood of name conflicts. In order to prevent name conflicts, the framework will have to use some form of name spacing for the parameters. The second impact is the potential for inefficient management of information. The framework must thus use a technology that optimizes access to the parameters. Another challenge is that the configuration should be distributed to foster site autonomy and maintainability, but,

on the other hand, should be easy to manage from anywhere within the grid and resilient to concurrent management attempts. For example, in order to describe the configuration of different groups of resources and services at a site, the system should allow the organization of the information into a single repository, central to the site, as well as into multiple repositories managed by different administrators, with different access policies, et cetera. In any case, the specific organization strategy should be transparent to the information management mechanisms. In addition, it is crucial to present a consistent view of the configuration throughout the system, a task not trivial considering the distributed nature of the services and the fact that different services need to represent the configuration in different formats. For example, the resource advertisement service (see section 3.3) and the monitoring service must provide the same information regarding the resource characteristics of a certain site, even if they are physically instantiated on different machines. Moreover, in order to increase the usability of the system, the configuration framework should provide the same interface to describe the resources and services at a site, as well as to configure the individual services and software products.

In the dissertation, we will present our work on developing a system that is designed to satisfy the requirements stated above. At this time, the system has been implemented and has been used for the initial deployment of the SAM-Grid.

In order to address the concerns relative to the large parameter space, we have chosen to represent the system configuration in XML format. The primary reason for this choice is the fact that XML is a language that naturally expresses context-based structures, thus easily allowing the definition of name spaces for the parameters. At the same time, available to the community are a set of querying and transformation languages, such as XPath, XUpdate, XSLT and XQuery, which address the problem of efficiently accessing and manipulating XML documents. The challenge of allowing remote and concurrent access to the configuration is addressed by storing the XML configuration in XML databases. All of the XML databases available today support remote data management via transport mechanisms such as CORBA, XMLRPC and SOAP. Furthermore, they provide database semantics for concurrent client accesses via the use of emerging standards such as XUpdate. For the SAM-Grid deployment, the database that we have chosen is Xindice [79], an opensource software developed within the XML project of the Apache Software Foundation. Xindice is implemented as a java servlet,

that we generally run using the Tomcat Servlet Engine.

In the SAM-Grid deployment scheme, every site makes available at least one XML database. The SAM-Grid products and services installed locally are configured with the URL of the site database from which they can read their own configuration as well as the portion of the site configuration that is relevant to them. Thus, the remote accessibility of the configuration repository allows the distribution of the services within the site. In addition, through the use of the SAM-Grid advertisement service (section 3.3), the URL of each database is registered with a central information collector, hence allowing the traversal of all the grid configuration repositories from anywhere within the grid.

2.1.1 The configuration process

The configuration of the SAM-Grid services and products, as well as their hierarchy and relationship with the physical resources at a site, are mainly gathered through interviews with the system administrators. For this reason, a dedicated software tool to drive the interviews is of central importance in the configuration process. A dedicated tool guarantees a consistent integration with the configuration framework, so that, in the case of SAM-Grid, the resulting answers could be represented in XML format and stored at the site in an XML database. On the other hand, in order for this tool to be usable, it should be easily configurable and flexible enough to fit the logic of all the most relevant interviews. A way to achieve this is for each product to come with a template that drives the tool through the interview. Incidentally, it should be noted that this template could be thought of as a meta-configuration, alluding to the fact that it would be the configuration of how to gather a product or service configuration. The meta-configuration language should address specifically the problem of driving interviews, offering the developers characteristics that could be preferable to a generic language, such as Perl, Python or UNIX shell. The first minimum requirement is that it should allow the expression of simple logic, such as loops and branches in the questions. Second, it should provide facilities to determine the best default to a question, considering previous answers as well as the configuration of other services and resources. Third, the structure of the template should reflect the final configuration of the given product, a characteristic useful to ease maintenance.

While the literature on system configuration languages is abundant [80,

81, 82], the study of meta-configuration languages is far from being fully explored. At this time, the SAM-Grid configuration framework uses a prototype tool developed by the team, called the meta-configurator, which is a first attempt to address the concerns expressed above. The template consists of an XML document, where the tags and attributes define the name of the configuration parameters. The logic of the interview is defined at each tag with special attributes, which are interpreted as directives to the meta-configurator. In the dissertation, we will thoroughly describe the meta-configuration language presenting examples and commenting on our experience using the tool.

2.1.2 Site and Product configuration

The SAM-Grid organizes the configuration of its services and resources, as well as their mutual relationships, breaking it down in sub-domains, corresponding approximately to the participating sites and institutions. Although other breakdown structures are, in principle, also possible, ranging from a configuration model where each service and resource is independent to a completely central model, the site-centric approach follows naturally considering the importance of site autonomy. Hence, in this context, the boundaries of a site correspond to the ownership boundaries of the resources. On the other hand, defining all the configuration parameters of a site in a single document becomes verbose and soon difficult to manage. Therefore, a modular approach is preferable, since many parameters are only meaningful to the service that uses them. In other words, it is beneficial to organize the information in a single site description that outlines the relationships between services and resources, while the internal details of their configuration are deferred to different units. As it is possible to organize the configuration of the whole grid in different sub-domains, similarly there are multiple ways of defining such units. Possible models range from grouping the information relative to a service in a single document, to producing a configuration structure for every tool and program that composes the service. Then again, software is typically organized in products, which group together tools and programs that cooperate to accomplish a well defined set of tasks. Thus, we have decided to organize the configuration of the SAM-Grid services according to their natural breakdown in software products.

In this research proposal, we argue that in order to promote the maintainability of the system, both the site and the product configuration should

be manageable using a single set of tools. This toolkit should provide an interface to the configuration framework, which other programs, such as the meta-configurator, could use. Moreover, although it should use XML to represent configurations and XML Databases to implement persistency and remote accessibility, it should provide robustness in case of occasional network access problems, in order for the SAM-Grid not to trade accessibility for reliability. On the other hand, such a system faces the nontrivial problem of maintaining the synchrony between a local and remote database. In the dissertation, we will present in detail the SAM-Grid configuration tools.

Despite the benefits of treating site and product configurations symmetrically, their usage by the rest of the system poses a different set of constraints on them and their management infrastructure. In fact, while it is crucial for the site configuration to have unambiguous parameter names, organized in a well thought out schema that directly represents the grid view of the relationship among services and resources, such care is not always necessary in the definition of internal configuration parameters of software products. Conversely, a generic tool that manages product configurations needs to be able to interface to different product management systems, such as PACMAN [83], Linux RPM, UPS (Unix Product Support) [84]: this is a requirement that simply does not apply to a site configurator. In the dissertation, we will discuss how the product configuration is organized and how the configuration tool is interfaced to the UPS product management system. We will also show the mechanism used to eliminate the configuration process during upgrades, in those cases where this is possible.

For what concerns the site configuration, today the grid computing community is still investigating how to describe the services and resources at a site. This description is used by other services to represent and interact with the sites and affects the design of grid services, such as monitoring or brokering, as well as other basic infrastructures, such as the job description language. One of the most interesting ongoing studies of the subject is represented by the GLUE schema [85], which will be adopted by the Large Hadron Collider (CERN, Geneva, Switzerland) Computing Grid Project (LCG) [37]. The GLUE schema describes the entities at a site at a very fine-grain level, using UML notation to represent the relationship among them. The entities considered not only include high-level grid services and standard resources, such as gateway nodes, storage elements or computing clusters, but also traditionally lower-level resources and their characteristics, such as single nodes in a cluster, their memory, local disk size, processor speed, et cetera.

While the studies related to the GLUE schema are of extreme practical interest, we argue in this research that its level of detail may not be necessary to describe a grid site. We believe, in fact, that the most interesting information for users and other grid services is of aggregate nature. Of course, there is a minimal level of detail that the grid infrastructure must be able to provide, especially when dealing with characteristics of principal importance to the user, such as job status or cluster utilization. Despite the arguable usefulness of these local details, we believe that there is little incentive today for the system administrators to keep this information up to date. In fact, maintenance and trouble shooting are generally operated using site-specific tools, which today give a variety of diverse views of the services and resources, views not necessarily compatible with the details sought by these grid schemas. Also, automating the collection of this type of information implies the installation of servers at the worker nodes of a cluster, a practice that in general is contrary to the paradigm of distributed ownership of the resources. Then again, studies of the type of the GLUE schema may lead to the definition of standard quantities of interest, which eventually could be gathered with site-specific implementation adhering to the standards agreed upon by the community.

In the dissertation, we will present the site description of the SAM-Grid. This site description is aggregate in nature and the entities considered are organized in a simple hierarchical structure. The site configuration is managed through the SAM-Grid configuration framework, in a uniform way with respect to the configuration of the products.

2.2 The Monitoring and Logging Infrastructures

The monitoring, logging and configuration infrastructures are the three aspects of the SAM-Grid information management component. The monitoring infrastructure captures mostly dynamical information relative to main entities of the grid, such as services, resources, jobs, et cetera. On the other hand, the historical information relative to the same entities is recorded by the logging infrastructure. Both of these infrastructures are discussed in the present chapter.

The monitoring infrastructure of the SAM-Grid mainly implements two different approaches to access information. In the first approach, events relevant to the entities of the grid are published to the information repositories, hence adopting a “push” model. This approach is best suitable to record a

change of state of these entities, independently from the external interest on the information at the time. The information gathered with this model is generally maintained persistently, as it is always significant with respect to some entity, and it is often used by the logging infrastructure as well. The second approach, instead, consists in gathering information upon request, adopting a “pull” model. In this model, the monitoring services maintain the information in transient caches, where the knowledge is built incrementally when needed. This information is generally not maintained persistently, since its gathering is prompted by the interest in the status of the entity at the time and not by some relevant event occurring within the entity itself. For both push and pull models, the information repositories must be distributed and it must be easy to compose information coming from different repositories.

It should be noted that other information gathering models are also possible, for example most notably the “heart beat”. In this model, information is gathered periodically and recorded persistently. The drawbacks of this model are that, for logging, it does not necessarily record relevant statuses of the system, and, for monitoring, it shows by design stale information. On the other hand, this model is ideal to take a series of measurements for quantities that vary slowly with respect to the sampling rate. It is also straightforward to implement and the SAM-Grid uses it to monitor the stability of some data handling services, using a tool called SAM TV [86].

Various technologies have been developed to address the problem of monitoring on the grid. The solution provided by the Globus Toolkit is the Monitoring and Discovery Service (MDS) [87, 88, 89, 98]. The Condor team has developed a product called Hawkeye [90], while the European Data Grid and, now, LCG use RGMA [74]. Zhang, Freschk and Schopf compare the three technologies for scalability and performance [91]. Another popular monitoring system in the grid community is MonALISA [92], based on Java and Jini [93] technologies. Plale, Dinda and Laszewski compare the performance of hierarchical versus flat table organization of the information [94]. Plale also compares the performance of a MySQL relational database [95] versus the Xindice XML Database [79] for a set of standard database operations [96]. Incidentally, Xindice is the technology chosen by the SAM-Grid for its XML Databases.

The technology used by the SAM-Grid for its pull model monitoring [97] is the Globus MDS, complemented by the information collector of the resource selection service (see section 3.3). The core of MDS is a Lightweight

Directory Access Protocol (LDAP) server [99, 100]. The information is gathered by information repositories (GRIS) and it is organized in trees, where each node has a unique identity code. The uniqueness of the nodes facilitates the composition of information trees coming from different repositories. The composition of information is also facilitated by a registration service (GIIS), used by the information repositories. In addition, the organization in a tree structure is well suited for a monitoring system where knowledge is built incrementally, a typical pattern when the information is explored by a human being. The information for each node is gathered by launching executables called information providers. The information providers are launched when MDS is queried for the information contained in a certain node. The information in the node is then maintained in transient caches in order to optimize access time and server load [91].

The SAM-Grid deploys an MDS per site. The information monitored via MDS tends to be aggregate in nature and reflects the status of distributed services at the site, such as the SAM data handling services. The configuration of MDS is derived from the SAM-Grid site configuration, by transforming the XML representation of the site resources to the LDAP Data Interchange Format (LDIF) [101]. This translation mechanism provides for a consistent view of the resources and services at the site, irrespectively of the service that publishes the information (section 2.1). A full account of how MDS is used to monitor the SAM-Grid services is described in the dissertation.

For the push model monitoring, the SAM-Grid uses the same XML database infrastructure that is used by the configuration framework. As for the configuration infrastructure, the SAM-Grid has developed a library that facilitates the insertion and the update of information to the database. This event-driven monitoring infrastructure is currently used to log the statuses associated with the user jobs [102]. A job submitted to the SAM-Grid is decomposed in an appropriate number of parallel instances at the remote execution cluster and submitted to the local job scheduler (see section 4.1). Being able to monitor and to log the complex status of the job, i.e. at the granularity of the single job instance running at a certain node of a cluster, is of crucial importance.

To present a uniform view of the job status, we need a representation independent from the status given by the particular type of local job scheduler. The schema representing the job status should also be flexible enough, to allow the addition of extra information, as our understanding of the associated relevant metrics evolves with time. Moreover, in the spirit of the grid,

we promote a distributed logging architecture, where every local monitoring service is remotely accessible. These considerations are a strong incentive for the reuse of the XML databases deployed at each site. In addition, the hierarchical structure of the jobs in the SAM-Grid is naturally represented in XML format. The details of the information logged with this mechanism are described in the dissertation.

The information saved in the SAM-Grid logging infrastructure can be decomposed in three major categories. First, the data processing history, which consists of information of fundamental importance for the reproducibility of the scientific results. This information is recorded in the SAM relation database as a side effect of using the data handling services. It includes an identifier of the processing application, as well as the dataset processed with all its relevant metadata. The dissertation will talk about the SAM database, even if it is not the focus of the present work. Second, the history of the statuses of main entities of the grid, such as services and jobs. This information can be used to study the system, but do not participate in the accounting of the scientific discoveries. The complex job status logging information is part of this category as are a series of SAM tools [103, 86]. Third, the debugging messages logged by each service on the grid. This information is of interest to developers only and in case some error condition occurs. The infrastructure consists of a set of distributed logging servers, capable of logging unstructured messages. The information transfer is unreliable but not blocking by design, using the UDP network protocol. This system could be improved by making the logging servers more easily distributed and allowing the handling of structured method [104].

The SAM-Grid did not adopt a uniform solution to address all the categories of information, mainly for historical reasons. The SAM team is looking into the adoption of a more integrated solution, similar to the LCG rGMA system [74] at this time. On the other hand, the system allows a semi-uniform access to the information for the users by the use of various web interfaces. For the SAM-Grid monitoring system, the web site [105] uses a set of PHP [106] scripts that present a consistent hierarchical extensible view of the whole system. This development is the result of a collaboration with NorduGrid [39], who shared with us their initial implementation of their web monitoring pages. The content of the SAM-Grid monitoring web site is elaborated in the dissertation.

3 The Job Management Component

Modern high energy physics experiments, such as DZero and CDF, typically acquire more than one TB of data per day and move even ten times as much. To give an example, during the past year the SAM system has stored 400 TB of data for DZero alone. However, this much data is not only a challenge for storage and data handling, but for the data processing as well, which must make use of the computational resources of all the participating sites, in order not to be the limiting factor in achieving the goals of the physics program.

Aside from the stream of data from the detector, various other computing activities contribute to the one TB of data stored per day. Three of the most typical activities are data filtering, also called data “reconstruction”, the production of simulated events, and data analysis. This third activity broadly consists of the selection and the statistical study of particles with certain characteristics, with the goal of achieving physics measurements. In addition, it should be noted that the first two activities are indispensable for the third one. During data reconstruction, the binary format of events from the detector is transformed into a format that more easily maps to abstract physics concepts, such as particle tracks, charge, spin, et cetera. The original format, called in jargon “raw”, is instead very closely dependent on the hardware layout of the detector, in order to guarantee the performance of the data acquisition system, and is not suitable for data analysis. On the other hand, the production of simulated events, also called “montecarlo” production, is necessary to understand the characteristics of the detector either related to the hardware, such as the particle detection efficiency, or to physics phenomena, such as signal to background discrimination.

These three typical activities, which ultimately correspond to software application families, differ among themselves principally, but not uniquely, by the usage of the computing resources. The communities that run the software range from a handful of almost dedicated experts in the case of reconstruction and montecarlo activities, to potentially the whole physics community in the case of data analysis. The typical duration of a single reconstruction or montecarlo job is dozens of hours, while data analysis ranges from a few minutes to days, depending on the problem studied. All the activities are CPU intensive, but while both reconstruction and analysis are highly I/O intensive, montecarlo is not. In fact, montecarlo almost never requires any input data, while for reconstruction and, especially, analysis the input ranges

Activity	Description	Community	Load	time/job
Reconstruction	data filtering	Small	CPU & I/O	10 hours
Montecarlo	data simulation	Small	CPU	10 hours
Analysis	data mining	Large	CPU & I/O	hours to days

Activity	Input/Job	Output/Job	Input/Year	Output/Year
Reconstruction	GB	GB	100 TB	100 TB
Montecarlo	None	10 GB	None	TB
Analysis	100 GB	GB	varies	varies

Table 1: Comparison of different characteristics among three typical computation activities of the DZero experiment. The bottom table focuses on the input/output data size. The numbers represent the order of magnitude.

from a GB to hundreds of GB. In addition, while the data access pattern of reconstruction is highly predictable, since all the “raw” data have to be filtered a few times throughout the lifetime of the experiment, the data access patterns of data analysis varies widely, as a few datasets can be accessed over and over again, while others may be almost neglected. All three activities can be run trivially in parallel because of the independent nature of particle physics events. On the other hand, while montecarlo and reconstruction are purely “batch” activities, analysis is run in both interactive and “batch” modes.

Table 1 summarizes the order of magnitudes of different characteristics of reconstruction, montecarlo and data analysis for the DZero experiment.

Various requirements can be identified for a job management infrastructure capable of running these three types of application families. The problem that we propose to solve with this research activity is implementing a job-handling system that can run applications in batch mode, addressing the requirements formulated below.

1. This infrastructure must foster site autonomy, as to satisfy the grid paradigm of distributed ownership of the resources. This means that the grid must not impose any specific choice of local fabric management systems, such as local schedulers, intra-cluster data distribution mechanisms, or monitoring tools. In particular, the grid should not impose the presence of any daemon running on the worker nodes of the

local batch system.

2. It should not require continuous network connectivity with the machine from which the user manages the job. In other words, the user should not need to have login access to special job management machines, but rather the system should provide some lightweight user interface to a core of highly available job management services. This user interface software, in principle, should be able to run on the user's laptop.
3. The job management infrastructure should be reliable, handling the job instance persistently, and guaranteeing the retrieval of any output and/or errors of the application and the middleware. Incidentally, this feature is also of fundamental importance during the development of the infrastructure itself. This is in fact the time when errors and log files are studied with particular care.
4. The system should provide an automatic resource selection service. This service, sometimes called "broker", should analyze the job requirements and select the best resource available for the job, according to some dynamically configurable algorithm. Given the complexity and dynamic nature of a grid, though, the system should be able to react to suboptimal decisions of the broker and consider the resource selection process mainly as a "recommendation".
5. The system should implement some form of fault tolerance, especially to temporary disruption of service. During job submission, for example, there should be a mechanism for automatic resubmission to the same resource, with capabilities of asking the broker to select a different one in case of prolonged unavailability.
6. It should be possible to automatically execute jobs that expose to the grid their internal interdependencies. We call this type of jobs "structured", as opposed to "unstructured" jobs, which are treated as atomic processing units. The job management infrastructure should execute the atomic jobs composing a structured job in the right order, possibly relying on the data handling system to handle input and output. It should also provide some mechanism to check the success of each atomic job not only by its exit status, but also in the context of the whole structure. In addition, in case of failure, it should allow the resubmission of the jobs that failed or had not yet run.

7. The system should comply with minimum performance requirements for metrics such as the number of jobs that can be submitted per unit time or the “cost” of job submission. In this regard, we want to stress that for CDF and DZero, the former is not a concern in the case of reconstruction and montecarlo. In fact, as previously discussed, these jobs run for hours or tens of hours and are submitted by a small number of experts in a coordinated fashion. Thus, this community is willing to wait minutes for every single job submission, if this means running more consistency checks over the job request, and therefore increasing the probability of terminating the job successfully.

We address these requirements with an infrastructure composed of four major components, organized in a three-tier architecture, as shown in figure 2. The first tier consists of a thin layer of software that interfaces the user to the system. For job submission, this user interface should accept two pieces of information: first, the description of the job, specified in a high-level language meaningful to the physicists; second, and optionally, the set of all the files necessary to run the application i.e. executables, libraries, configurations, et cetera. After submission, the user interface should assign the job a unique identifier, usable as a handle for any further management. Aside from when managing the jobs, the user interface should not mandate network connectivity in order to comply with the above requirements. The envisioned multiplicity of this tier is up to one per user i.e. up to hundreds of instances. The proposed functionalities and implementation of this software layer is described in detail in section 3.1. The second tier maintains a persistent queue of grid jobs and, being interfaced with the first tier, acts as a mediator between the user and the job instance. Because this tier acts on behalf of the user, submitting the job to a resource capable of executing it, it is also called the “submission” tier. The envisioned multiplicity of this tier is of a few instances per nation, or, in other words, a few dozens throughout the grid. The submission tier is described in detail in section 3.2. The third tier consists of the sites that ultimately run the jobs. These sites must provide computing and storage resources, tied together by a set of local and grid services. In jargon, these local resources are sometimes called the grid “fabric”. The fabric management and execution site grid services are described in chapter 4. Execution sites advertise their characteristics to a Resource Selector service, including such information as computing cluster availability and their gateway entry point, the reference to the local data

handling services, the local monitoring system URL, et cetera. The Resource Selector acts as the glue of the job-handling infrastructure, recommending to the submission sites what resource best matches the requirements of each job. Today this service is centralized. Nevertheless, should scalability problems arise, the service could be easily decentralized. The resource selector is described in details in the section 3.3.

The SAM-Grid implements this job-handling architecture using Condor-G as the underlying middleware. We now discuss what modifications have been made to Condor-G in order for it to manage large high energy physics applications on the grid. The details of how the Condor framework was integrated with the SAM-Grid will also be presented.

3.1 The SAM-Grid Client

The SAM-Grid client software implements the user interface to the collective job management services of the grid. Since the goal of the SAM-Grid project is to address the distributed computing needs of the DZero and CDF experiments, the language used to interact with the grid via this user interface must be tailored for high energy physicists. This means that the client software must recognize terms related to the typical activities of a high energy physics experiment: montecarlo event production, data reconstruction and data analysis. Moreover, it must provide a way for the user to specify the internal structure of the jobs related to these activities. In fact, montecarlo and reconstruction especially require multiple stages of data processing before producing data usable by the whole community. A typical example of a post processing stage is the merging of the files produced by the parallel processes involved in reconstruction and montecarlo production. Being able to declare to the user interface the dependencies among these stages and having the system automatically carry over the related tasks without human intervention significantly increases the overall computing efficiency. Furthermore, the client software must not require continuous network connectivity, thus not imposing on the grid system specially maintained client machines with user accounts for a large number of collaborators.

The client software is organized in two distinct layers. The top layer, closest to the user, exposes the interface specific to the high energy physics domain. The bottom layer, which consists of standard middleware from the Condor-G system, implements the low-level job management mechanisms. In particular, for job submission, the top layer corresponds to the SAM-Grid

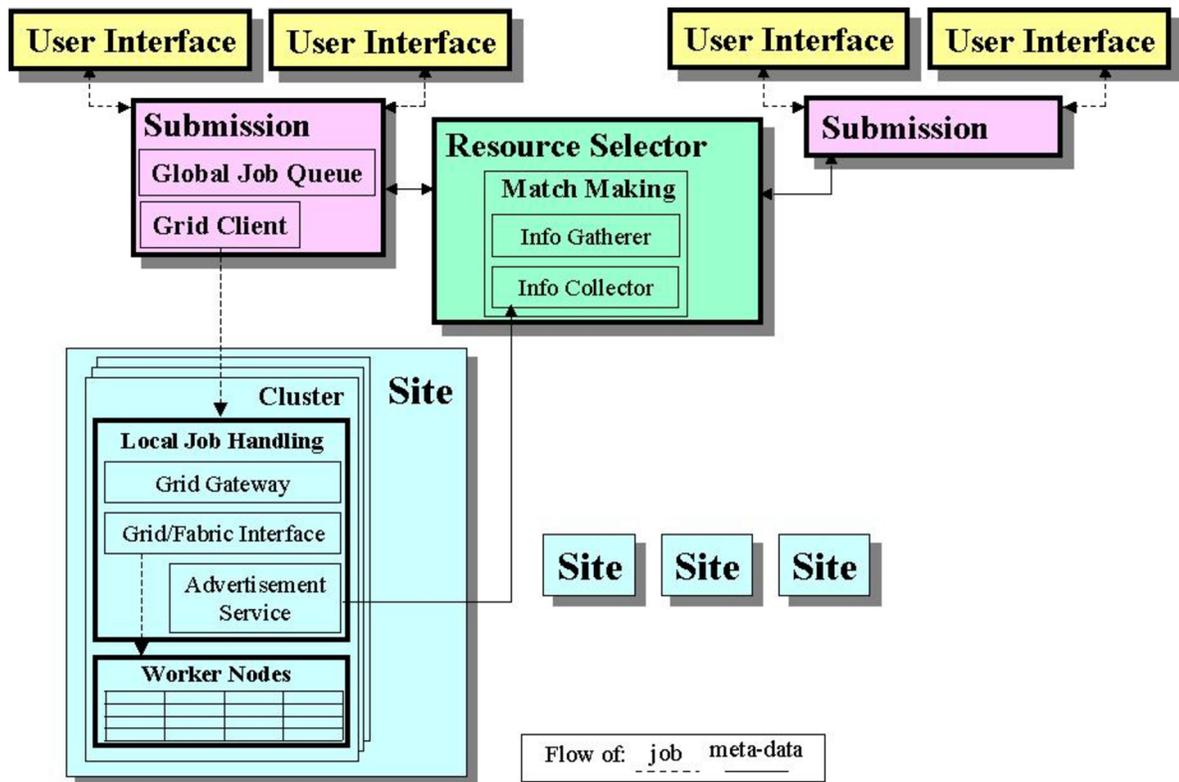


Figure 2: The SAM-Grid job submission management infrastructure is based on a three-tier architecture. The first tier is the user interface (top), a thin layer of software used to manage the jobs. The second tier is the submission site, a suite of services that maintain the queue of grid jobs, mediate the interaction between the users and the remote resources and can submit jobs on behalf of the user. The third tier is the execution site (bottom), where the jobs are run on the available resources (the figure shows only local job handling resources and services, for clarity). The resource selector collects the resource characteristics advertised by each execution site, and assists the submission site in deciding where to run each job.

Job Description Language (JDL) interpreter and the bottom layer to the native Condor-G job dispatcher. In the rest of the section we discuss the JDL interpreter.

The main responsibility of the JDL interpreter is to translate the description of the job provided by the user into a low-level set of directives to the services of the grid. In addition, the interpreter is a natural place to check the consistency of the parameters presented by the user: this is important because the typical latencies of a grid imply that trivial, application specific, mistakes in the value of the parameters are detected by the infrastructure hours after the job submission and they generally result in unrecoverable failures. The consistency checks are essential to implement a fast fail submission mechanism.

The interpreter translates the user job description into a set of directives, which can be categorized based on what grid services they affect:

- Directives sent to the interpreter itself: these have an effect in particular on the creation of the software archive containing the user executable, libraries and configuration files. In jargon, this archive is called the “user input sandbox”. It should be noted that in the SAM-Grid model the user sandbox is, in principle, only part of the software required to run the overall application. The rest of the software is retrieved via the data handling system and dynamically deployed by the sandboxing fabric services (chap. 4).
- Directives sent to the Resource Selection Service: these affect the logic used by the resource selector to associate a job with an execution site. An example of a typical logic used for data analysis gives priority to those sites that have already cached the highest percentage of the data requested by the job. Another typical logic is random site selection. We propose to study and implement site ranking heuristics that optimize various metrics, such as minimal time to job completion or minimal bandwidth, for the three types of high energy physics applications. See section 3.3 for more details.
- Directives sent to the submission site services: most notably, these configure the logic for resubmitting/rematching jobs in case of failure. It should be noted that not all the jobs can safely be automatically resubmitted (section 3.2). Another example of a typical directive in this

category is the configuration of the email address to which notification should be sent after job completion.

- Directives affecting the fabric services at the execution site: these are aimed at recreating the job execution environment. In particular, they can program the dynamic product installation mechanism, declaring specific products as necessary to the job (section 4.2).
- Directives to the data handling system: these include, for example, what dataset the job requests or what physics group is accountable for the “cost” of the data caching.

The design and implementation of the interpreter, as well as the relevant parts of the SAM-Grid JDL, are discussed in the dissertation. At this time, SAM-Grid is supporting Montecarlo and Merging job types. Other job types, such as Analysis and “Structured” jobs, are in the prototypical phase, while others, such as Reconstruction jobs, are under development. In the dissertation, we will provide a full solution for the prototypical job types and compare the SAM-Grid JDL with other grid JDL [107, 108]. We summarize below the main problems that this involves.

The requirements of the data analysis job type are more stringent than the montecarlo type, since it involves many more users accessing data in unpredictable patterns. We foresee potential scalability problems in the middleware, in particular at the gateway machine of the execution sites. The GRAM protocol, in fact, instantiates a process for every grid job submitted to the site. If the number of grid jobs running concurrently reaches the hundreds, the load of such processes may surpass the processing power of the typical gateway machine. Another problem is the local monitoring infrastructure (section 2.2). It is based on an XML database, a technology known to still have problems in dealing with documents larger than 1 MB and with the transaction management of hundreds of concurrent updates. The solution that we plan to investigate for both problems consists in limiting the number of jobs running at the execution sites, by throttling the flow of jobs from the submission sites. In the SAM-Grid architecture, this can be achieved by developing Resource Selection Service algorithms that are aware of the maximum number of jobs acceptable by each site. These limits can be made available to the Resource Selection Service by the sites themselves using the Advertisement service. A similar solution has been adopted within the context of the Condor-G framework to do resource load balancing in a

BioInformatics grid and it was demonstrated at the Super Computing 2003 Conference, Baltimore, by the Condor Team [109]. That grid consisted of half a dozen sites, each available to queue up to two dozens jobs. Even if no scalability limit was hit in that context, the demonstration was nevertheless a proof of principle of the functionality of the framework to deal with this types of problems.

Structured jobs manage a combination of other job types. Structured jobs capabilities have been requested by the community since the beginning of our work on the SAM-Grid. The ability for the infrastructure to automatically submit dependent jobs, checking at each stage the success of the dependencies before submitting new tasks, is appealing, mainly because it virtually eliminates the dead times that occur between job submissions when a human operator is involved. Structured jobs also allow organizing in a single processing unit a potentially complex task, thus reducing the errors that an operator could make when managing the task step by step through days of processing. Other groups interested in high energy physics make also use of structured jobs [42, 110]. It is interesting to note that these groups, as well as the SAM-Grid, have implemented structured jobs using the same middleware, the Direct Acyclic Graph job Manager (DAGMan) [111]. DAGMan is a workflow management layer on top of the Condor middleware. The dependencies among the jobs can be declared to DAGMan, so that it can compute the order of job execution. In case some jobs fail, DAGMan automatically computes a “recovery” DAG, to allow the reprocessing of the failed jobs and its dependencies only. In the dissertation, we will discuss in detail how the SAM-Grid uses DAGMan to implement structured jobs and how DAGMan could be improved to be more usable by large grid infrastructures.

3.2 The Job Submission Service

The job submission service is the second tier of the job management component, interacting with both the client software at the first tier and the execution site services at the third tier. This component has three main responsibilities in the overall architecture.

1. It maintains the queue of the grid jobs, holding the job description as well as the user input sandbox. The information is stored persistently and the status of the service and of the jobs can be recovered after a shutdown.

2. It interacts with the Resource Selection service, providing upon request the description of the pending jobs and receiving a recommendation on where each job should be submitted (see section 3.3).
3. It is a mediator of the interactions between the user and the remote resources for the management of the jobs, when the user is online. Conversely, it acts on behalf of the user, when the user is offline. In particular, the submission service maintains up to date the status of the job, resubmits the job to the grid resources in case of failure, and receives the output sandbox from the job upon completion.

In the SAM-Grid model, the output sandbox consists of “small” user files and meta-processing information, such as the standard output and error streams and the application log files. In other words, the output that is of little interest to the community as a whole and does not need to be catalogued. On the other hand, the main results of the computation, often many GB in size, are catalogued and handed over to the data handling system. The information in the output sandbox is of fundamental importance for troubleshooting the system as well as the application, despite the fact that it is not permanently stored nor catalogued. The user can download the sandbox from the submission site via the web. If the output is not retrieved, it is deleted after a week, according to the typical submission site policy. Another popular model of output sandbox management requires that the users specify a location to store the job output. This model is adopted with some variations by LCG as well as by the CDF distributed Analysis Farms (CAF) [112]. In the case of CAF, the user is responsible to check that the chosen storage accepts the user credentials and that it has enough space to store the output. If these conditions are not true, generally the system buffers the output for some time at the execution site. In practice, today the CAF system is fully integrated only with kerberos, instead of more common grid standards, such as the Globus Security Infrastructure, and the storage system of choice is usually the user’s home area. Within these restrictions, though, most users consider that the convenience of finding the output directly in the user’s home area overcomes the responsibilities that the model imposes to them. On the other hand, we believe that the SAM-Grid model offers a reasonable trade off between user convenience and responsibilities, as it requires the user to download the output, but does not require the users to be responsible for the availability of the storage system.

Another open problem for the SAM-Grid is the integration of a credential management system. The Globus security infrastructure allows the creation of short-lived proxies for long-lived identity certificates. This mechanism minimizes the risk of long-lived credentials theft, as the short-lived proxies can always be used in place of the long-lived credentials for the user authentication. On the other hand, it is not always clear how long the proxy credential should last. In general, a job needs a proxy to access resources on behalf of the user, but the lifetime of a job spans from a few hours to several days. Typically, users tend to create proxies that last days, being more concerned about a delayed job not storing its results rather than about the risks of identity theft. Yet, we believe that a grid system should offer the users a convenient mechanism to overcome this problem. At the minimum, the system should allow the distribution of proxies, renewed periodically by the user, to all the services that act on behalf of the user, including the job. On the other hand, this distribution system is non trivial, since it is difficult to coordinate the propagation of the new proxies from the client to all the entities in need. In addition, requiring the user to manually renew the proxy is not a very usable mechanism. An elegant solution to this problem is integrating an online credential repository service with the grid. In this model, initially, the user can upload medium-lived proxy credentials to the repository, so that, subsequently, all the entities in need can prompt the service to issue short-lived proxies on behalf of the user. We propose to integrate the SAM-Grid with the MyProxy server [113], which is the de facto standard technology in this field. We expose hereby our plan.

The SAM-Grid components that should be integrated are the submission site service, the user jobs and the execution site job managers. A job manager is a process forked by the Globus Gatekeeper at the execution site gateway machine. The job manager is the grid interface to the local fabric services, most notably to the local job scheduler. The SAM-Grid has developed a suite of such interfaces, tailored for each of the main high energy physics computation activities (chap. 3). The job manager runs under the user identity and it uses delegated proxy credentials, which we want to renew with MyProxy. The job manager can obtain the new credentials directly from MyProxy or from the submission site service. This second option is possible because as of April 2004 MyProxy is fully integrated with the Condor framework. On the other hand, providing a consistent mechanism to renew the credentials for the job will be more difficult. In general, the credentials can be propagated using either a push or a pull model. In the push model, the

job manager uses the facilities of the local scheduler to update the credentials of the job running at a worker node. In the pull model, a process started at the worker node monitors the lifetime of the local credentials and invokes MyProxy upon their expiration. The disadvantage of the push model is that not all the local schedulers can move files to the job environment after job submission. Thus, this model would limit the types of local schedulers to which the SAM-Grid could interface. The disadvantage of the pull model, on the other hand, is that firewalls may block the communication with the MyProxy server, hence incapacitating the credential renewal mechanism. We propose to further investigate both approaches and implement a solution that will best serve our system.

Another interesting problem consists in developing the logic for the submission site to resubmit jobs on behalf of the user, in case of failure. As already mentioned, the instructions of the resource selector to the submission service must be considered as recommendations only. The resource selector, in fact, may not know of local problems at the resources or may have stale information (section 3.3). Developing algorithms to react to the unavailability of grid resources can improve the fault tolerance of the whole system. On the other hand, the resubmission logic should be tuned to recognize different types of faults. For example, the submission service should try to resubmit a job in case a resource has rejected it because of local policies, internal problems, or a temporary disruption of service. On the other hand, it should not try to resubmit a job whose delegated credentials are expired, as it would only generate traffic without any chance of success. As of today, the SAM-Grid team has only experimented with some very simple resubmission algorithms and has decided to disable this function until a more complete study on the matter can be done. We propose to complete this study by categorizing the types of faults and implementing in each case an automatic reaction to deal with them.

3.3 The Resource Selection Service

Like most of the grid systems deployed today, the SAM-Grid offers an automatic resource selection service as part of its job and resource management component. Many groups have worked on the problem of resource management. Buyya, Chapin and DiNucci [114] study different architectures for resource management, comparing models that organize resources and services in a hierarchy with models inspired by the economic principles. A more

comprehensive work is proposed by Krauter, Buyya and Maheswaran [115], who present a taxonomy of the resource management systems using a dozen characteristics. The three major types of Grids identified are Computational Grids, Service Grids and Data Grids. In a Computational Grid, applications are executed in parallel on multiple machines as if the aggregate system were a supercomputer. Storage management is provided via specialized infrastructures. A Service Grid is a system that provides aggregate services that are not provided by a single machine. Typical examples are systems that connect users and applications into collaborative workgroups and infrastructures for realtime multimedia applications. A Data Grid integrates grid level data management services with computing resource management services. Its focus is generating new information by processing data from distributed repositories. The SAM-Grid is an example of a Data Grid.

For computational grids, the literature on resource selection and scheduling algorithms is conspicuous [116, 117, 118, 119, 120]. Fujimoto and Hagi-hara [121] measure the performance of a few classical algorithms and compare their relative performance. Buyya, Stockinger, Giddy and Abramson [122] concentrate on economical scheduling models.

In addition to the theoretical work, various groups have implemented resource selection middleware for computational grids [29, 32, 123, 124, 125], and for hybrid computational and service grids [126, 127, 128, 28]. Buyya compares these technologies in his Ph.D. thesis [129], investigating the advantages of an economical model using the NimrodG system.

Because of their higher level of complexity, the research on data grids is not as advanced as for computational grids. In particular, the SAM-Grid has the unique opportunity of implementing and measuring the performance of scheduling algorithms for data grids, since it deploys an infrastructure used to run real high energy physics job. The different algorithms can be implemented within the framework of the resource selection service.

In the case of the SAM-Grid, the resource selection service gives recommendations to the submission sites on how to match jobs to execution sites. In other words, in the architecture of the job management component, this service multiplexes the jobs queued at the second tier to the third tier. Thus, having knowledge of both these tiers, the Resource Selection service is also an invaluable hub of information for the whole grid, making it suitable as a component of naming and monitoring services.

As for the submission and client site software, the SAM-Grid implementation of the Resource Selection service is based on the condor middleware.

In fact, one of the outcomes of the collaboration between the Condor and the SAM-Grid teams was enhancing the Condor match making service [130] to select grid resources [131]. To achieve this goal, resources and job-related services register with this matchmaker, exposing characteristics of their interfaces and internal statuses. For example, submission site services advertise their current address, so that other entities can manage jobs or acquire details on the job queue. Execution sites, instead, advertise attributes such as their grid entry point, the URL of the local XML database (section 2.1), the name of the local SAM data handling services, the address of the SAM naming service, and other aggregate information on the cluster (see also chap. 2). It is worth noting that while the technology used to describe resources and services is the Condor Classad [132], most of the syntax and the semantics of the description are specific to the SAM-Grid. In order to adapt to the dynamism of a grid system, the match making service treats the incoming information as a soft-registration. This means that the registration is automatically discarded after a configurable amount of time, thus reducing the impact of stale information. It should be noted that this does not prevent stale information from being in the system. In fact, this is one of the reasons why the resource selection should be considered as a “recommendation” only.

The resource advertisement service uses the site configuration description as input (section 2.1.2). All the resources and services at a site are organized in a hierarchical structure, represented in XML and stored in the site’s XML database. The advertisement service is responsible for decomposing this hierarchy in a set of flat descriptions, represented in the form of Classads. Each Classad aggregates a series of services and resources at the site, which, together, are used to run grid jobs. The content of the Classads is determined by policies, presented to the advertisement service in the form of plug-ins that use XQuery to manipulate the XML description of the site.

The resource selection is processed in match making cycles. Periodically, every submission site is queried about the details of the jobs queued therein. For every job, resources are initially skimmed by looking at the job requirements. The remaining matching candidates are then ranked according to various configurable algorithms. When the “best” match is selected, the job description is enriched with information from the resource. The job description, in fact, can refer to resource attributes using specially named variables. The foremost attribute used in this process is the grid entry point, or Globus URL. This URL is used by the submission site to submit and manage the jobs at the resource. This attribute is now a standard of the Condor middleware

and was one of the modifications introduced to the Condor match making service as part of the collaboration of the Condor team with the SAM-Grid. Other attributes from the resource description, such as the XML database URL or the name of the SAM station, are used to build the execution environment of the job.

It is recommendable to limit the amount of information contained in the registration Classad to a few dozen attributes. The use of this technology for much larger data volumes, in fact, has not been fully investigated, but it is likely to lead to an inefficient match making process. On the other hand, it should be possible to base the resource selection on criteria that require a large amount of information, such that it would hardly fit into a Classad because of the size limits stated above. For example, a popular criterion to match analysis jobs is selecting the resource that has already in its storage elements most of the data requested by the job. The amount of data at every site may easily reach tens of thousands of files, making thus impractical sending the information to the match making service for every execution site, and, in particular, sending it via a Classad. To overcome this problem, the resource selector of Condor has been modified to call externally provided functions when evaluating a match. Both the job and the resource descriptions can define any attribute of their Classad using these functions, passing to them as arguments any other attributes of the two Classads.

Using this mechanism, the SAM-Grid is able to rank the matches for analysis jobs according to the amount of data cached at the execution site. This feature, together with the ability for SAM to pre-stage the data as soon as the job enters the site, is a prototypical solution to the problem of job and data colocation. In the dissertation, we will describe the details of this mechanism.

We believe that this approach to job brokering is very promising. The modification to the Condor resource selector is now part of the standard middleware, which can be used in different contexts by different groups. This effort promotes the development of a modular resource selector, which implements matching algorithms that plug into a thin middleware layer, potentially common to several grids.

Within the context of this research, we propose to study brokering criteria specific to the typical high energy physics computing activities. In particular, we are initially interested in optimizing the selection of resources to produce montecarlo events. This type of job is computing intensive with little requirements on I/O (see Table 1). Therefore, the prototypical criterion for job and

data colocation cannot be applied and new solutions must be studied. Another longer-term study is the selection of resources for reconstruction jobs, which are both I/O and CPU intensive. Today, the SAM-Grid offers only a random selection criterion, while it would be nice to have at least round-robin and load-balancing criteria. This task involves the study of what information needs to be advertised by the resources as well as the implementation of the ranking function. Matching criteria for other types of computing activities will be contingent on the request by the experimenters.

Other groups have already studied interesting solutions to the load balancing problem in the context of LCG. In particular, researchers from the Leiden University and NIKHEF, Holland, have approached the problem by estimating how long a job would stay idle in the queue of the local scheduler at a site [133]. A statistical model produces this estimate using a simulator of the local scheduler and the status and history of the local job queue. This estimate is prompted by the LCG resource broker, in order to select the least busy resource.

4 The Execution Site

The modern computing infrastructures of a high energy physics experiment consist of resources pooled together by a set of grid services. Because the resources at a site are owned and managed by the collaborating institutions, the scope of the grid services is limited to the granularity of the site. Within each site, a different set of services is responsible to coordinate the local resources and execute the jobs that are coming from the grid. The local services and resources at these execution sites are called the “Fabric”.

This separation of responsibilities between fabric and grid services functions because of an intermediate layer that acts as an active interface between them. This interface has two main responsibilities. First, it adapts the input and the output between the grid and the fabric in order to comply with the specifics of the fabric. Second, it coordinates the usage of the local resources according to the specification of the grid jobs and the local policies. For example, when data intensive jobs enter the site, the interface should be responsible to trigger the local data handling services to enable the pre-staging of the data, while the job is idle in the scheduler queue. Other examples are the splitting of a grid job in multiple local job instances or the recreation of the local job environment. It should be noted that these types of tasks

couldn't be accomplished by the jobs discovering and accessing directly the fabric services [102].

In our experience, the local fabric services and the grid/fabric interfaces do not have the same level of maturity of the grid services. In particular, the SAM-Grid project suffered because of the lack of standards for basic fabric services, such as local scratch management or local storage management. It is in fact only recently that the Storage Resource Management interface is arising as a standard [134]. The lack of standards was overcome by programming the grid/fabric interface against a set of "ideal" fabric services. We present three of these ideal services, or "idealizers":

1. the adapter to the local batch system: this idealizer is a uniform interface to "any" underlying local job scheduler.
2. the dynamic product installation: this idealizer is used to recreate the job environment. The worker nodes of a batch system, in fact, do not have installed experiment specific software, in order to foster resource sharing.
3. the local sandbox management: it is responsible for packaging and delivering within the cluster the software needed to run the job and for gathering the job's output.

At each site, these idealizers have been implemented by aggregating different physical services. It should be noted that this approach clearly encapsulates the parts of the software that are site specific.

4.1 Local Batch System Adaptation

After our first experience deploying the SAM-Grid, we concluded that the "standard" batch system interfaces implemented in the Globus Toolkit are not flexible enough to include most of the resources of DZero and CDF. Even at sites running the same batch system, we observed that different administrators frequently configure the batch system differently because of local constraints, thus requiring the local users to submit jobs using slightly different commands. In some other cases, the terms of the agreement to use the resources could be respected only by adding special attributes to the job submission request. For example, when DZero submits montecarlo production jobs at the condor cluster of the University of Wisconsin at Madison,

the job description file must include special attributes to prevent job eviction. Another example is running montecarlo jobs at the IN2P3 computing facility in Lyon, France: the BQS batch system is locally configured to let the job overcome the downtime of their local mass storage system only in the case when the job is submitted with a special option. These attributes and submission options are all nonstandard and site specific. Other grids, such as LCG [37], solve this problem by exposing to the grid the interface of the local scheduler. Conversely, our approach encapsulates this level of details within the boundaries of the fabric services.

Another point to consider, besides the site-specific peculiarities of the batch system, is that the typical high energy physics applications are seldom submitted directly to the batch system. They are rather submitted through experiment specific local interfaces that take care of the job preparation. Such preparation steps include the triggering of data handling systems such as SAM, the use of local job sandboxing mechanisms, or the decomposition of the job into smaller tasks, generally executed as parallel instances.

In order to address this concern, the SAM-Grid team has developed a series of job management scripts that use the experiment-specific interfaces. These scripts are invoked via standard grid mechanisms, such as the Globus Gatekeeper. From within these scripts, the invocation of the local batch system commands is done via an intermediate layer that abstracts the basic interactions with the batch system. This layer is configured locally with the specific commands used for job submission, look up and cancellation. The batch system adapter is also configured to interpret the output of the batch system commands, to enable the extraction of relevant information, such as the local job id after submission, the status of the job after lookup or the error messages after any command invocation.

This additional level of indirection in the configuration of the local job management has proved to be of fundamental importance during the phase of deployment. In the dissertation we will discuss in detail the advantages and disadvantages of this approach.

4.2 Dynamic Product Installation

In order to execute jobs on the grid, the code must be portable. DZero has developed tools that recreate the Run Time Environment (RTE) [135] of the typical DZero applications. These infrastructures give the users the ability of packaging their programs with all the software dependencies, thus

enabling the execution in rather "hostile" computing environments. There are certain classes of jobs, though, that use standard applications driven by user specified configuration parameters, which require little or no user-provided code. These applications can be quite large (on the order of a few Gigabytes, even when compressed) and packaging them for every job would be quite costly. These costs add up, considering that every such archive needs to be transported and temporarily stored possibly in more than one place throughout the lifetime of the job.

On the other hand, experiments maintain dedicated clusters, configured to provide access from any node to a series of experiment specific software products. When running on these clusters, users just need to provide their custom version of the code or the configuration files to run the standard applications. Administrators are responsible for installing and maintaining these products, trying to compromise between the available disk space and the large span of product versions that the users need. This model is quite expensive for the administrator and does not let the users take advantage of non-dedicated clusters that may be available to the community using RTE mechanisms.

Within the SAM-Grid, we use a hybrid model that promotes the advantages of the two approaches. Many commonly used applications, such as the montecarlo production products, which are made portable via RTE techniques as in the first model, are stored into the data handling system of the experiments. The clusters that are part of the grid are configured to provide a local data cache, managed by a SAM data handling service. Users who run jobs that use these standard applications can specify them in the Job Description File as dependencies, as in the second model. After the job has entered a cluster, the middleware is responsible for the delivery of the dependencies to the worker nodes, before passing control to the user's application. This approach has many advantages. First, products are no longer installed and maintained by the system administrator, but rather, brought into the data handling cache and installed upon request. Therefore, the size of the software provided directly by the user can be rather small. Second, since the cache is automatically managed, there is no longer a maintenance concern for the availability of older software versions at a site. Third, applications and their usage can be thoroughly catalogued using the metadata mechanisms of the SAM data handling system. Fourth, this mechanism can be used in conjunction with user-provided RTE executables, providing a high degree of flexibility in running the jobs.

4.3 Local Sandbox Management

When submitting a job to the grid, a user is required to supply a description of the characteristics of the job, such as application name, product dependencies, optional input dataset, etc., and/or an archive containing the software and configuration files needed to run the application. This archive is sometimes referred to as the input sandbox. The output of the job, such as the error and output streams, relevant log files and output files is sometimes called the output sandbox. It should be noted the SAM-Grid handles the large input and output data files and potentially the product releases via the SAM data handling service, hence dramatically reducing the size of the sandboxes, since these files do not need to be part of them.

The standard grid tools implement a protocol, GRAM [17], that allows the transport of the sandboxes and the submission and monitoring of the job. Nevertheless, there are no standard tools to manage the sandboxes at the local cluster. Ideally, the sandbox management should be able to rely on a local storage service, with a well-defined interface to, at a minimum, store, retrieve and remove input and output files from anywhere within the cluster. Even if implementations of local storage services are available [136, 67], they are not very widely deployed and generally considered nonstandard. Instead, what is generally provided at the cluster is disk space accessible from the gateway node, and some mechanism of intra-cluster communication. This is achieved at every site using a wide variety of different strategies. Typically, either nodes use a common network file system, or the batch system is configured with some form of input file stagein/output file stageout, or the nodes have access to an open network. Each of these strategies is not a general solution and each has weaknesses. For example, the typical network file systems used, NFS, has scalability problems, especially during writes; stagein/stageout often can only be triggered from special places within the cluster and at certain times only, such as e.g. from the head node at the time of submission; open networks are becoming less and less popular, considering the proliferation of site firewalls.

Considering that no standard local storage service exists today, the SAM-Grid sandbox management infrastructure, instead of trying to adapt to all possible different cluster configurations, starts up dynamically a gridftp server, hence guaranteeing a uniform intra-cluster transport mechanism. Input sandboxes coming from the grid are kept compressed at the gateway node in a disk area unique to the job. For every job, the infrastructure creates a self-

extracting archive, containing the user's delegated credentials, the gridftp client and the directives necessary for the delivery of the input sandbox and the dependent products. It then submits an appropriate number of parallel instances of the job, using the batch system adaptation mechanism described above, relying only for this first executable on the native intra-cluster transport mechanism. At the worker node, the environment of the job is recreated and the control passed to the user's application. After the execution has terminated and before cleaning up the scratch space, the custom output is packaged and transferred to the gateway node, in order for it to be bundled together with the output of all the other job instances and sent back to grid. The SAM-Grid makes this output bundle available to the user for download from the web. The details of this mechanism are thoroughly discussed in the dissertation.

5 Summary

This chapter summarizes the research topics proposed in this document. The topics are put in context in a highly condensed fashion. For more information, the reader is referred to the sections relative to each topic.

The SAM-Grid is an integrated grid system for high energy physics (HEP). It consists of data, job, and information management components. The system is used by the DZero and CDF experiments at Fermilab for part of their distributed computing needs. This dissertation focuses on the job and information management components of the SAM-Grid and addresses problems that have arisen during the phase of design, implementation, and deployment of the system (chap. 1).

The SAM-Grid information management component consists of the configuration, monitoring, and logging infrastructures. The focus on configuration management is relatively new for data grids and our solution is a distributed configuration system based on a network of XML databases (chap. 2). The information is decomposed in two categories of configurations (section 2.1.2):

1. the relationship among services and resources at a site. This site configuration is central to the main activities of the fabric and grid services. Specifying a functional and descriptive site configuration is one of the goals of our research;

2. the configuration of the software products, which implement the services. A goal of this research is providing a tool that treats the two categories of configuration uniformly.

The system uses a dedicated language, developed by the SAM-Grid team, tailored to drive interviews with the persons responsible to install the software. We will study the limitations and advantages of such a language (section 2.1.1).

The monitoring infrastructure of the SAM-Grid is based on both push and pull models, implemented on the Globus MDS and a network of XML databases. In the dissertation we will discuss the architecture and implementation of such monitoring system (section 2.2).

The research topics on information management summarized above are mainly completed.

The job management component of the SAM-Grid is organized in a three-tiers architecture: the user interface, the submission, and the execution site tiers. The resource selection service cooperates with the submission tier recommending the “best” execution site to run the jobs (chap. 3). Part of this work consisted in enhancing Condor-G so that it could be used as the SAM-Grid job management middleware. The match making service of the condor batch system was modified to match jobs with grid resources, ranking the matches according to externally provided logic. We propose to implement various brokering algorithms for HEP applications. This work should be completed by the end of the year. Using these brokering algorithms together with SAM, the data handling component, the SAM-Grid provides a solution to the problem of job and data colocation and data pre-staging. We will study what characteristics of the grid sites should be advertised to the resource selection service. The advertisement process is implemented plugging into the advertisement framework algorithms that manipulate the description of the services and resources at the site. The implementation of an extensible advertisement framework is completed and will be discussed in the dissertation (section 3.3).

In the next 3 months, the SAM-Grid will be extended to run Reconstruction and structured jobs. In particular, structured jobs are a composition of other types of HEP jobs. We will augment the SAM-Grid using the Directed Acyclic Graph Manager (DAGMan) to manage structured jobs (section 3.1).

We also propose to investigate

1. the management of security tokens using an online credential repository

service, such as MyProxy (section 3.2);

2. the study of error recovery mechanisms via the automatic resubmission/rematching of jobs to the grid sites (section 3.2).

The SAM-Grid implements a thick active interface between the grid and the fabric services at the execution sites, in order to coordinate the local resource usage. The fabric services do not have the same level of maturity of the grid services. To overcome this problem, the grid/fabric interface was implemented against a set of “ideal” fabric services. Part of our research consists in implementing these “idealizers” for various configurations of the local services: this work is mostly done (chap. 4). These fabric services include local batch system adaptation, dynamic product installation service and local sandbox management service.

A final important part of the dissertation consists in reporting to the community the “lesson learned” in deploying and operating an integrated functional grid for two running HEP experiments.

References

- [1] I. Foster, C. Kesselman, S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, *International J. Supercomputer Applications*, 15(3), 2001.
- [2] The BaBar Collab., D. Boutigny et al., “Technical Design Report”, SLAC-R-95-457.
- [3] The BELLE Collab., M. T. Cheng et al., “Technical Design Report”, KEK-Report 95-1.
- [4] The D0 Collab., “The D0 Upgrade: The Detector and its Physics”, Fermilab Pub-96/357-E.
- [5] CDF Collab., R. Blair et al., “The CDF II Detector Technical Design Report”, FERMILAB-Pub-96/390-E.
- [6] D. G. York, et al., “The Sloan Digital Sky Survey: Technical Summary”, *The Astronomical Journal* 120 (2000) 1579-1587

- [7] Bruce Allen, et. al., "Determining Upper Limits on Event Rates for Inspiralling Compact Binaries with LIGO Engineering Data", LIGO technical report T010025-00-Z (2001).
- [8] CERN: <http://www.cern.ch/>
- [9] The CMS Collaboration, "The Compact Muon Solenoid Technical Proposal", CERN/LHCC 9438, LHCC/P1 1994
- [10] The CMS Collaboration, "Computing Technical Proposal", CERN/LHCC 96-45, (Geneva 1996)
- [11] P. Saiz, L. Aphetche, P. Buncic, R. Piskac, J. E. Revsbech and V. Sego, "AliEn - ALICE environment on the GRID", Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 502, Issues 2-3 , 21 April 2003, Pages 437-440
- [12] The ALICE Collaboration, "ALICE Technical Proposal for A Large Ion Collider Experiment at the CERN LHC", CERN/LHCC/95-71, 15 December 1995
- [13] The LHCb Collaboration, "LHCb: Technical Proposal", CERN-LHCC-98-004;
- [14] The Atlas Collaboration, "Atlas - Technical Proposal", CERN/LHCC94-43, CERN, December 1994.
- [15] Global Grid Forum: <http://www.ggf.org/>
- [16] Globus Alliance: <http://www.globus.org/>
- [17] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", International Journal of Supercomputer Applications, 11(2): 115-128, 1997
- [18] A.S. Grimshaw, A. Natrajan, M.A. Humphrey and M.J. Lewis, A. Nguyen-Tuong, J.F. Karpovich, M.M. Morgan, A.J. Ferrari, "From Legion to Avaki: The Persistence of Vision", Grid Computing: Making the Global Infrastructure a Reality, eds. Fran Berman, Geoffrey Fox and Tony Hey, 2003.

- [19] Platform Computing, "PLATFORM GLOBUS TOOLKIT: Open-source, commercially supported toolkit for building grids", On line <http://www.platform.com/products/Globus/>
- [20] A. Chien, B. Calder, S. Elbert, and K. Bhatia, "Entropia: Architecture and Performance of an Enterprise Desktop Grid System", Journal of Parallel Distributed Computing, Vol 63, Issue 5, May 2003, pages 597-610.
- [21] Sun Grid Engine: <http://www.sun.com/software/gridware>
- [22] United Devices: <http://www.uniteddevices.com/>
- [23] Parabon: <http://www.parabon.com>
- [24] ProcessTree: <http://www.processtree.com/> , Distributed Science Inc, Nov. 2000.
- [25] Popular Power: <http://www.PopularPower.com/>
- [26] Mojo Nation: <http://www.mojonation.net/>
- [27] DataSynapse: <http://www.datasynapse.com/>
- [28] R. Buyya, D. Abramson, and J. Giddy, "Nimrod-G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", The 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000), May 2000, Beijing, China, IEEE Computer Society Press, USA.
- [29] F. Berman and R. Wolski, "The AppLeS Project: A Status Report, Proceedings of the 8th NEC Research Symposium", Berlin, Germany, May 1997.
- [30] J. Novotny, "The Grid Portal Development Kit", Concurrency: Practice and Experience 2000; 00:1-7
- [31] G. Allen, T. Dramlitsch, I. Foster, T. Goodale, N. Karonis, M. Rippeanu, E. Seidel, and B. Toonen, "Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus", in Proceedings of Super Computing 2001, Nov. 2001, Denver, Colorado.

- [32] J. Basney and M. Livny, "Deploying a High Throughput Computing Cluster", High Performance Cluster Computing, R. Buyya (editor). Vol. 1, Chapter 5, Prentice Hall PTR, May 1999.
- [33] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", in Proceedings of the 10th International Symposium on High Performance Distributed Computing (HPDC-10), IEEE CS Press, Aug. 2001.
- [34] I. Foster, "Grid Technologies & Applications: Architecture & Achievements", in Proceedings of Computing in High Energy and Nuclear Physics (CHEP01), Beijing, China, Sep. 2001
- [35] J. Moore, "Portals could lower grid barriers", Federal Computer Week, Oct 2003
- [36] W. Hoschek, J. Jean-Martinez, A. Samar, H. Stockinger, K. Stockinger, "Data Management in an International Data Grid Project", 1st IEEE / ACM International Workshop on Grid Computing (Grid 2000), Bangalore, India, Dec. 2000.
- [37] J. Apostolakis, G. Barrand, R. Brun, P. Buncic, V. Innocente, P. Mato, A. Pfeiffer, D. Quarrie, F. Rademakers, L. Taylor, C. Tull, T. Wenaus, "Architecture Blueprint Requirements Technical Assessment Group (RTAG)", Report of the LHC Computing Grid Project, CERN, Oct. 2002
- [38] P. Buncic, F. Rademakers, R. Jones, R. Gardner, L.A.T. Bauerdick, L. Silvestris, P. Charpentier, A. Tsaregorodtsev, D. Foster, T. Wenaus, F. Carminati, "LHC Computing Grid Project: Architectural Roadmap Towards Distributed Analysis", CERN-LCG-2003-033, Oct-2003
- [39] P. Eerola, B. Konya, O. Smirnova, T. Ekelof, M. Ellert, J.R. Hansen, J.L. Nielsen, A. Waananen, A. Konstantinov, J. Herrala, M. Tuisku, T. Myklebust, F. Ould-Saada, and B. Vinter, "The nordugrid production grid infrastructure, status and plans", in Proceedings of the 4th International Workshop on Grid Computing, pages 158-165. IEEE CS Press, 2003.
- [40] EGEE - Enabling Grids for E-science in Europe: <http://egee-intranet.web.cern.ch/egee-intranet/gateway.html>

- [41] E. Hjort, J. Lauret, D. Olson, A. Sim, A. Shoshani, "Production mode Data-Replication framework in STAR using the HRM Grid", in Proceedings of Computing in High Energy and Nuclear Physics (CHEP04), Interlaken, Switzerland, Oct. 2004 (to appear)
- [42] I. Foster, J. Vckler, M. Wilde, Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation", in Proceedings of 14th International Conference on Scientific and Statistical Database Management (SSDB '02), Edinburgh, July 2002.
- [43] The Grid2003 Project, "The Grid2003 Production Grid: Principles and Practice", iVDGL, Technical Report, 2004: On line www.ivdgl.org/grid2003.
- [44] The Particle Physics Data Grid, "GRID2003 Lessons Learned", PPDG Document 37, <http://www.ppdg.net>
- [45] L. Pearlman, C. Kesselman, S. Gullapalli, B.F. Spencer, Jr., J. Futrelle, K. Ricker, I. Foster, P. Hubbard, C. Severance, "Distributed Hybrid Earthquake Engineering Experiments: Experiences with a Ground-Shaking Grid Application", in Proceedings of the 13th IEEE Symposium on High Performance Distributed Computing (HPDC-13), 2004.
- [46] SAM-Grid project: <http://www-d0.fnal.gov/computing/grid>
- [47] I. Terekhov, A. Baranovski, G. Garzoglio, A. Kreymer, L. Lueking, S. Stonjek, F. Wuerthwein, A. Roy, T. Tannenbaum, P. Mhashilkar, V. Murthi, R. Walker, F. Ratnikov, T. Rockwell, "Grid Job and Information Management for the FNAL Run II Experiments", in Proceedings of in Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, Ca, USA, March 2003, La Jolla, California, March 2003.
- [48] R. Walker, A. Baranovski, G. Garzoglio, L. Lueking, D. Skow, I. Terekhov, "SAM-GRID: A System Utilizing Grid Middleware and SAM to Enable Full Function Grid Computing", in Proceedings of the 8th International Conference on B-Physics at Hadron Machines (Beauty 02), Santiago de Compostela, Spain, Jun. 2002
- [49] G. Garzoglio, A. Baranovski, H. Koutaniemi, L. Lueking, S. Patil, R. Pordes, A. Rana, I. Terekhov, S. Veseli, J. Yu, R. Walker, V. White,

- "The SAM-GRID project: architecture and plan.", talk at the 8th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT-02), Moscow, Russia, Jun. 2002, Nuclear Instruments and Methods in Physics Research, Section A, NIMA14225, vol. 502/2-3 pp 423 - 425
- [50] I. Terekhov et al., "Meta-Computing at D0"; talk at the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT-02), Jun. 2002, Nuclear Instruments and Methods in Physics Research, Section A, NIMA14225, vol. 502/2-3 pp 402 - 406
- [51] M. Burgon-Lyon et al., "Experience using grid tools for CDF Physics"; talk at the IX International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT-03), Tsukuba, Japan, Dec 2004; to appear in Nuclear Instruments and Methods in Physics Research, Section A
- [52] Fermi National Accelerator Laboratory: <http://www.fnal.gov/>
- [53] PPDG: <http://www.ppdg.net/>
- [54] GridPP: <http://www.gridpp.ac.uk/>
- [55] SAM project: <http://d0db.fnal.gov/sam>
- [56] L. Loebel-Carpenter, L. Lueking, C. Moore, R. Pordes, J. Trumbo, S. Veseli, I. Terekhov, M. Vranicar, S. White, V. White, "SAM and the particle physics data grid", in Proceedings of Computing in High-Energy and Nuclear Physics. Beijing, China, Sep 2001.
- [57] I. Terekhov, R. Pordes, V. White, L. Lueking, L. Carpenter, J. Trumbo, S. Veseli, M. Vranicar, S. White, H. Schellman, "Distributed Data Access and Resource Management in the D0 SAM System", in Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), San Francisco, California, Aug. 2001
- [58] I. Terekhov, V. White, L. Lueking, L. Carpenter, H. Schellman, J. Trumbo, S. Veseli, and M. Vranicar, "SAM for D0-a fully distributed

- data access system”, talk at Advanced Computing And Analysis Techniques In Physics Research (ACAT 2000) Batavia, Illinois, Oct 2000, in American Institute of Physics (AIP) Conference Proceedings Vol 583(1) pp. 247-249. August 20, 2001
- [59] V. White et al., ”D0 Data Handling”, in Proceedings of Computing in High Energy and Nuclear Physics (CHEP01), Beijing, China, Sep. 2001
 - [60] L. Carpenter et al., ”SAM Overview and Operational Experience at the D0 experiment”, in Proceedings of Computing in High Energy and Nuclear Physics (CHEP01), Beijing, China, Sep. 2001
 - [61] L. Lueking et al., ”Resource Management in SAM and the D0 Particle Physics Data Grid”, in Proceedings of Computing in High Energy and Nuclear Physics (CHEP01), Beijing, China, Sep. 2001
 - [62] L. Lueking et al., ”The Data Access Layer for D0 Run II”; in Proceedings of Computing in High Energy and Nuclear Physics (CHEP 2000) Padova, Italy, Feb. 2000
 - [63] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Jagatheesan, C. Cowart, B. Zhu, S.Y. Chen, R. Olschanowsky, ”Storage Resource Broker - Managing Distributed Data in a Grid”, Computer Society of India Journal, Special Issue on SAN, Vol. 33, No. 4, pp. 42-54 Oct 2003.
 - [64] H. Stockinger, A. Samar, S. Muzaffar, and F. Donno, ”Grid Data Mirroring Package (GDMP)”, Scientific Programming Journal - Special Issue: Grid Computing, 10(2):121-134, 2002.
 - [65] H. Stockinger, F. Donno, E. Laure, S. Muzaffar, P. Kunszt, G. Andronico, P. Millar, ”Grid Data Management in Action: Experience in Running and Supporting Data Management Services in the EU DataGrid Project”, in Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, Ca, USA, March 2003
 - [66] A. Chervenak, E. Deelman, I. Foster, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, H. Stockinger, K. Stockinger, and B. Tierney, ”Giggle: A Framework for Constructing Scalable Replica

- Location Services”, in Proceedings of the International IEEE Supercomputing Conference (SC 2002), Baltimore, USA, November 2002.
- [67] J. Bent et al., ”Flexibility, Manageability, and Performance in a Grid Storage Appliance”, in Proceedings of the The 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, Jul. 2002
- [68] W. Deng, T. Wenaus, ”Magda - Manager for grid-based data”, in Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, California, March 2003
- [69] M. Beynon, T. Kurc, U. Catalyurek, C. Chang, A. Sussman, J. Saltz, ”Distributed Processing of Very Large Datasets with DataCutter”, Parallel Computing, 2001, pp. 1457-1478.
- [70] V. Raman, I. Narang, C. Crone, L. Haas, S. Malaika, T. Mukai, D. Wolfson, C. Baru, ”Services for Data Access and Data Processing on Grids”, Global Grid Forum Document GFD-I.14 Feb. 2003.
- [71] A. Tsaregorodtsev, V. Garonne, J. Closier, M. Frank, C. Gaspar, E. van Herwijnen, F. Loverre, S. Ponce, R. Graciani Diaz, D. Galli, U. Marconi, V. Vagnoni, N. Brook, A. Buckley, K. Harrison, M. Schmelling, U. Egede, A. Bogdanchikov, I. Korolko, A. Washbrook, J.P. Palacios, S. Klous, J.J. Saborido, A. Khan, A. Pickford, A. Soroko, V. Romanovski, G.N. Patrick, G. Kuznetsov, M. Gandelman, ”DIRAC - Distributed Infrastructure with Remote Agent Control”, in Proceedings of Computing in High Energy and Nuclear Physics, La Jolla, California, March 2003
- [72] K. Harrison, W. T. L. P. Lavrijsen, C. E. Tull, P. Mato, A. Soroko, C. L. Tan, N. Brook, R. W. L. Jones, ”GANGA: a user-Grid interface for Atlas and LHCb”, in Proceedings of Computing in High Energy and Nuclear Physics, La Jolla, California, March 2003
- [73] G. Avellino et al., ”The EU DataGrid Workload Management System: towards the second major release”, in Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, California, March 2003

- [74] The DataGrid team, "DataGrid Information and Monitoring Services Architecture: Design, Requirements and Evaluation Criteria", Technical Report, DataGrid 2002
- [75] A. Iamnitchi and I. Foster, "A Peer-to-Peer Approach to Resource Location in Grid Environments", In J. Weglarz, J. Nabrzyski, J. Schopf, and M. Stroinski eds. Grid Resource Management, Kluwer Publishing, 2003.
- [76] GridWeaver Technical Reports: <http://www.epcc.ed.ac.uk/gridweaver/docs/>
- [77] P. Goldsack, "SmartFrog - a framework for configuration", Talk at the Large Scale System Configuration Workshop, Nov. 2001
- [78] P. Anderson et al, "Towards automation of computing fabrics using tools from the fabric management workpackage of the EU DataGrid project", in Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, Ca, USA, March 2003.
- [79] The Apache Software Foundation, Apache Xindice: <http://xml.apache.org/xindice>
- [80] P. Anderson, "What is This Thing Called Configuration?", Talk at the Configuration Workshop LISA XVII, San Diego, California, USA, Oct. 2003
- [81] M. Holgate, W. Partain, "The Arusha Project: A Framework for Collaborative Unix System Administration", in Proceedings of the LISA 2001, 15th Systems Administration Conference, San Diego, California, USA, Dec. 2001
- [82] The World Wide Web Consortium, "RDF Primer" W3C Recommendation Feb. 2004
- [83] PACMAN: <http://physics.bu.edu/~youssef/pacman/>
- [84] UNIX Product Support (UPS): <http://www.fnal.gov/docs/products/ups/>
- [85] S. Andreozzi, M. Sgaravatto, C. Vistoli, "Sharing a conceptual model of Grid resources and services", in Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, California, March 2003

- [86] A. Lyon, S. Veseli, et al., "SAM-Grid Monitoring and Information Service and its Integration with MonALisa", in Proceedings of Computing in High Energy and Nuclear Physics, Interlaken, Switzerland, Oct. 2004 (to appear)
- [87] Czajkowski, K., S. Fitzgerald, I. Foster, C. Kasselmann, "Grid Information Service for Distributed Resource Sharing", in Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, 2001
- [88] MDS: <http://www.globus.org/mds/>
- [89] MDS References: <http://www-unix.globus.org/toolkit/mds/papers.html>
- [90] Hawkeye: <http://www.cs.wisc.edu/condor/hawkeye>
- [91] X. Zhang, J. Freschl, and J. Schopf, "Performance Study of Monitoring and Information Services for Distributed Systems", in Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12), Seattle, Washington, Jun. 2003
- [92] MonALISA: <http://monalisa.cacr.caltech.edu/>
- [93] Jini: <http://www.jini.org/>
- [94] Plale, B., P. Dinda, G. Laszewski, "Key Concepts and Services of a Grid Information Service." ISCA 15th International Parallel and Distributed Computing Systems (PDCS), 2002
- [95] V. Vaswani, P. Smith, "MySQL: The Complete Reference", published by The McGraw-Hill Companies, Aug. 2002
- [96] Plale, B., "Whitepaper on Synthetic Workload for Grid Information Services/Registries", DataWorkshop 2003 held in conjunction with GlobusWorld 2003, San Diego.
- [97] A.S. Rana, "A globally-distributed grid monitoring system to facilitate HPC at D0/SAM-Grid (Design, development, implementation and deployment of a prototype)", Thesis of Master in Computing Science, The University of Texas, Arlington, Nov. 2002

- [98] I. Foster, G. von Laszewski, "Usage of LDAP in Globus", On line: ftp://ftp.globus.org/pub/globus/papers/ldap_in_globus.pdf
- [99] OpenLDAP: <http://www.openldap.org/>
- [100] Yeong, W., T. Howes, and S. Kille, "Lightweight Directory Access Protocol", The Internet Engineering Task Force (IETF) RFC 1777, Mar. 1995
- [101] The Network Working Group, "The LDAP Data Interchange Format (LDIF)", Technical Specification, RFC 2849
- [102] G. Garzoglio, I. Terekhov, A. Baranovski, S. Veseli, L. Lueking, P. Mhashilkar, V. Murthi, "The SAM-Grid Fabric services", talk at the IX International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT-03), Tsukuba, Japan; to appear in Nuclear Instruments and Methods in Physics Research, Section A
- [103] SAM-Grid history plots: <http://dbsmon.fnal.gov/samgrid/samgrid.html>
- [104] M. Satyanarayanan, "The Evolution of Coda", ACM Transactions on Computer Systems, Vol. 20, No. 2, May 2002, Pages 85-124
- [105] SAM-Grid monitoring page: <http://samgrid.fnal.gov:8080/>
- [106] PHP: <http://www.php.net>
- [107] Grid Job Submission Project: <http://auger.jlab.org/grid/>
- [108] The EDG Team, "DataGrid JDL ATTRIBUTES", Internal Document of Work package 1, DataGrid-01-TEN-0142-0_2, Oct. 2003
- [109] Personal Communication, Todd Tannenbaum and Zach Miller, Department of Computer Sciences, University of Wisconsin, Madison
- [110] MOP <http://www.uscms.org/s&c/MOP/>
- [111] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid", in Grid Computing: Making the Global Infrastructure a Reality, published by John Wiley & Sons Inc., Dec. 2002

- [112] M.S. Neubauer, "Computing for Run II at CDF", in Proceedings of the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT-02), Jun 2002
- [113] J. Novotny, S. Tuecke, V. Welch, "An Online Credential Repository for the Grid: MyProxy", in Proceedings of the 10th International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, Aug. 2001
- [114] R. Buyya, S. Chapin, and D. DiNucci, "Architectural Models for Resource Management in the Grid", First IEEE/ACM International Workshop on Grid Computing (GRID 2000), Springer Verlag LNCS Series, Germany, Dec. 17, 2000, Bangalore, India.
- [115] K. Krauter, R. Buyya, and M. Maheswaran, "A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing", International Journal of Software: Practice and Experience (SPE), Wiley Press, New York, USA, May 2002.
- [116] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors", Journal of the ACM, 24(2):280-289, 1977.
- [117] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems" in Proceedings of the 8th IEEE Heterogeneous Computing Workshop (HCW-99), pages 30-44, 1999.
- [118] D. A. Menasce, D. Saha, S. C. D. S. Porto, V. A. F. Almeida, and S. K. Tripathi, "Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures", Journal of Parallel and Distributed Computing, 28:1-18, 1995.
- [119] D. Paranhos, W. Cirne, and F. Brasileiro, "Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids, In International Conference on Parallel and Distributed Computing (Euro-Par), Lecture Notes in Computer Science, volume 2790, pages 169-180, 2003.

- [120] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman, "Heuristics for scheduling parameter sweep applications in grid environments", in 9th Heterogeneous Computing Workshop (HCW), pages 349-363, 2000.
- [121] N. Fujimoto, K. Hagihara, "A Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks", in 2004 Symposium on Applications and the Internet-Workshops (SAINT 2004 Workshops), Jan. 2004, Tokyo, Japan, p.674
- [122] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson, "Economic Models for Management of Resources in Peer-to-Peer and Grid Computing", In Proceedings of International Conference on Commercial Applications for High-Performance Computing, SPIE Press, August 20-24, 2001, Denver, Colorado, USA.
- [123] M. Neary, A. Phipps, S. Richman, P. Cappello, "Javelin 2.0: Java-Based Parallel Computing on the Internet", in Proceedings of European Parallel Computing Conference (Euro-Par 2000), Germany, 2000.
- [124] S. Chapin, J. Karpovich, and A. Grimshaw, "The Legion Resource Management System", in Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing, Apr. 1999, San Juan, Puerto Rico, Springer Verlag Press, Germany, 1999.
- [125] J. Gehring and A. Streit, "Robust Resource Management for Metacomputers", in Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing, Pittsburgh, USA, 2000.
- [126] H. Casanova and J. Dongarra, "NetSolve: A Network Server for Solving Computational Science Problems", International Journal of Supercomputing Applications and High Performance Computing, Vol. 11, No. 3, pp 212-223, Sage Publications, USA, 1997.
- [127] H. Nakada, M. Sato, S. Sekiguchi, "Design and Implementations of Ninf: towards a Global Computing Infrastructure", Future Generation Computing Systems, Metacomputing Special Issue, October 1999.
- [128] N. Kapadia, R. Figueiredo, and J. Fortes, "PUNCH: Web Portal for Running Tools", IEEE Micro, May-June, 2000.

- [129] R. Buyya, "Economic-based Distributed Resource Management and Scheduling for Grid Computing", Ph.D. Thesis, Monash University, Melbourne, Australia, Apr. 2002. Online at <http://www.buyya.com/thesis/thesis.pdf>
- [130] R. Raman and M. Livny, "Matchmaking: Distributed Resource Management for High Throughput Computing", in Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, Chicago, IL, Jul. 1998
- [131] A. Baranovski, G. Garzoglio, I. Terekhov, A. Roy, T. Tannenbaum, "Management of Grid Jobs and Data within SAM-Grid", In Proceedings of Cluster 2004, Sept. 20-23 2004, San Diego, California (to appear)
- [132] R. Raman, "Matchmaking Frameworks for Distributed Resource Management", Ph.d Dissertation, University of Wisconsin, Madison, Oct. 2000. Online: <http://www.cs.wisc.edu/condor/doc/rajesh.dissert.pdf>
- [133] H. Li, D. Groep, J. Templon, L. Wolters, "Predicting Job Start Times on Clusters", In Proceedings of the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid Chicago, Illinois, USA, April 19-22, 2004
- [134] I. Bird, B. Hess, A. Kowalski, D. Petravick, R. Wellner, J. Gu, E. Otoo, A. Romosan, A. Sim, A. Shoshani, W. Hoschek, P. Kunszt, H. Stockinger, K. Stockinger, B. Tierney and J. Baud, "SRM (Storage Resource Manager) Joint Functional Design", Global Grid Forum Document, GGF4, Toronto, Feb. 2002
- [135] RTE project: <http://www-d0.fnal.gov/~ritchie/CPBdemo.html>
- [136] Disk Farm project: <http://www-isd.fnal.gov/dfarm>