BRIDGING TWO GRIDS: THE SAM-GRID/LCG INTEGRATION PROJECT

by

TUMMALAPALLI SUDHAMSH REDDY

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTUER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

MAY 2006

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Mr. David Levine for his patience, support and encouragement. He taught me how to think and write about my research. I would like to thank the members of my thesis committee Dr. Ramesh Yerraballi and Dr. Jaehoon Yu.

I would like to extend my gratitude to Mr. Levine, Dr Yu and Fermilab for providing me a wonderful opportunity to work with the computing research team at Fermilab.

I would like to mention my supervisor at Fermilab, Mr. Gabriele Garzoglio, for his valuable advice, explanations and support during the course of this research, Mr. Andrew Baranovski, the project leader for the SAM-Grid/LCG interoperability project, for his support and indulgence, Mr. Parag Mhashilkar, for his support and explanations.

The entire Dzero team, past and present, deserves a special mention.

Finally, I would like to thank my family and friends without whose constant support and encouragement this work would not have been possible.

March 25, 2006

ABSTRACT

BRIDGING TWO GRIDS: THE SAM-GRID/LCG

INTEGRATION PROJECT

Publication No. ______

Tummalapalli Sudhamsh Reddy, M.S

The University of Texas at Arlington, 2006

Supervising Professor: Mr. David Levine

SAM-Grid is an integrated data, job, and information management system. SAM-Grid addresses the distributed computing needs of the Dzero experiment at Fermi National Accelerator Laboratory, Batavia, IL. The system typically relies on SAM-Grid specific services deployed at the remote sites in order to manage the computing and storage resources. Such deployment requires special agreements with each resource provider, and it is a labor intensive process. Some members of the Dzero VO also have access to computing resources through the Large Hydron Collider Computing Grid (LCG) infrastructure. Therefore, Dzero users can enter into resource sharing agreements and deployment of standard middleware within the framework of the LCG project. The SAM-Grid/LCG interoperability project was started to enable the Dzero users to access the LCG pool of resources while, retain the user-friendliness of the SAM-Grid interface. This "bridging" between grids is beneficial for both SAM-Grid and LCG, as it minimizes the deployment efforts of the SAM-Grid team and tests the LCG computing infrastructure with data intensive production applications of a running high energy physics experiment, which, are also the types of applications that LCG will have to run once LHC goes into production.

The interoperability system is centered on "forwarding" nodes, which receive jobs prepared by the SAM-Grid interface and submits them to LCG. In this thesis, we discuss the architecture of the forwarding system and how it addresses issues of service accessibility, scalability, security challenges, operational and support challenges, which arise when operating this system in production.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

1.1 General Introduction

The amount of computation required to process the data produced by current high energy physics experiments is enormous. No single institution in the world has the resources or can afford to have resources to process such vast amounts of data in a short time scale. Therefore, new models of distributed, shared computing are required to meet the goals of processing such substantial amounts of data. Currently the run II experiments at Fermilab, Dzero [1] and CDF [2] are producing experimental results data at a rate of one petabyte per year. This data is in raw format and, has to be processed and converted into meaningful physics data for scientists to analyze. Two of the biggest issues in processing this data are the amount of data and the optimal utilization of geographically distributed resources, computing resources and human resources.

Resources are geographically distributed and often join and leave the research collaboration during the course of the experiment, there is no direct control over the resources by the experiment. These resources, working together for some amount of time to accomplish a common goal is called a “virtual organization” (VO) [3]. Virtual organizations provide an efficient, accountable and secure computing infrastructure.

This infrastructure is called “Grid Computing” [3]. Many major physics and astrophysics experiments are already using grid computing technology (examples: BaBar[4], Belle[5], Dzero[6], CDF[7], SDSS[8], LIGO[9], ATLAS[10], CMS[11, 12], LHCb[13], Alice[14, 15]).

1.2 Principle of Grid Computing

The definition of grid computing has evolved over time. In their seminal book in 1998, I. Foster and C. Kesselman defined a grid as “a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities” [16]. This was later widened to include accounting as well as access and resource sharing policy by additionally noting that grid computing also considers “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.”[17]. Finally in the article, “What is the Grid”[18], a three point checklist is presented, in which a grid “coordinates resources which are not under any centralized control, using standard, open, general-purpose protocols and interfaces to deliver non-trivial qualities of service.”[18].

According to CERN, “the Grid is a service for sharing computer power and data storage capacity over the Internet” [19].

Even though the definition of the grid has changed over the years, the core objectives remain the same: to be able to combine geographically distributed resources to provide users with vast computing power.

High energy physicists today need higher computing power then ever before. Yet such large computing power is not available in one place. Even large physics labs like CERN [20] or Fermilab [21] or Brookhaven National Lab [22] do not have the substantial computing power required for the current experiments The only way they can meet their needs are to combine all the computing resources that are available at major sites such as those previously listed in addition to computing resources available at various academic and research institutions across the globe. Grid computing is a solution to this problem.

1.3 Cluster Computing

Computational clusters can be loosely defined as a group of computers that work together in a unified way, such that in many aspects, they appear similar to a single computer. Clusters are groups of computers which are generally connected using a high speed network usually running the same operating system and same software toolset, and more significantly under one administrative domain. This fact is important with respect to grid computing, because one of the key distinctions between the two is that, clusters are generally under one administrative domain and belong to the same group of people, and hence policies regarding the use of available resource can be entered unilaterally. Whereas in grids, which are made of independent clusters any rules about resources can’t be entered unilaterally because they are under different administrative domains.

Clusters can be classified into three main types: high performance clusters, load balancing clusters and high-availability clusters. High-availability clusters are mainly used for the purpose of improving the availability of services which the cluster provides. They operate by having additional nodes, which are used to provide service when system components fail. Load balancing clusters operate by accepting all incoming workloads through one or two front end computers and then distributing the tasks amongst a collection of back end machines. These types of clusters are generally used as server farms. High performance clusters are the ones we generally deal with in grid computing. High performance clusters are implemented to provide increased performance by splitting a task into several sub tasks, which are then executed on different nodes in the cluster. Note that the term “node” refers to a machine which is a part of the cluster. This type of cluster is generally used in scientific computing.

Cluster computing can be used to provide computing power higher then the fastest single computer. Clusters are generally cheaper then mainframe computers and are highly preferred. But a cluster is generally not big enough to provide the amount of computing that current experiments require. This is because, even though they are cheaper then a mainframe computer, the cost/performance ratio is considerably better. The biggest cluster in the SAM-Grid collaboration is the Westgrid cluster in Canada, which has 1000 nodes. More often, having a big cluster is not always an advantage compared to having a set of clusters, since often the diverse number of groups that come together to form such a cluster are very large as in the case of Westgrid, which generally is able to provide SAM-Grid with only 200 nodes. Hence, we use grids which connect these clusters and provide a vast amount of computing resources to the users.

The major issues that appear when we move from clusters to grids are security and data handling. Security is a major concern because the users who access these resources are generally at a remote site and the system administrators often do not know them. Data handling is another issue because the data that is used to run a job as well as the data that has been generated by the job needs to be transferred between the client and the node at which the job executes. We handle both these issues using standard grid middleware, described in the following sections.

1.4 Grid Middleware

The current state of the art in terms of the middleware for the grid is based on the Globus Toolkit [23], which is provided by the Globus Alliance [24] and is supported by the Global Grid Forum [25]. The main components of this toolkit are:

1. Resource Management: Provided by Globus resource allocation and management [26] (GRAM) protocol. It provides job managers and a server called a gatekeeper. The gatekeeper is mainly responsible for authentication and authorization. The gatekeeper is also responsible for creating a grid service for the user. The job manager instance is a service providing job control and resource management. GRAM

also allows users to also express the requirements of the jobs in a Resource Specification Language (RSL)

2. Security: The Globus security infrastructure (GSI) [27-29], is based on public key encryption techniques based on X509 Certificates, which are provided by the certificate authority. These certificates are requested by users and administrators and are signed by a Certificate Authority (CA), which is trusted by the users and the resource providers. The Globus security infrastructure is used by all components of the grid system.

3. Data transfer protocols: A FTP service is used which is GSI-enabled called GSIFTP [30, 31]. Additionally data is replicated and can be accessed by using the replica location service which uses the Globus Replica Catalog.

4. Monitoring and discovery Services (MDS): A GSI enabled LDAP service. MDS [32, 33] provides services which can be used to monitor the resources and jobs running on these resources.

Various other grid enabling software are available (Avaki [34], Platform Computing [35], Sun Grid Engine [36], United Devices [37], Parabon [38], ProcessTree [39], and DataSynapse [40]), but Globus is the most widely accepted and used. There are several other projects that offer higher level services based on these middleware [41, 42, 43], such as Condor [44]. Condor is essentially a batch system which has been integrated with Globus to provide a much easier to use grid solution known as Condor-G [45]. Condor-G has been widely used as a part of many grid middleware solutions

[46-54]. Condor-G provides a powerful, full-featured task broker which can be used as a front-end to a computational grid. It also provides job monitoring, logging, notification, policy enforcement, fault tolerance and credential management. Condor provides matchmaking service, which can be used between clusters in the grid.

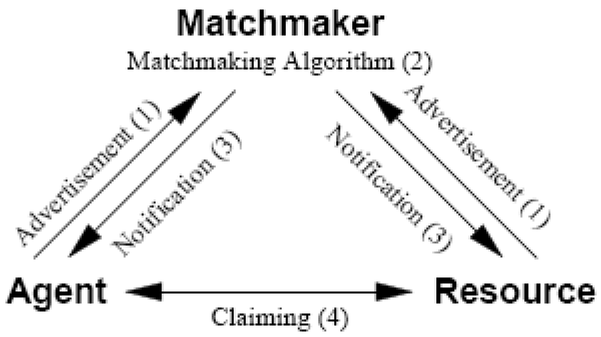
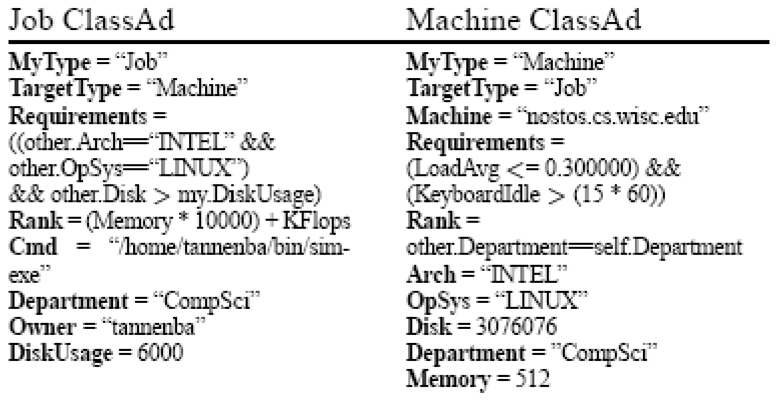


Figure 1.1 Matchmaking in Condor.

As can be seen from figure 1.1 [55], the agent (user) requests for a resource to run his/her job. The resource sends a request for a job. The requests are sent in the form of Classads, figure 1.2 [55]. The matchmaker accepts both the classads and resolves the requests from both the user and resource. After which it sends a notification to both the user and the resource. At this point the user can claim the resource for executing the job.



Job ClassAd

```
MyType = "Job"
TargetType = "Machine"
Requirements =
((other.Arch=="INTEL" &&
other.OpSys=="LINUX")
&& other.Disk > my.DiskUsage)
Rank = (Memory * 10000) + KFlops
Cmd = "/home/tannenba/bin/sim-
exe"
Department = "CompSci"
Owner = "tannenba"
DiskUsage = 6000
```

Machine ClassAd

```
MyType = "Machine"
TargetType = "Job"
Machine = "nostos.cs.wisc.edu"
Requirements =
(LoadAvg <= 0.300000) &&
(KeyboardIdle > (15 * 60))
Rank =
other.Department==self.Department
Arch = "INTEL"
OpSys = "LINUX"
Disk = 3076076
Department = "CompSci"
Memory = 512
```

Figure 1.2 Example of a simple job and resource classads.

These middleware have some disadvantages: the resource specification language not being user friendly, not being fault tolerant enough, not providing reliable job management and scratch space management. Therefore, we need to integrate the standard middleware with some in-house development so that we can meet the huge requirements of high energy physics applications. One such effort is the SAM-Grid project, currently underway at Fermi National Accelerator Labs. Another effort is the LCG [47, 56] project, currently underway at various institutions in Europe.

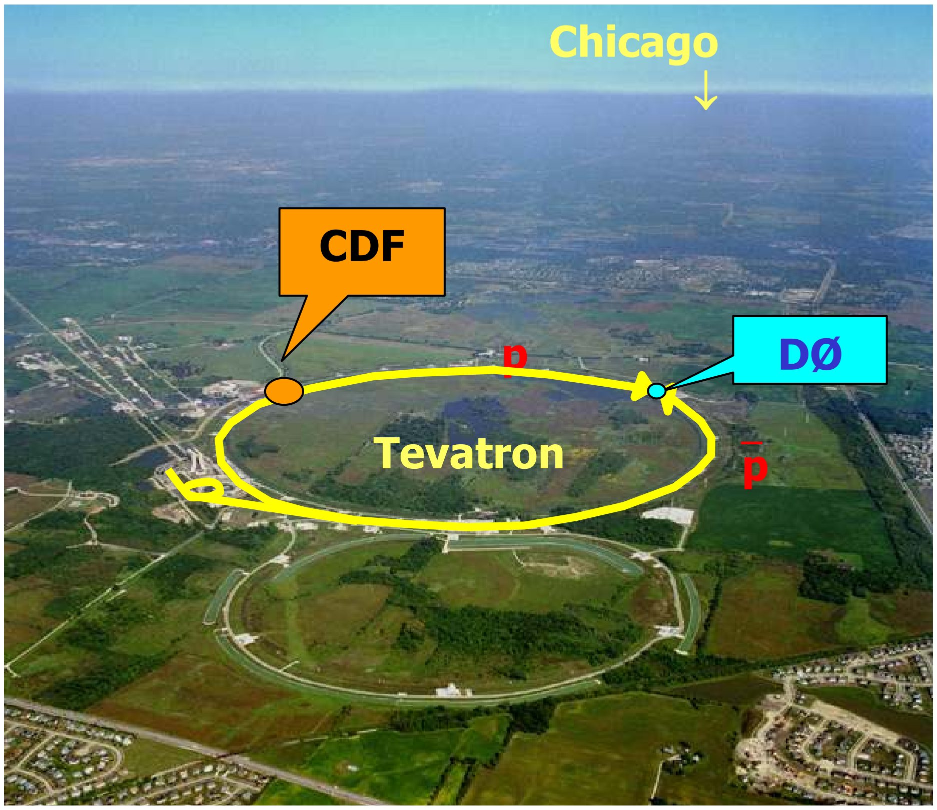


Figure 1.3 Tevatron accelerator at Fermilab.

The SAM-Grid project is used by the Dzero experiment, which is currently taking data at Tevatron (figure 1.3 [97]), the particle accelerator at Fermilab. The Dzero experiment records data by colliding particles of matter with particles of anti-matter. These particles are accelerated in the highest-energy particle accelerator in the world called Tevatron. It is capable of accelerating particles to energy levels of up to 1TeV (1 teraelectron volt = 1.60217646 × 1e-7 joules). During these collisions a vast amount of energy is released in the form of emissions. These emissions are recorded by using extremely sensitive detectors. Each collision is considered as an event. These events are then passed through a trigger system which contains three levels of triggers. If the event satisfies all three levels of triggers then it is sent to be stored in the mass storage system. The recordings are stored on tape drives and are then analyzed.

The process of analysis is extremely CPU intensive as events are recorded at the rate of 50 Hz and each event is of the size of 250 KB. Therefore, the data is broken down into smaller data set often containing just a few events. Even these datasets, often take more then a month of computing time if done on just one computer. Therefore, the datasets are run on clusters and depending on the size of the cluster they finish in a couple of days to a week.

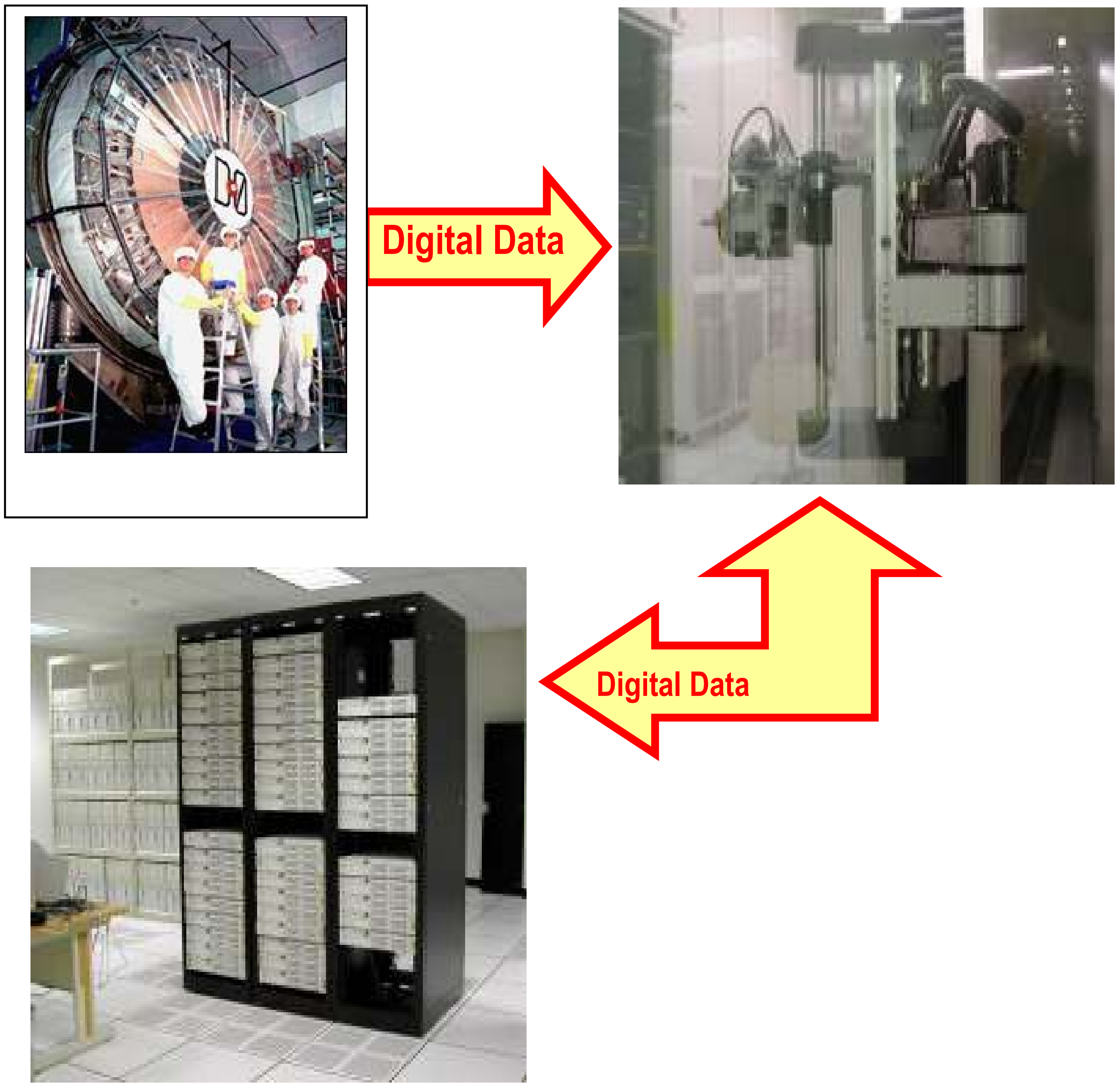
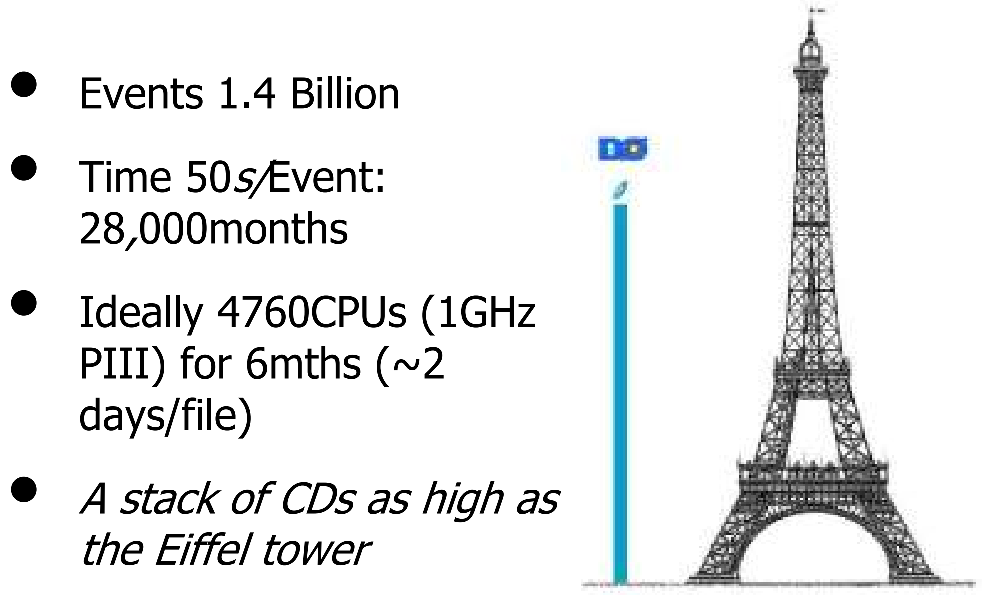


Figure 1.4 The flow of data from the detectors to the Grid



- Events 1.4 Billion
- Time 50*s/*Event: 28,000months
- Ideally 4760CPUs (1GHz PIII) for 6mths (~2 days/file)
- *A stack of CDs as high as the Eiffel tower*

Figure 1.5 Some numbers as of March 2006.

The Dzero computing mainly involves three main categories [71], which are data filtering or as is commonly known among the Dzero collaborators “reconstruction”, Monte Carlo simulation of events, and data analysis. The analysis phase consists of the selection and the statistical study of particles with certain characteristics, with the goal of achieving physics measurements. For the analysis phase, the reconstruction and Monte Carlo phases are indispensable. During data reconstruction phase, the binary format of events from the detector is transformed into a format that is closer to the abstract physics concepts, such as particle tracks, charge, spin, et cetera. The original format, known as “raw format”, is instead very closely dependent on the hardware layout of the detector i.e. it is dependent on the number of channels, calorimeters and silicon detectors, in order to guarantee the performance of the data acquisition system, and this format is not suitable for data analysis. On the other hand, simulation of 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. This process of doing Monte Carlo simulations to increase the efficiency of the system is a continuing process to advance the understanding of the physics involved. These three application types differ amongst themselves but are similar with respect to the usage of computing resources. The typical duration of a single reconstruction or montecarlo job is dozens of hours, while data analysis ranges depending on the problem studied from a few hours to a few minutes. All the activities are CPU intensive, but while both reconstruction and analysis are highly I/O intensive, montecarlo is not. In fact, montecarlo requires very little input data, while for reconstruction and analysis the input ranges 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.

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

Figure 1.6 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. [71]

The LCG project is to be used by the experiments at LHC, in CERN. Some of these experiments are ATLAS, CMS, and LHCb.

CHAPTER 2

RELATED WORK

2.1 SAM-Grid Project

The SAM-Grid [57-62] project contains three major components: data handling, the job and the information management systems. The data handling component is known as SAM [63-71], the job and information management systems are provided by JIM [71-76] (job and information management). The match making services of the SAM-Grid is provided by the Condor MMS services. The data handling system, Sequential Access via Metadata (SAM), is a result of an in-house development effort which has been developed for the CDF and Dzero experiments. The JIM software is used for submission, execution and monitoring of a grid job. The general architecture of SAM-Grid is shown in figure 2.1 [71].

2.1.1 Sequential Access via Metadata (SAM)

The main questions that SAM as a data handling system needs to answer are how to store the data securely and reliably, how to catalog the data, how to handle the resources that are used to store this data and finally how to distribute the data globally. SAM has been designed to answer these questions. The main tasks of SAM are:

1. To keep track of the location and comprehensive metadata of all files in the system.

2. To provide an interface to add the files to a permanent mass storage system (Which is implemented on robotic tape store).

3. To reduce the time to get a file to the user by caching files on local disks for the duration specified by the local SAM admin, which are then available to other users to copy to their local cache.

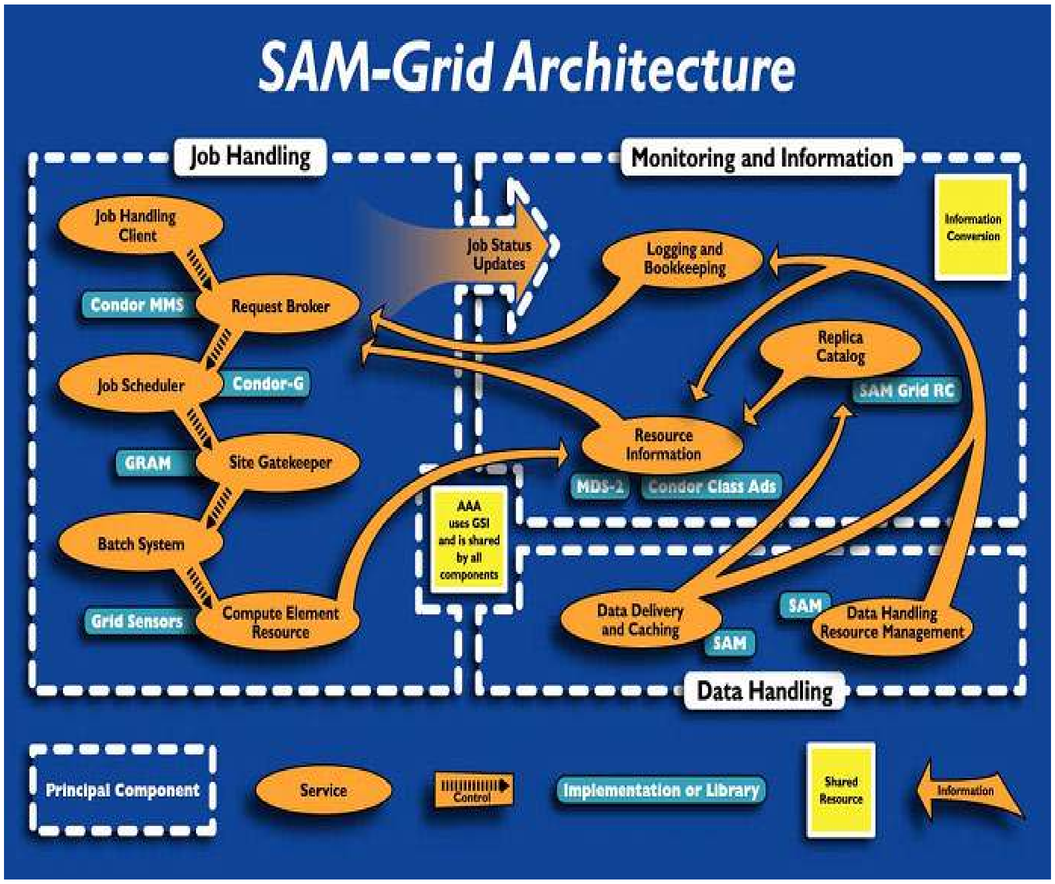


Figure 2.1 The general architecture of SAM-Grid.The SAM-Grid is divided into three major components: data handling, job handling and information management. All three components are integrated with strong security mechanisms. Each orange bubble represents an abstract aggregated service, whose implementation appears in the blue label next to it.

4. To deliver the files on request using a secure file delivery and routing protocols, like GridFTP.

5. Provide interfaces which can be used during job submission.

6. Keep track of files during the job execution stage to ensure weather a file has been used.

The SAM components generally belong to two categories:

- Global Data Handling Services [77, 72]
- Local Data Handling Services. [77, 72]

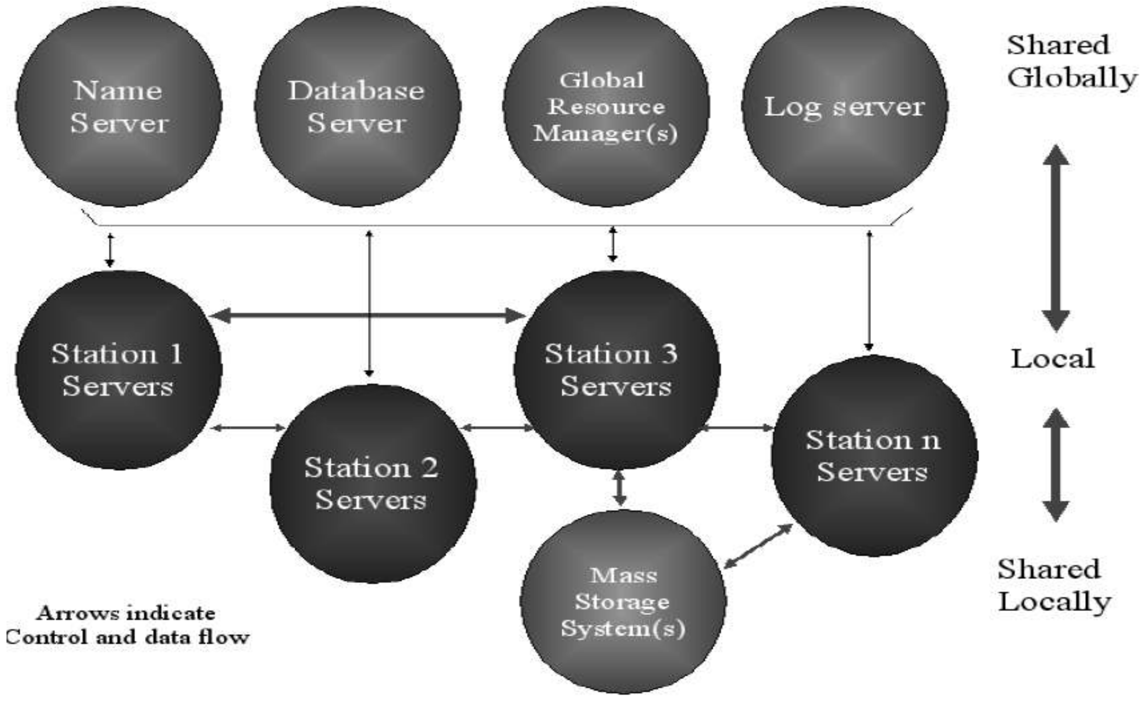


Figure 2.2 Overview of the SAM System

2.1.1.1 Global Data Handling Services

These services are common to all the components and are also accessible to any client which requests for their services. This is shown in figure 2.2 [77].

1. Name Server: The objective of the name server [72] is to provide naming services to all components in the system. All resources in the SAM domain have a name assigned to them by which they are identified in the system. When any component in the system needs to find any other component, then they look up the component by passing the naming service the name of the component. The naming service resolves the name to a component. This is done using CORBA.

2. Database Servers: This server is responsible for maintaining a stable and coherent view of all the files in the system. Each file has its properties described by meta-data. This meta-data is stored in these database servers [72].

3. Resource Managers: The resource managers are responsible for managing storage locations, like mass storage system, buffers and caches. It is also responsible for storing and retrieving data so as to make optimum use of resources.

4. Log Server: This component provided logging services and it records all interactions within the system, such as, request for files, start of process etc.

2.1.1.2 Local Data Handling Services

These services are local to the execution site and they are accessible to the resources at the site by using the interfaces that are provided by them. This is shown in figure 2.3 [72].

1. SAM Station [77]: This is the entry point for all data into an execution site. The parameter, which is mainly used for mapping a job to an execution site, is its associated station name. The station name is registered with the SAM naming service. The main responsibility of each station is to manage the data on SAM "caches", which are disk areas at the execution site that are used to store the data files temporarily before being given to the executing jobs. The stations have additional responsibilities such as ensuring that all the processes which are involved in getting the data do not crash and also in case of a failure revive them. The station is mainly used to obtain data from either SAM stations in Fermilab or stations running elsewhere and provide them to the jobs which request them.

2. SAM Stagers: The stager [77] is responsible for actual data transfers. The station decides from where we get the data and passes this information to the stager. The stagers are then responsible for actually getting the data from the remote site using tools such as GridFTP. Each stager is associated with a station.

3. File Storage Server (FSS): The FSS [77] is responsible for storing the processed data back into SAM. Once a job finishes execution and has produced the output files, the output files are sent to the head node where the FSS runs. The FSS determines the location at which the output files must stored. It then passes this information to the Stager.

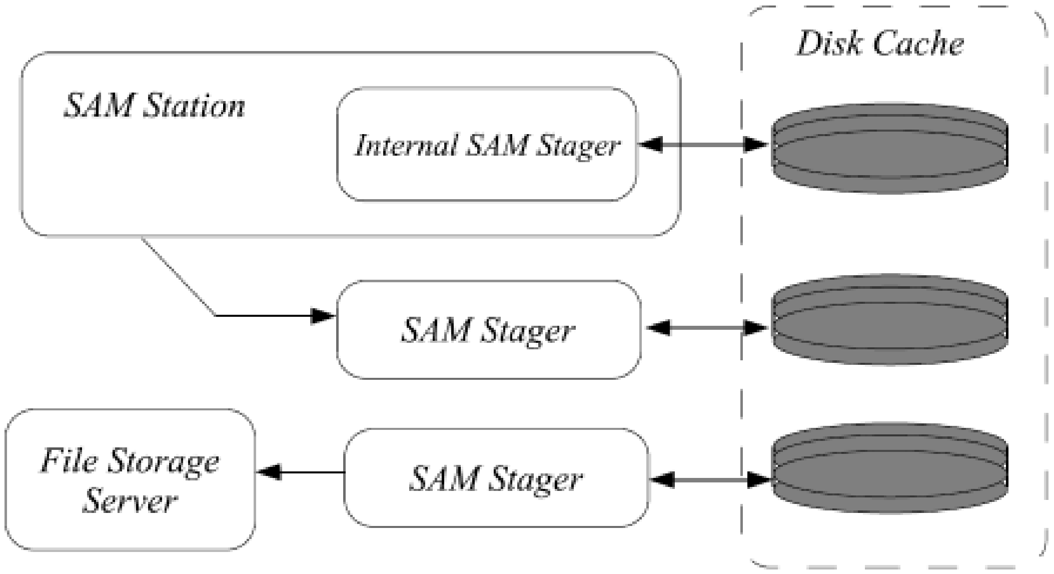


Figure 2.3 Local Data Handling Services

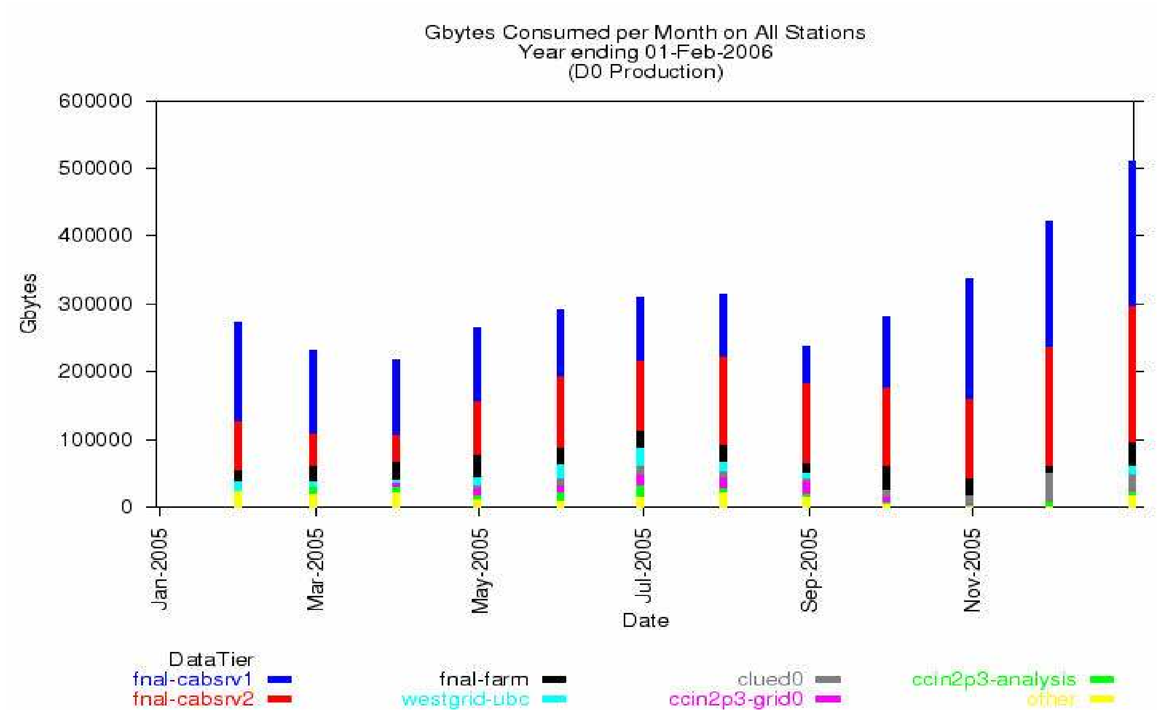


Figure 2.4 The amount of data in Gigabytes used per month by Dzero applications.

As can be seen from the Figure 2.4 the amount of data that is used or consumed by the Dzero applications are around 300 Terabytes per month.

2.1.2 Job Management System

Some of the questions that a job management system must generally address are how to provide reliability execution of jobs i.e. each job must reach a final state irrespective of the final state being a success or failure. It needs to ensure the system is fault tolerant i.e. even in case of a failure of any service we increase the probability of success of the job by retrial or other methods. It needs to hide the complexities of job submission from the users by providing a user friendly interface. The job management system does all of above and also does additional services such as dynamic software deployment, workspace configuration management and workflow management. It also implements application sensitive policies and job aggregation so that multiple applications of the same type can be aggregated to present the user with a single initial request.

2.1.3 Information Management System

The information management system deals with monitoring, configuring and bookkeeping of jobs and sites.

The information we deal with can be classified as:

1. The static or semi-static information which deals with the configuration and setup of services and resources available at a site.

2. The dynamic information which is generally related to monitoring

3. The logging information which is stored for bookkeeping.

The configuration and setup information is stored at the site in Xindice XML [78] databases. The monitoring information about the jobs is also stored in these databases, and is used by the SAM-Grid monitoring service which mines these local databases to present the monitoring information to users in a more user friendly format [79]. SAM-Grid relies on SAM for bookkeeping information. For information regarding the resource discovery process we use the information gathering mechanism provided by the Condor match making system. Each cluster site advertises information regarding the site to a central condor collector, which uses this information to provide match making of the jobs to the site.

2.2 LHC Computing Grid (LCG) Project

The LCG [47, 56] is the grid middleware that is used by the LHC experiments in Europe. We have no involvement in its design or implementation, and it is being included here for the sake of completeness, so that the reader has an understanding of the system. It is based on standard middleware such as Globus and Condor. It also includes in-house development carried out by various institutions across Europe. The current middleware is based on the development work that was done for the European Data Grid (EDG) project [80-83]. Additional components such as glite [84, 85] are currently being added to the LCG software. Current state of LCG (also known as LCG-1) can be described as a user interface which provides a front end to the users. A resource broker, replica catalog and information index which does the match making, this is based on Condor and the computing element, which provides a gateway to computing resources. A computing element can be a cluster gateway which provides access to the worker nodes in the cluster.

A unique feature of LCG is that the output gathering is based on a pull based approach instead of a push based approach as is used in SAM-Grid. In SAM-Grid when a job finishes, the output of the job is pushed by the execution site to the submission site. But in the case of LCG, when a job finishes execution, the user must get the output from the system using an interface which has been provided.

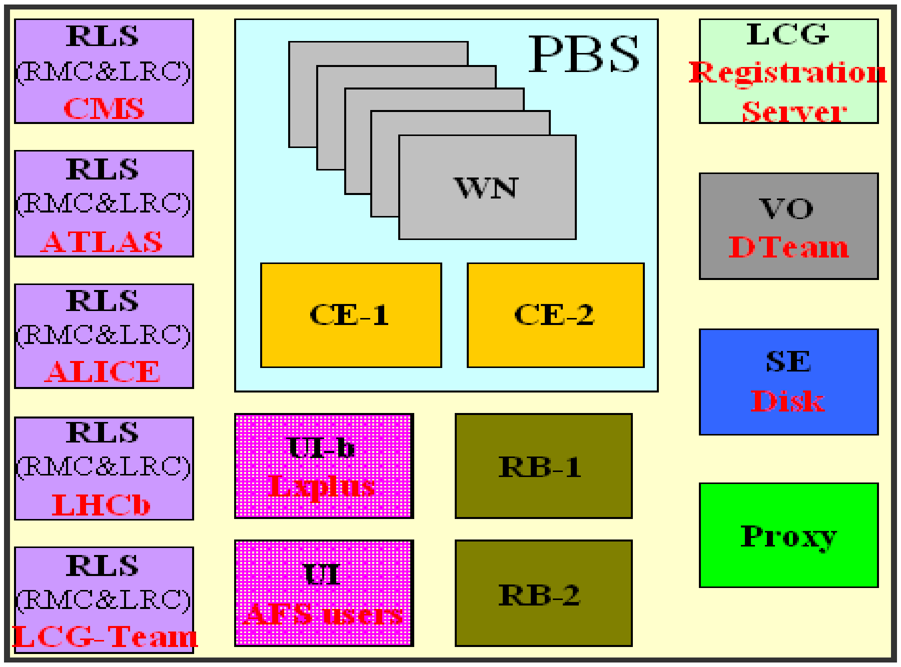


Figure 2.5 Components of LCG-1 System

A description of the components shown in Figure 2.5 [86] is given below

RLS: Replica location services. They maintain information about data locations and the metadata entries associated with the data. All files in LCG-1 are identified with a unique ID called Grid universal ID (GuID).

UI: User Interfaces. This is the initial point of access to the grid. This component allows the users to access the services provide by the workload, data and information management systems. It provides a command user interface to submit a job to the grid, list the resource available for the job, cancel jobs, get output of jobs, copy files and find the status of the jobs.

CE: Computing elements. This is a queue provided by the batch system for grid users. It is uniquely identified by a hostname/batch queue name parameter. This provides access to computing nodes which are also known as worker nodes (WN). The CE in a sense acts as a gateway to these resources. It contains the Globus-gatekeeper, GRAM, master server of the batch system and EDG monitoring and bookkeeping services.

SE: Storage Element. This provides uniform access to storage resources. And it runs a GridFTP server to provide data to users.

RB: The resource broker [47] is where the workload management system runs. It accepts the job from the users and based on the requirements matches the jobs to sites which are known to it via the information index servers.

LCG Registration Server and VO: All users need to register their grid certificate to this server as this periodically informs all sites of new users in a VO.

Proxy: This is a proxy server based on MyProxy [87-89]. Users can store their proxies at this server, and the jobs can retrieve the proxy in case the proxy used during submission is about to expire. This increases the probability that the job will run to finish and not fail due to proxy expiration.

CHAPTER 3

GOALS AND MOTIVATION FOR SAM-Grid/LCG INTEROPERABILITY PROJECT

3.1 Motivation

As explained in Chapter 1, the Dzero experiment is currently recording data and will continue to record data for another 4 years. The data that has been recorded will be processed for 5 years after stopping recording the data. But the LHC projects have not started recording data, and will not be starting for another year or two. LCG is still in the initial phases and needs testing to provide higher reliability. Meanwhile, the members of Dzero are also members of the LCG project. The Dzero VO can access some of the resources that are currently being deployed as part of LCG. Therefore, Dzero can use these resources also.

The SAM-Grid system relies on SAM-Grid services to be deployed on the remote sites to manage and use the resources at these sites. Such deployments require special agreements with the resource providers and are also a very labor intensive process. The installation and configuration of the SAM-Grid software on a remote site is a non-trivial process and often requires members of the SAM-Grid team to intervene during the initial installation and testing of the sites. On the other hand, the Dzero VO can access the resources present in Europe via the LCG infrastructure which is already installed at all the sites. Resource sharing agreements and deployment of standard middleware are negotiated within the framework of the EGEE [90] project.

The SAM-Grid/LCG interoperability project was designed to allow the users of Dzero to retain the user friendliness provided by the SAM-Grid user interface and at the same time allowing them to make use of the resources provided by LCG. Some features of the SAM-Grid system are important and must be retained to allow the users to express their requirements, for example the way in which the users can specify the data they want to process. The way the data is handled during all stages of job execution, the data cataloging and bookkeeping, and more importantly not requiring the users to change the way they do their job submission. This interoperability project is designed to allow SAM-Grid users to submit jobs to the LCG resources and not vice versa. This project is important as it minimizes the deployment efforts for the SAM-Grid team and provides LCG with data intensive physics applications of currently running experiments, which LCG will have to start handling as soon as the LHC goes into operation.

3.2 Statement of the Project

The goal of this project is providing the Dzero users access to the pool of resources provided by LCG, while;

1. Retaining the user-friendliness provided by SAM-Grid user interface.

2. Using the services provided by the SAM system which are critical for success of the experiment.

3. Reducing the deployment and management effort of SAM-Grid services.

4. To provide LCG with a rich mixture of data intensive production applications from a running experiment to test their infrastructure.

I believe this work is important because, interactions between grids have currently not been studied and, with the advent of large Grid systems like European data Grid, the Open Science Grid [90], Teragrid [91] (in the USA) and the world wide grid [92] (based in USA, Asia and Europe), it might soon be the case that one VO can be part of more then one such Grid system. In which case, users can submit jobs from one grid to another, which reduces the deployment efforts and provides users with a larger pool of resources.

After the completion of the initial SAM-Grid/LCG project, it was deemed a success by the Dzero experiment and a similar project, based on the development work done towards the SAM-Grid/LCG project, was undertaken to further increase the resource pool available to the Dzero users by making SAM-Grid and Open Science Grid (OSG) interoperable. It is our belief that many more such projects will be undertaken soon.

The rest of the thesis is organized as follows. Chapter 4 presents the current deployment model used by SAM-Grid. Chapter 5 presents the architecture and deployment model of the SAM-Grid/LCG interoperability project. Chapter 6 presents the integration of MyProxy with the SAM-Grid infrastructure. Chapter 7 presents additional deployment issues and solutions. Chapter 8 presents some results and parameters regarding the amount of success archived. Chapter 9 presents the conclusions and future work.

CHAPTER 4

CURRENT DEPLOYMENT MODEL OF SAM-Grid

The SAM-Grid architecture can be divided into four parts, the client site, the submission site, the SAM stations and the execution site. This chapter discusses the various components and describes the JIM and SAM architecture involved in these four parts. SAM as has been explained in chapter 2 was built to handle data for both CDF and Dzero experiments. JIM was built around Globus and Condor.

4.1 Components of the SAM-Grid System

1. Client Site: The client site is used by the users to submit their jobs. It does not need to have a high connectivity to the network. It only requires to be connected to the network during job submission. The client site has a minimal installation of JIM and SAM software. The JIM software on the client site is generally a collection of wrappers around Globus and Condor job submission tools. The JIM client takes a user provided SAM-Grid job description file (JDF) and converts it into a condor JDF. This is mainly done because the condor JDF is complex and the users do not need to provide all the details for a job type. The users specify the job type and some parameters, and based on this information the JIM client generates additional parameters which are then converted into a condor JDF. This condor JDF is then used to

submit the job to the submission site. During the phase of converting the JDF from SAM-Grid JDF to condor JDF the SAM component at the client side kicks in and checks to ensure that the datasets supplied by the users are consistent with the data available in the databases.

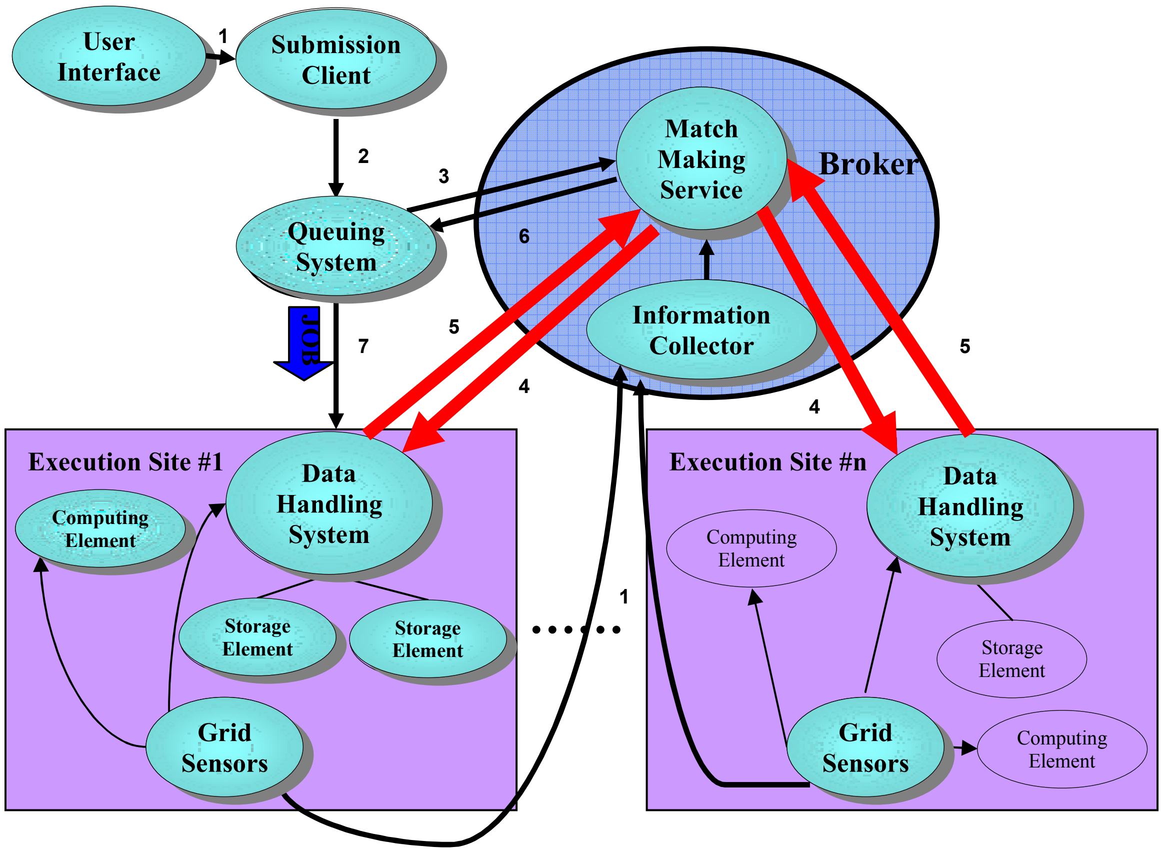


Figure 4.1 The job management architecture in SAM-Grid.1, 2- Jobs are submitted while resources are being advertised. 3- Match making services matched the jobs. 4, 5- Ranking functions retrieve additional information about the sites. 6, 7- Resource is selected and job is scheduled. [93]

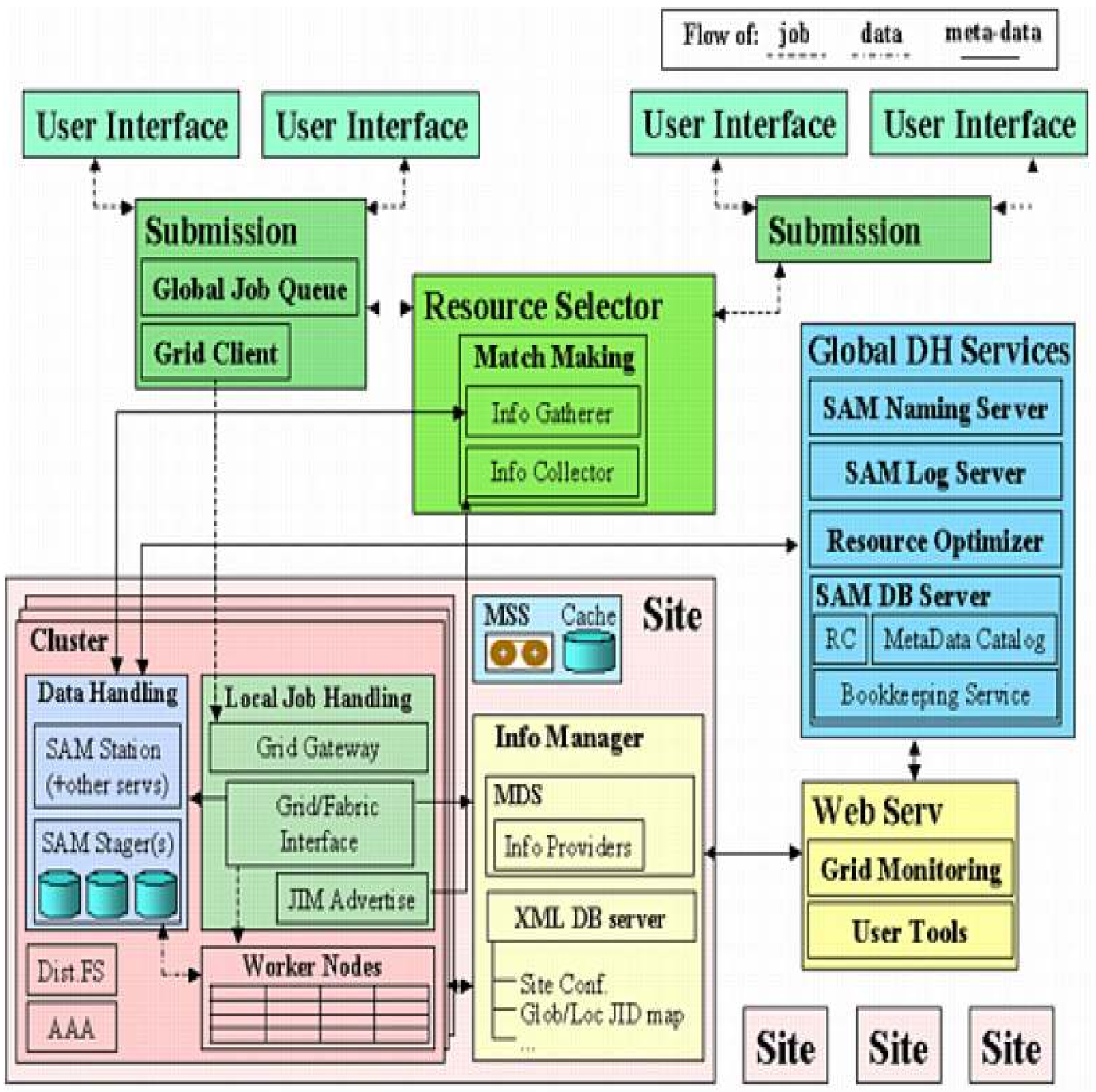


Figure 4.2 The SAM-Grid architecture decomposed into Fabric and Global services. Fabric services, local in nature, are shown in the pink boxes labeled “Site”. Global services are represented by the remaining boxes. The diagram also shows the division of the SAM-Grid in three components. The Job Management, in green, is composed of a Local Job handling service per site, a central Resource Selector, dozens of Submission services, and hundreds of User Interfaces. The Data Handling component, in blue, has Data Handling services at each site, and semi-central Global Data Handling Services. The Information Management, in yellow, is mainly constituted by Fabric services, interacting with information visualization mechanisms. Different types of arrows show the flow of jobs, data, and meta-data, tying together services and components. [71]

2. Submission Site: The submission sites are part of the JIM infrastructure and are responsible for the actual submission of the jobs. The submission site contains the Condor Master, Condor Collector, Condor Negotiator and Condor Scheduler demons. All these demons are part of the Condor Toolkit and provide the match making service (as explained in Chapter 1).

- Condor Scheduler Daemon: The scheduler demon is responsible for collecting all the jobs submitted from the user and do the actual task of scheduling them to the selected resources.

- Condor Collector Daemon: The collector demon is responsible for collecting the job and resource classads and providing them to the negotiator. The resource classads contain parameters such as the URL of the Globus gatekeeper, the amount of storage space left, current number of jobs in submission state and the maximum number of jobs that can be run at any point.

- Condor Negotiator Daemon: The negotiator demon is responsible for matchmaking. It takes the job and resource classads from the collector demon and uses the information to provide matchmaking (see chapter 1.)

- Condor Master Daemon: The basic function of the master demon is to monitor the other demons. If any demon crashes then it is the responsibility of the master demon to restart the demon.

3. Execution Site: The execution site is the site where the job is executed. It has the following components:

- Gatekeeper: This is the entry point that is presented by the site to the grid. Any job which needs to be executed at the site must enter using this entry point. Condor creates a Globus-RSL, which is used to submit a job to the gatekeeper running at this gatekeeper node.

- Job Managers: After a job enters the execution site, it is given to the job manager. The job manager must be given a Globus-RSL file which it understands. Based on this RSL, the correct type of job adapter is chosen and the control is passed to the job adapter. The job adapter is responsible to understand the various arguments that are passed to it as part of the RSL and then set the environment for the job, submit the job to the batch system, poll the batch system regularly to update the status of the job in an XML database and on the completion of the job pass the exit codes along with the output and log files produced by the job back to the user.

- Batch System: This is a component that must be present on all execution sites. A batch system is responsible for executing the job on the site. The batch system in effect brings all the resources available at an execution site together to present a single view to the user. The batch system has a head node from where the jobs are assigned to individual worker nodes. The batch system is responsible for executing the job on the cluster and getting the output and logs back to the head node. Common examples of batch systems are PBS, Condor, Torque, SGE and LSF.

- Batch Adapters: The job managers need to interface to the batch system using a standard set of tools. But there are various different types of batch system available. Each of them have different tools by which they submit jobs to the worker nodes, they check the status of jobs and get the output of jobs from worker nodes. Therefore, to allow the job managers to interface with different batch systems, we introduce a layer known as batch adapters. The purpose of this layer is to free the job managers from keeping knowledge of individual batch systems, by acting as an abstraction layer between the job managers and batch systems. The job mangers interact with the batch adapters using a standard set of tools, and the batch adapter then interacts with the batch system to submit the job. The batch adapters have complete information about the batch system. More information is available at [94]. The complete view of the Grid to Fabric Job submission can be seen in figure 4.3 [94].

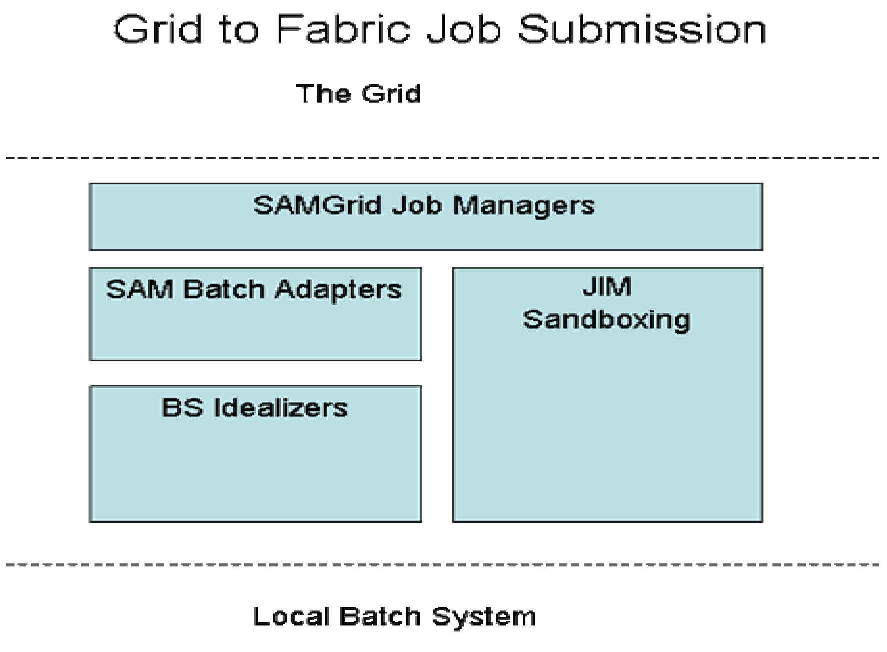


Figure 4.3 The mini-architecture of the Grid-Fabric Interface job submission service suite

- Batch system idealizers: The batch adapters also have another component which is known as batch system idealizers. The objective of these idealizers is that all batch systems do not have the functionality that is required. Such as retries in lookup commands for certain batch systems, generation of easy to parse output (batch system commands return output that is usually too terse or too verbose), compensation for confusing exit status from batch system commands. We use the batch system idealizers to overcome these limitations and to provide additional features like scratch management and explicit preference or avoidance of nodes that may or may not be well suited for our jobs.

- XML Databases: At every execution site, a XML database is deployed. The objective of this database is to store the job monitoring information and the information of various products which are installed at the site. The central monitoring service can then pull information about the jobs from these databases.

- Sandboxing: At the head node of each execution site we have a sandbox area, where all the files related to a job are stored. These include both the files which are explicitly specified by the user and the files which are implicit to the job type, such as X509 user proxy, the configuration of products, file transfer clients. The files are then bootstrapped into a bootstrapping script which is compressed. This compressed bootstrapping script is then passed and extracted at the worker nodes. The control is then passed to the script to begin execution of the job.

- Information Providers: This is a part of the JIM software. The information providers run a GSI enabled LDAP service, which is used to provide some information about the site. The information is pulled by the central monitoring site. This is included here for the sake of completeness and does not play a major role in the operations of SAM-Grid.

- SAM Tools: At the execution site we deploy a set of SAM tools which are used by the worker nodes to get the data from the SAM station. The SAM stations get the data from servers based in Fermilab. During the execution of the job the input/output data is transferred between the worker nodes and SAM stations.

- GridFTP: The GridFTP is a GSI-enabled FTP protocol. The GridFTP server runs at the head node of the execution site and the clients are sent to the worker node as part of the file which is created in the sandbox area for each job. The GridFTP client is used at the worker node to contact the server at the head node and then download the SAM tools. These SAM tools are useful in getting the remaining files from the station.

- Advertise: The information about the execution site has to be periodically sent to the submission site. This information is used to do matchmaking. The site administrator enters the configuration of the system and the jim_advertise product periodically updates this information and sends a Condor classad to the collector at the submission site.

4. SAM Station: The SAM station is the entry point for the data to the execution site. Along with the stager and FSS it forms the SAM components that are deployed at the execution site. (Refer to chapter 2.)

4.2 Flow of control during the lifetime of a Job

The Job starts at the client side, where the user creates the job using the SAM-Grid JDF. The client then takes the JDF and using the parameters specified looks up the requested files in SAM. The client also ensures that the user's proxy has a sufficient lifetime so as to ensure that the jobs do not fail due to a proxy failure. The client then converts the SAM-Grid JDF to a Condor JDF, and submits the Condor JDF to the submission site. The client can be configured to submit to any submission site, which he is authorized to use. There are currently 6 submission sites in operation.

At the submission site the Condor Scheduler demon accepts the request from the client and submits it to Condor Negotiator for matchmaking. The Condor Negotiator gets the list of resources from the Condor Collector demon and matches the job to a resource. The name and information of the matched job is then given to the Condor Scheduler, which then submits the job to the site where the job is executed.

At the execution site, first the user has to be authenticated and authorized by the gatekeeper. After this process, the job is given to a job manager instance which is instantiated by the gatekeeper. This job manager instance can understand the parameters that are passed to it. Based on these parameters, the job manager instance then sets up the environment for the job, starts a project to get the files that are required by the job from SAM to its SAM station cache, does sandboxing of the job, and creates a self executable of the job which can be executed on the worker node. It must be noted that this self executable contains all the files which are required for the execution of the job such as, the user proxy, the configuration files, and the GridFTP executables. After the above processes are completed, the job is submitted to the batch system, which is interfaced to the job managers via the batch adapters and batch idealizers. The figure 4.4 [72] describes the above process.

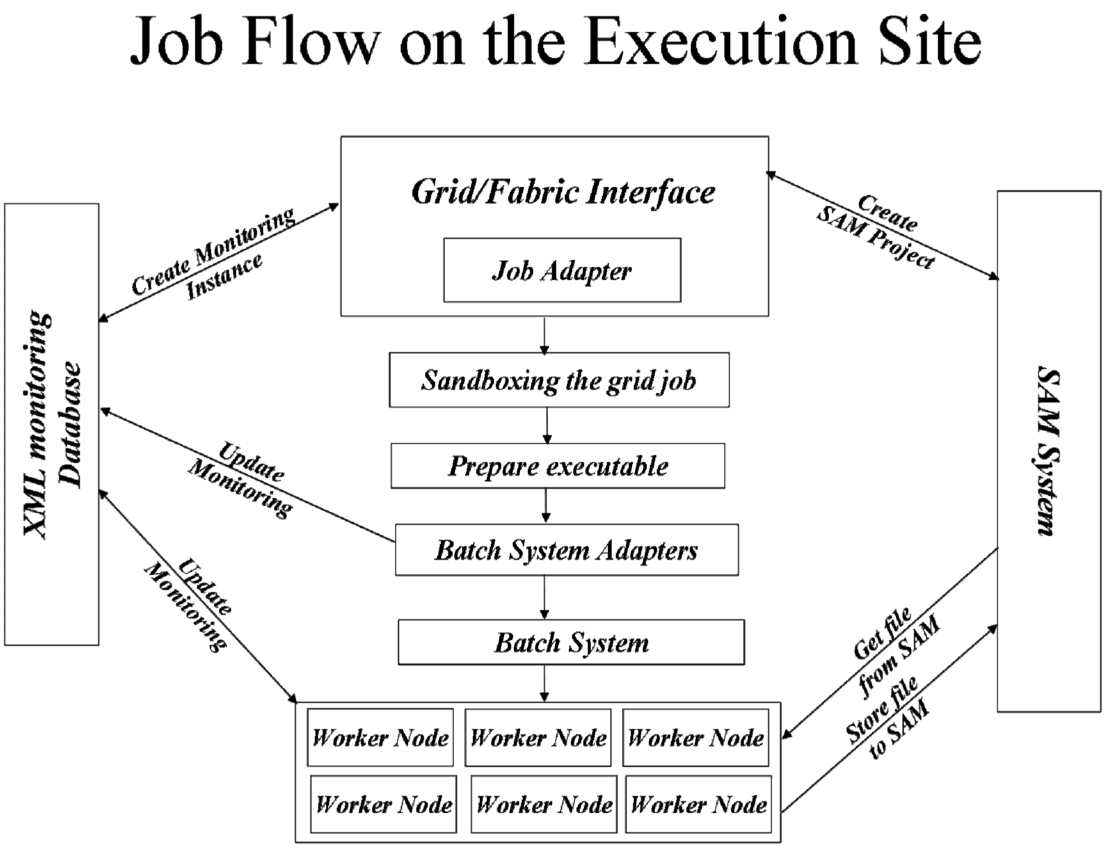


Figure 4.4 Flow of Job Control at the execution site

Each grid job is divided at the job manager level into one or more batch jobs. These batch jobs are given unique identifiers, which are used to monitor and poll the jobs during the execution of the jobs. Once the job lands on the worker node the self extracting executable is launched. This starts the process of actual execution of job. During the execution additional files are downloaded from the head node and from the SAM station. The process of execution of a reconstruction job at the worker node is shown in the figure 4.5 [72].

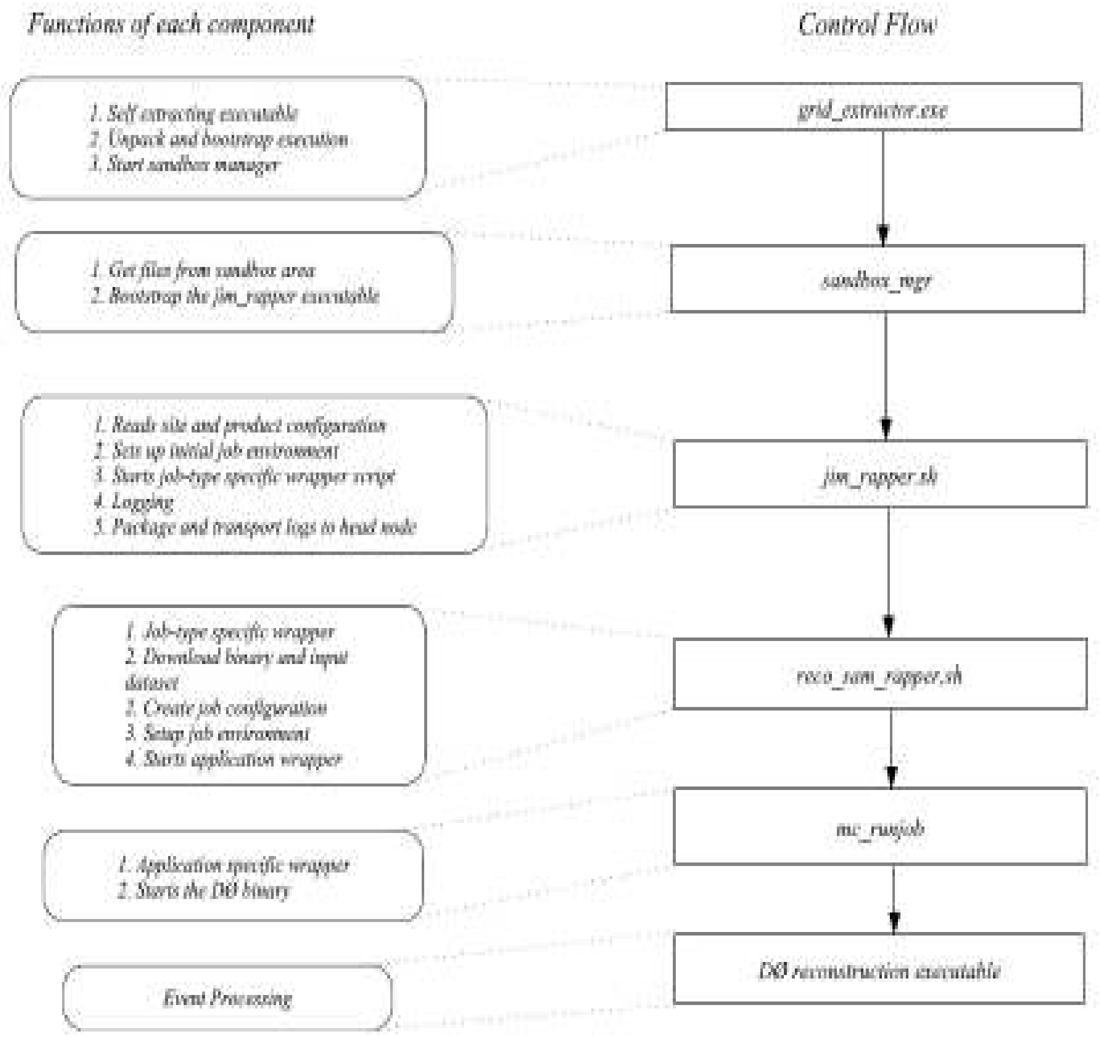

Functions of each component
Control Flow
1. Self extracting executable
2. Unpack and bootstrap execution
3. Start sandbox manager
grid_extractor.exe
1. Get files from sandbox area
2. Bootstrap the jim_rapper executable
sandbox_mgr
1. Reads site and product configuration
2. Sets up initial job environment
3. Starts job-type specific wrapper script
4. Logging
5. Package and transport logs to head node
jim_rapper.sh
1. Job-type specific wrapper
2. Download binary and input dataset
2. Create job configuration
3. Setup job environment
4. Starts application wrapper
reco_sam_rapper.sh
1. Application specific wrapper
2. Starts the DØ binary
mc_runjob
Event Processing
DØ reconstruction executable

Figure 4.5 The Control flow during the execution of a reconstruction job at the worker node.

After the completion of the executables the output is stored back in SAM using the FSS stager that is present at the head node.

CHAPTER 5

SAM-Grid/LCG INTEROPERABILITY DEPLOYMENT MODEL

5.1 Introduction

As has been described in the earlier chapters the SAM-Grid system is an integrated grid-solution which provides data, job and information management infrastructure to the Dzero users. The main objective of undertaking the SAM-Grid/LCG interoperability project is to increase the resource pool of the SAM-Grid system, while reducing the deployment efforts of SAM-Grid system. This project also enables the users to retain the user-friendliness of the SAM-Grid system and provides the LCG project with data intensive applications of a running experiment, which will help in improving the infrastructure for the LHC experiments.

The project is centered on job "forwarding" nodes, which receives jobs prepared by the SAM-Grid and submits them to LCG. The concept of job forwarding is similar to that of data forwarding in IP routers. In this chapter we discuss the architecture of the system, the issues of scalability and service accessibility.

The main features of the SAM-Grid system that must be preserved are:

1. Reliable data storage, from the detector and data processing facilities around the world.

2. Data distribution between all the collaborating institutions.

3. Cataloguing of data, which is generally described by the meta data associated with the files.

4. Job environment preparation including dynamic software installation, configuration management and workflow management.

5. Job aggregation i.e. aggregating a number of batch jobs into one single grid job.

5.2 Architecture

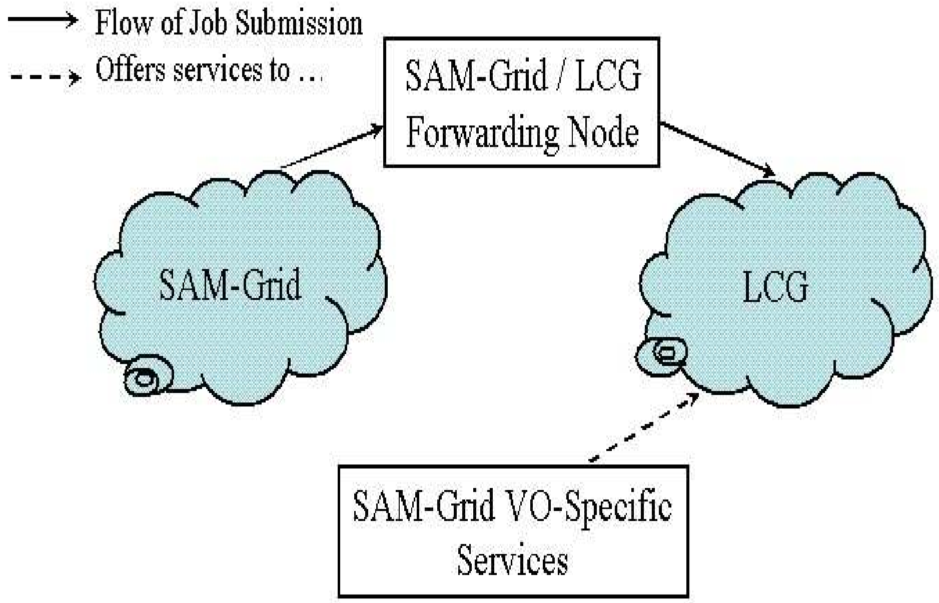


Figure 5.1 A high-level diagram of the SAM-Grid to LCG job forwarding architecture

As can be seen from figure 5.1 [95], the system is built around "forwarding nodes". These forwarding nodes act as an interface between SAM-Grid and LCG. From the SAM-Grid point of view the forwarding node acts as the head node or the gateway of an execution site. The forwarding nodes have a LCG user interface available at them. Jobs are submitted to SAM-Grid and are in turn given to the forwarding node. The forwarding node then submits the job to LCG systems, using the user interface present at the node. Within the LCG system, the jobs are first sent to the LCG resource broker and are in turn submitted to the LCG resources. A SAM installation offers remote data handling services to jobs running on LCG. The multiplicity of resources and services are shown in figure 5.2 [95].

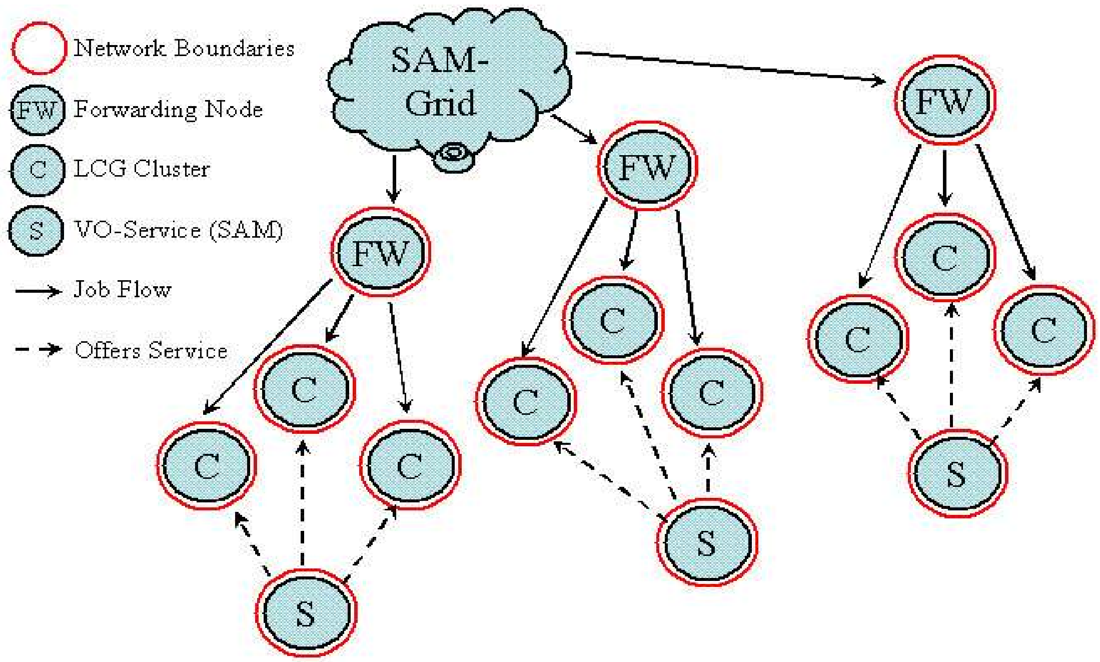


Figure 5.2 Multiplicity diagram of the forwarding architecture

As can be seen from figure 5.2, there are multiple forwarding nodes and sam stations involved in this system. This is mainly done to tackle issues of service accessibility, usability and scalability.

5.3 Job flow in the SAM-Grid/LCG model

The job originates on the client side, when a request is specified by the user. This request is then passed to the submission site. At the submission site the classad is matched to a particular resource (in our case the matched resource is the "forwarding node"). The job is then sent to the gatekeeper running on the head node of the execution site. Since this head node is the "forwarding node", the job arrives at the forwarding node and all functions that generally are executed at the head node of the execution site are executed at the forwarding node. The job managers submit the job to the batch system. In our case the batch system is the LCG grid system. Therefore, the job manager submits the job to the LCG grid system via the batch adapter. The LCG user interface then submits the job to the LCG resource broker, which in turn matches the job to a site and submits the job to that particular site. At the site the job again enters the site via the gatekeeper and is then directly sent to the worker node. At the worker node the job starts executing. At the start of executing the job downloads additional executables and files from the head node. It then gets the data from a SAM station which is located at a different site. After the execution of the job it stores the output files back in SAM using the FSS at the station node.

One important functionality difference that should be noted at this point is, generally in batch systems the output is "push" based i.e. when the job finishes execution the output is pushed back to the head node. But in LCG systems the output is not pushed back to the forwarding node, rather the output is stored at the resource broker and has to be "pulled" by the job manager. Therefore, when the job has finished execution then the output log files are pulled by the forwarding nodes.

5.4 Requirements of the System on execution nodes

The SAM-Grid system tries to minimize the dependencies of the job on the worker nodes. The only requirements that are present are the presence of "tar" software which is distributed along with all versions of Linux and UNIX systems, the presence of an environment variable called TMP_DIR and the proper configuration of the system with respect to time and scratch area.

Another important issue that should be noted at this point is that there is no clear consensus on the issue of scratch management, i.e. is scratch management a responsibility of the site or the application? The Dzero applications have a requirement of 4GB of scratch space. It is preferable that the scratch space is local instead of NFS mounted, this is because the I/O is very intensive and often causes failures when NFS areas are used.

SAM-Grid does "smart management" of scratch space. We include an additional scratch management script along with the main job so that the script executes first and chooses the scratch area and then executes the job in that area. After the completion of the job the scratch area is cleaned up by this script. Possible choices of scratch area are made available using the LCG job managers and are used by the application.

5.5 Current State of the system

Currently there are two forwarding nodes in the SAM-Grid/LCG system. Both the forwarding nodes are present at Wuppertal, Germany. The SAM station that is currently being used is deployed at Lyon, France. The sites which are currently a part of the system are Wuppertal, Nikef, Clermont-Ferrand, CCIN2P3 (France), Prague, Lancaster, Manchester, Rutherford-Appleton Labs (U.K) and Imperial College.

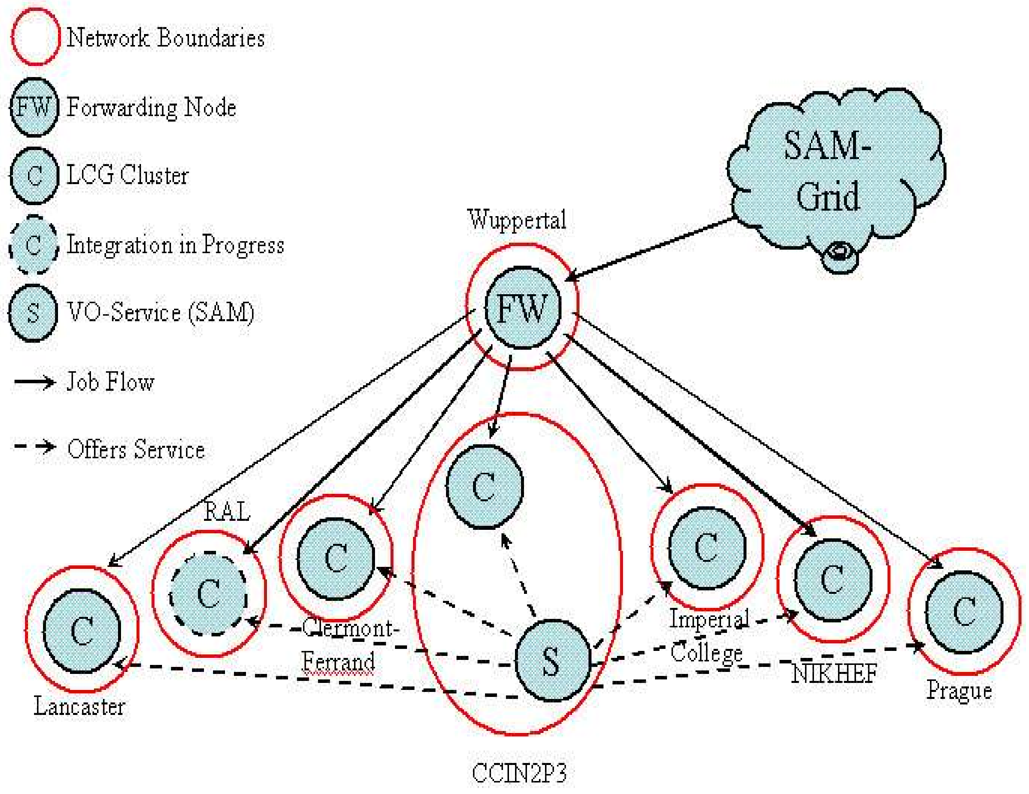


Figure 5.3 The forwarding architecture of the production system

5.6 Modifications to SAM-Grid JIM infrastructure and work done as part of this thesis

The SAM-Grid Model does not natively support job forwarding. Several modifications had to be done for SAM-Grid to support this feature. The changes are listed below.

1. Client side: Additional features needed to be added to the existing client tool to support job forwarding. Mainly we need to add parameters such as "lcg_requirement_string", "use_myproxy" and "myproxy_server". The lcg_requirement_string is used to provide the address of the LCG resource to the LCG resource broker. The use_myproxy and myproxy_server variables are used to generate and store the proxy of the user in a MyProxy server and the address of the server is passed to the forwarding node, where the server is contacted and a proxy is obtained. This proxy is used for job submission. An alternative to this solution, which was initially used, was that the user copies his proxy to a particular location and then the proxy file is shipped over the network using GSIFTP to the forwarding node. This method was deemed insecure because the proxy was being shipped over the network. Another modification that has been made is that the configuration of the client software is now extended to maintain information regarding the MyProxy server which can be used during job submission. We describe more details about the MyProxy integration in the next chapter.

2. Submission Site: No changes were required at the submission site.

3. Execution Site:

- At the execution site the job managers were modified to pull the output from the resource broker. As explained in the previous sections, the LCG system has a pull based output gathering model. That means that when the job finishes execution the user has to get the output of the job. Therefore, the job manager was modified to provide this functionality. The job manager regularly polls the status of the job and when the job enters the finished status the job manager pulls the output of the job.

- In order for the job manager to pull the output the batch adapters were modified to support this functionality.

- In SAM-Grid the job manager interfaces to the batch system using the batch adapter. Therefore the batch handler for LCG was written and modified over time to provide the submission of jobs to the LCG system, to gather output from the job, to poll the job at regular intervals. The initial batch adapter was written by our collaborators in France which provided some functionality with respect to job submission and job polling.

- The scratch management script was designed and implemented to provide scratch management on the LCG resources.

- One short coming of Globus is that when we use our proxy to submit the job then the proxy that is sent to the gatekeeper at the execution site becomes a limited delegated proxy and has limited privileges, and hence this proxy cannot be used for further job submission. Therefore, to overcome this problem we use MyProxy. During the job submission a fresh proxy is obtained from the MyProxy Server and this is used to submit the job. This functionality is added to the batch handler.

There were some modifications done to the SAM station software. These changes were not done as part of this thesis and are discussed here only for the sake of completeness.

SAM had to be modified to allow service accessibility to the jobs within private networks. SAM was further modified to accept TCP-based communications as UDP does not work well with this system over WAN. The sites with the SAM station, stager and FSS must allow all incoming network traffic from forwarding nodes as well as all LCG clusters. The SAM system was modified to provide port range control.

After these changes jobs were successfully run on the LCG system.

CHAPTER 6

INTEGRATION WITH MyProxy

6.1 Introduction

In grid computing, there is a need for secure communication between various components of the grid. In the current Globus middleware that we are using, this security is provided by public key infrastructure based on X509 certificates. This works well with the traditional models of the grid, such as the general deployment model of SAM-Grid as explained in chapter 4. But the current GSI architecture lack one important feature. In the current model, Globus does not allow a user to submit a job to the grid from another job in the grid. In the SAM-Grid/LCG model we need this feature, for this purpose we use MyProxy.

6.2 Overview of Grid Security

In the Globus middleware the X509 public key infrastructure is used to provide Grid security. This is a part of the GSI, which is primarily used to provide security in the grid environment. The main motivations behind GSI are:

1. The need for secure communications between resources in the grid.
2. The need to support security across organizations, eliminating the need for a centralized security system

3. The need to support “single sign-on” for users. Once the users generate the proxies, the proxies are delegated across multiple sites which they choose to use eliminating the need for uses to sign-on at each site.

GSI is based on public-private key infrastructure. The user generates his public and private keys and then sends his public key to the Certificate Authority (CA). The CA is a third party which is trusted by all resources in the VO. The CA creates a certificate and binds it to the user’s public key and identity information embedded within the certificate itself. In case of GSI, the certificate contains four main pieces of information:

1. A subject name, which identifies the user or resource.
2. The public key belonging to the subject.
3. The identity of the CA that has signed the certificate.
4. The digital signature of the CA.

Once the user obtains the certificate from the CA, the user can use it to generate a proxy. The proxy is used to provide authentication and single sign-on. The proxy that is created by the user contains the subject of the user, the issuer of the user certificate, the start and expire time of the proxy. The proxy is generated by the user using his certificate. The proxy certificate contains a new certificate and a private key. This new certificate is signed by the original private key of the user and not the CA.

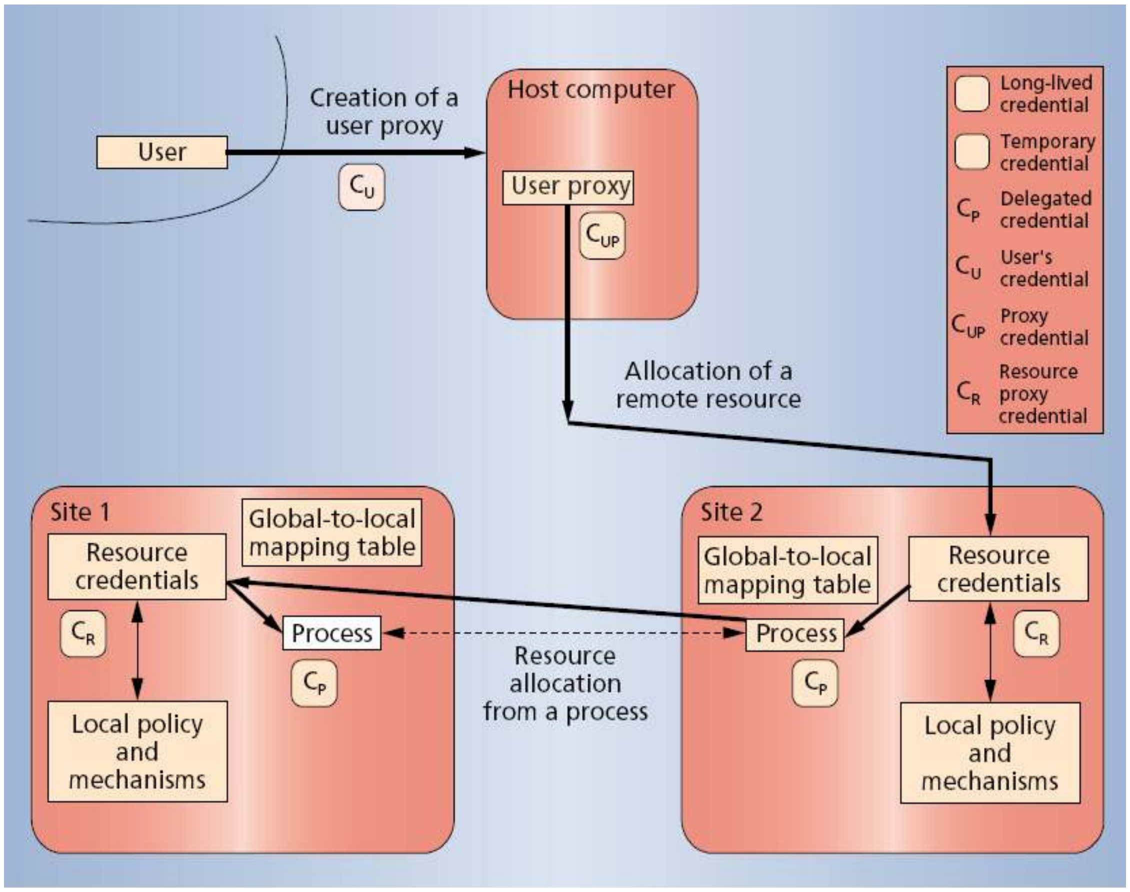

Creation of a user proxy
User
C_U
Host computer
User proxy
C_{UP}
Allocation of a remote resource
Long-lived credential
Temporary credential
C_P Delegated credential
C_U User's credential
C_{UP} Proxy credential
C_R Resource proxy credential
Site 1
Global-to-local mapping table
Resource credentials
C_R
Process
C_P
Local policy and mechanisms
Resource allocation from a process
Site 2
Global-to-local mapping table
Resource credentials
Process
C_P
C_R
Local policy and mechanisms

Figure 6.1 The basic operations that GSI supports. "Following the dark line from the top left-hand corner, we first see user authentication via public-key mechanisms applied to the user's credential (CU), followed by creation of a temporary user proxy credential (CUP), then subsequent requests to remote resources, represented by resource proxies holding resource proxy credentials (CR), and finally authorization and global-to-local identity mapping at an individual site, resulting in the creation of a remote process at Site 2, with its own delegated credential (CP). We also see how such a remote process can use its delegated credential to initiate further requests to other sites (in this case, a process creation request to Site) and engage in authenticated interprocess communication (the dashed line)." [96]

This proxy credentials can be used by the user to submit the jobs. When the user submits a job using this proxy credentials then the public keys are sent to the gatekeeper and the gatekeeper uses these public keys to authenticate the user. But we can't use these public keys to further submit jobs from the gatekeeper. This is a feature of Globus i.e. it does not allow one grid job to submit another grid job. This is exactly our requirement. We want to submit one grid job (LCG grid job) from another grid job (SAM-Grid grid job). This is mainly because Globus does not by default trust a middle man. But in our case we can convince our users that the MyProxy repository is safe against malicious hackers and is well maintained. Therefore, we need a repository where we can store our proxies securely and use these proxies to submit the jobs.

6.3 MyProxy and SAM-Grid

MyProxy can be simply described as a credential management system. It provides the users with a repository where the uses can store their credentials and retrieve them from anywhere. It ensures that the users do not need to ship their proxies across the network.

The two main components of the MyProxy system are:

1. The client side: At the client side the user's credentials are present. These credentials are used by the MyProxy tools to generate additional credentials and store them on MyProxy server. The MyProxy tools typically look for the users credentials in the $HOME/.Globus/ directory. After generation of proxy, we store it using encrypted communication mechanisms which are based on GSI's message confidentiality mechanisms like SSL. For the purpose of this project we provide the users with tools to store and retrieve the proxy. This was done to provide more robustness and as a means of protecting our users from mistakes. In these tools we enable the storing of information in an interactive manner as the users need to provide their pass phrase while storing the proxy. But while storing, we also specify that the

proxy can only be retrieved by providing a delegated credential for authentication. In other words we disable the retrieval of proxies based on passwords. While retrieving the credentials the uses has to provide with a copy of the delegated credentials that is received at the gatekeeper.

2. The MyProxy Server: The server is hosted on a machine which is very well protected. The servers are generally trusted by all the users and resource providers. The access to these machines is generally limited by using Kerberos or SSH-keys. The server also needs a host GSI certificate and must have a GSI setup. It uses GSI message confidentiality mechanisms like SSL to communicate to the clients. In our case we have one such machine available at "fermigrid4.fnal.gov" which we use as our MyProxy Server. Another important point that should be noted is that the server must be accessible by users from every where. This is a critical point as when we were using the MyProxy servers on "fermigrid1.fnal.gov" we could use it within the Fermilab network but out users were unable to use it from outside the network.

An alternate solution which was initially used and later replaced with the MyProxy solution was to ship the proxy using GSI-enabled FTP. This solution was rejected because even thou we are transferring the proxy in an encrypted format we are still shipping the users private key. This is against the basic principle of computer security, which states that a user's private key must never be shipped over the network. In this case we trust that the MyProxy server is secure and all the users and resource providers trust the MyProxy server.

6.4 Future Work

MyProxy can be used in the future to provide another important functionality that is currently available but not used. The users can have jobs which run for a long time but will fail because the proxy expires. This problem can be solved by using MyProxy, but currently there is no need for such functionality, as the users jobs get over within a weeks time and the users generate their proxy for a week.

CHAPTER 7

DEPLOYMENT ISSUES AND SOLUTIONS

7.1 Resubmission

In LCG systems when a job fails, then it is automatically resubmitted. Some jobs (such as “Reconstruction”) should never be resubmitted in case of failure. They must be processed as part of a separate activity. We experienced problems overriding this feature, because this feature is the default action in EDG. This feature can be expressed in two places, as a part of the job description file and at the configuration of the user interface. This was later suppressed by changing the configuration in the user interface. While this feature is not desirable for “reconstruction”, it is acceptable for “Monte Carlo” applications. And this feature was used regularly at for the Monte Carlo applications. This results in a higher success rate as the jobs which fail are resubmitted and after some resubmissions it eventually succeeds.

7.2 Broker Input Sandbox Space Management

On some LCG brokers, disk space was not properly cleaned up. Hence, administrative intervention was required to resume the job submission activity. This is still a problem and we are currently working on a solution for this problem.

7.3 Output Gathering

When the job finishes execution, the output is stored at the resource broker. It is then pulled from the resource broker by the SAM-Grid job managers. But there have been cases when the output gathering failed because the user's proxy has expired. This results in the accumulation of output at the broker. After a certain limit the broker will not accept any new jobs until the output is cleared up. The output also expires in 14 days. This problem was solved by extending the life time of the proxy that is obtained from the MyProxy server.

7.4 Job Failure Analysis

The output of failed jobs is ambiguous, in particular we were not able to get the output of jobs which were aborted and had to rely on getting the logging information of the jobs to understand the reasons of failure.

7.5 Local Cluster Configuration

When nodes on a cluster are misconfigured then we experience a significant drop the in success rates of the jobs. These worker nodes fail the job and then accept the next batch job and fail that too. This way a lot of jobs end up in the failed state. One way to rectify this problem was by using the resubmission parameter. Common miss-configurations include time asynchrony, scratch management and disk space.

7.6 SAM data handling issues

SAM uses UDP to communicate within the execution site. But in our case the UDP packets are often lost which result in a failure in getting data to and from the worker nodes to the stations. This was changed to use TCP, which works well over WAN.

7.7 Accountability

In Dzero, institutions get credit for the computing that they do at their site for the collaboration. Generally the sites which produce say, "Monte Carlo" also do the merging of all their generated files. The final merged file contains the name of the execution site, which is used for accountability purpose. But, in our model the execution site contains various institutions. Therefore, we changed the way the final merged files were named by giving it the name of the computing element, which is generally the name of the cluster, rather then the name of the execution site itself.

7.8 Scalability

The scalability of the LCG forwarding node is limited by the number of jobs that are currently running on LCG via that node. This issue becomes important when the jobs are submitted at worker nodes which are further away from the forwarding node in terms of the time taken for a polling operation to complete. The polling operations are done to maintain a status of the job, currently the polling interval is set to 5 minutes. As the number of jobs increase, due to this polling and, also due to additional operations like job submission and output gathering the load on the machine increases. To counter this problem we deploy multiple forwarding nodes. Currently there are two forwarding nodes in operation.

CHAPTER 8

RESULTS AND PARAMETERS

The SAM-GRID/LCG system was used to do “refixing” and Monte Carlo applications. “Refixing” applications are similar to reprocessing applications. These applications were created to fix some problems that were observed in the data that were obtained after the reprocessing effort. The results that are presented are obtained by querying the Xindices database at the Forwarding nodes.

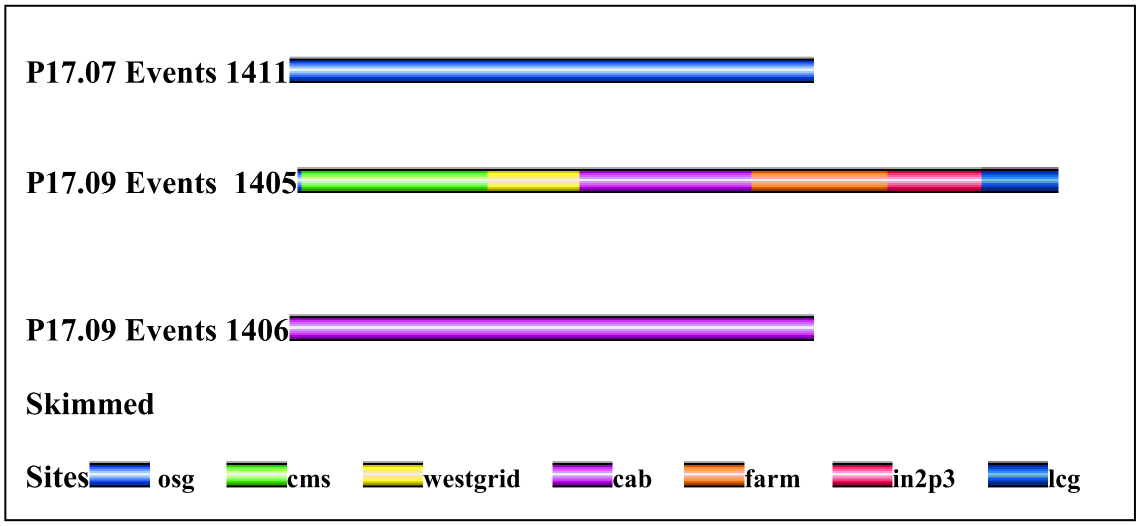


Figure 8.1 P17.09.03 Refix Status as of Mar-2006(all sites)

Figure 8.1 [98], shows the percentage of computation that was done by each of the sites involved in the effort. As can be seen from figure 8.1, approximately 8-10% of the re-fixing effort was done via the SAM-Grid/LCG system. The efficiency of the system, which we define as the number of times the input data was successfully downloaded and then output was stored back in SAM is 70%, this can be seen in figure 8.2. These jobs also include test and certification jobs. The number of files that are generated by the system should be equal to or less then the number of input files.

Name of the cluster	**Number of jobs submitted.**
Lancaster	**4697**
Prague	**1687**
Clermont-Ferrand	**1267**
Imperial College	**1223**
NIKHEF	**287**
Wuppertal	**502**

Total Number of submissions: 9663
Total Output Generated by the system: 6754

Efficiency: 70%

Figure 8.2 Some Numbers with respect to each site

Resource/Job-manager-name	**Job type**	**# of LCG Jobs**	**# of Successful Job Completions**	**# of Jobs Failed**	**Avg time for Starting**	**Avg time to Obtain job files**	**Number of events Produced**
cclcgceli02.in2p3.fr:2119/jobmanager-bqs-long	MC min bias 24380	86	85	1	16092.3 sec	493 sec	2125 events
(IN2P3)							
clrlcgce02.in2p3.fr:2119/jobmanager-lcgpbs-dzero (Clermont Ferrand)	MC min bias 24380	86	41	45	662.97 secs	600 sec	1025 events
tbn20.nikhef.nl:2119/jobmanager-pbs-qlong (NIKHEF)	MC min bias 24380	86	84	2	5455.36 sec	6033 sec	2100 events
cclcgceli02.in2p3.fr:2119/jobmanager-bqs-long (IN2P3)	MC min bias & phase 25666	24	24	0	37084 sec	141 sec	600 events
clrlcgce02.in2p3.fr:2119/jobmanager-lcgpbs-dzero (Clermont Ferrand)	MC min bias & phase 25666	24	3	21	2976.6 sec	127 sec	75 events
tbn20.nikhef.nl:2119/jobmanager-pbs-qshort (NIKHEF)	MC min bias & phase 25666	32	32	0	12662 sec	7980 sec	800 events
fal-pygrid-18.lancs.ac.uk:2119/jobmanager-lcgpbs-dzero	MC min bias 24380	86	85	1	930	8199	2125
fal-pygrid-18.lancs.ac.uk:2119/jobmanager-lcgpbs-dzero (Lancaster)	MC min bias & phase 25666	32	30	2	1043	24089	750

Figure 8.3 Results of stress tests obtained on all the systems.

As can be seen from figure 8.3, the average starting time of the jobs is different for almost all the systems. This is mainly because the starting time of the job depends on parameters such as network bandwidth, resource availability at the site and the priority of the VO at the site. We have observed that the time for a job to start also depends on the policies at the cluster site. The average time to get the job files depends on the network speed between the worker nodes and the SAM station. As can be seen from the results, the time to get the job files is shortest for IN2P3 and Clermont-Ferrand as they are closer to the SAM station which is present in IN2P3. The transfer times are the highest for NIKHEF as it has the slowest connection between the SAM station and worker nodes. These results show that the distance of the execution site from the SAM Station effects the average time to get the data for the job. Therefore, it is advantageous to have stations near different sites.

CHAPTER 9

CONCLUSION AND FUTURE WORK

The users of SAM-Grid have access to the pool of LCG resources via the "interoperability" system described in this thesis. This mechanism increases the pool of available resources without increasing the system deployment and support cost. The SAM-Grid is responsible for job preparation, for data handling and for interfacing the users to the grid. LCG is responsible for job handling, resource selection and scheduling.

The Dzero users have used this system for Re-fixing activities and are currently planning to use it for Monte Carlo simulation activities. We have been able to contribute to the experiment, by doing around 8-10% of the "refixing" activity on LCG. From the results we also observe that increasing the distance from the SAM station does impact the time to get the data files for the jobs. Therefore, we conclude that to increase the efficiency of the system it is better to increase the number of stations.

Based on this design and infrastructure, we have been able to make SAM-Grid "interoperable" with OSG.

In the future, we would expect this model to be more widely used such that users can submit their jobs from one grid system to another. This is advantageous to everybody because it reduces the amount of software on each cluster, which in turn reduces the deployment cost for the experiment and frees the administrator from having to take care of multiple grid software.

This is a first step towards our view of the future where a cluster can be a part of one grid system but at the same time support multiple VOs.

REFERENCES

1. The home page, Dzero Experiment: http://www-d0.fnal.gov/ ,2006

2. The home page, CDF Experiment: http://www-cdf.fnal.gov/ ,2006

3. I. Foster, C. Kesselman, S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, International J. Supercomputer Applications, 15(3), 2001.

4. The BaBar Collab., D. Boutigny et al., “Technical Design Report”, SLAC-R-95-457.

5. The BELLE Collab., M. T. Cheng et al., “Technical Design Report”, KEK-Report 95-1.

6. The D0 Collab., “The D0 Upgrade: The Detector and its Physics”, Fermilab Pub-96/357-E.

7. CDF Collab., R. Blair et al., “The CDF II Detector Technical Design Report”, FERMILAB-Pub-96/390-E.

8. D. G. York, et al., “The Sloan Digital Sky Survey: Technical Summary”, The Astronomical Journal 120 (2000) 1579-1587

9. 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).

10. The Atlas Collaboration, “Atlas - Technical Proposal”, CERN/LHCC94-43, CERN, December 1994.
11. The CMS Collaboration, “The Compact Muon Solenoid Technical Proposal”, CERN/LHCC 9438, LHCC/P1 1994
12. The CMS Collaboration, “Computing Technical Proposal”, CERN/LHCC 96-45, (Geneva 1996)
13. The LHCb Collaboration, “LHCb: Technical Proposal”, CERNLHCC-98-004;
14. P. Saiz, L. Aphecetche, 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
15. The ALICE Collaboration, “ALICE Technical Proposal for A Large Ion Collider Experiment at the CERN LHC”, CERN/LHCC/95-71, 15 December 1995
16. I. Foster and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers; 1st edition (November 1998)
17. I. Foster, C. Kesselman and S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, International J. Supercomputer Applications, 15(3), 2001.
18. I. Foster, "What is the Grid?”, Grid Today, Vol.1 No. 6, July 22, 2002.
19. CERN, Gridcafe: http://gridcafe.web.cern.ch/gridcafe/whatisgrid/whatis.html , 2006

20. CERN: http://www.cern.ch/, 2006

21. FNAL: http://www.fnal.gov/ ,2006

22. BNL: http://www.bnl.gov/world/ , 2006

23. I. Foster and C. Kasselman, “Globus: A Metacomputing Infrastructure Toolkit”, International Journal of Supercomputer Applications, 11(2): 115-128, 1997

24. Globus Alliance: http://www.Globus.org/, 2006

25. Global Grid Forum: http://www.ggf.org/ , 2006

26. K. Keahey, V. Welch, “Fine-Grain Authorization for Resource Management in the Grid Environment.” Proceedings of Grid2002 Workshop, 2002.

27. V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, F. Siebenlist, “X.509 Proxy Certificates for Dynamic Delegation”, 3rd Annual PKI R&D Workshop, 2004.

28. V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman and S. Tuecke, “Security for Grid Services”, Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), 2003.

29. R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch,” A National-Scale Authentication Infrastructure”, IEEE Computer, 33(12):60-66, 2000.

30. W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster, “The Globus Striped GridFTP Framework and Server”, Proceedings of Super Computing 2005 (SC05), November 2005.

31. W. Allcock et al., "GridFTP: Protocol extensions to FTP for the Grid," GGF Document Series GFD.20, Apr. 2003

32. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, “Grid Information Services for Distributed Resource Sharing”, Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

33. X. Zhang and J. Schopf. “Performance Analysis of the Globus Toolkit Monitoring and Discovery Service, MDS2”, Proceedings of the International Workshop on Middleware Performance (MP 2004), part of the 23rd International Performance Computing and Communications Workshop (IPCCC), April 2004.

34. 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.

35. Platform Computing, ”PLATFORM GLOBUS TOOLKIT: Opensource, commercially supported toolkit for building grids”, On line http://www.platform.com/products/Globus/ , 2006

36. Sun Grid Engine: http://wwws.sun.com/software/gridware , 2006

37. United Devices: http://www.uniteddevices.com/, 2006

38. Parabon: http://www.parabon.com ,2006

39. ProcessTree: http://www.processtree.com , Distributed Science Inc, Nov. 2000.

40. DataSynapse: http://www.datasynapse.com/, 2006

41. 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.

42. F. Berman and R. Wolski, “The AppLeS Project: A Status Report, Proceedings of the 8th NEC Research Symposium”, Berlin, Germany, May 1997.

43. J. Novotny, “The Grid Portal Development Kit”, Concurrency: Practice and Experience 2000; 00:1-7

44. 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.

45. 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.

46. 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.

47. 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

48. 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

49. 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.

50. EGEE - Enabling Grids for E-science in Europe: http://egeeintranet.web.cern.ch/egee-intranet/gateway.html ,2006

51. 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

52. 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.

53. The Grid2003 Project, "The Grid2003 Production Grid: Principles and Practice", iVDGL, Technical Report, 2004: On line www.ivdgl.org/grid2003.

54. The Particle Physics Data Grid, "GRID2003 Lessons Learned", PPDG Document 37, http://www.ppdg.net

55. Francesco Giacomini, Francesco Prelz, Massimo Sgaravatto, Igor Terekhov, Gabriele Garzoglio, and Todd Tannenbaum, "Planning on the Grid: A Status Report", DRAFT PPDG-20; EDG WP1, D0-Grid, Condor-Project; 10/02

56. The LCG Project, http://lcg.web.cern.ch/LCG/, 2006

57. SAM-Grid project: http://www-d0.fnal.gov/computing/grid 2006 ,2006

58. 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 Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, Ca, USA, March 2003.

59. 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

60. 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

61. 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

62. 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 2003; to appear in Nuclear Instruments and Methods in Physics Research, Section A

63. SAM project: http://d0db.fnal.gov/sam ,2006

64. 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.

65. 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

66. 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

67. V. White et al., ”D0 Data Handling”, in Proceedings of Computing in High Energy and Nuclear Physics (CHEP01), Beijing, China, Sep. 2001

68. 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

69. 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

70. 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

71. Gabriele Garzoglio, “A Globally Distributed System for Job, Data and Information Handling for High Energy Physics”, Phd Thesis, DuPual University, Chicago, Illinois, 2006.

72. A. Rajendra, "Integration of the SAM-Grid Infrastructure to the DZero Data Reprocessing Effort", Thesis of Master in Computing Science, The University of Texas, Arlington, Dec. 2005

73. B. Balan, "Enhancements to the SAM-Grid Infrastructure", Thesis of Master in Computing Science, The University of Texas, Arlington, Dec. 2005

74. A. Nishandar, "Grid-Fabric Interface For Job Management In Sam-Grid, A Distributed Data Handling And Job Management System For High Energy Physics Experiments", Thesis of Master in Computing Science, The University of Texas, Arlington, Dec. 2004

75. S. Jain, "Abstracting the hetereogeneities of computational resources in the SAM-Grid to enable execution of high energy physics applications", Thesis of Master in Computing Science, The University of Texas, Arlington, Dec. 2004

76. 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, Dec 2003; published in Nuclear Instruments and Methods in Physics Research, Section A, 534:33-37,2004
77. Lee Lueking, "SAM Overview and Operation at D0", In Proceedings of the Computing in High Energy Physics. Fermilab, Computing in High Energy Physics, 2001.
78. Xindice website, http://xml.apache.org/xindice/ ,2006
79. 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
80. H.Stockinger, F.Dono,E.Laure, S.Muzzafar et al, "Grid Data Management in action", computing in high energy physics (CHEP 2003).
81. D.Bosio,J.Casey,A.Frohner,L.Guy et al, "next generation EU DataGrid Data management services", computing in high energy physics (CHEP2003).
82. LGuy, P.Kunszt,E.Laure, H.and K. Tockinger, "Replica Management in Data Grids" , GGF5, Edinburgh.

83. Wolfgang Hoschek, Javier Jaen-Martinez, Asad Samar, Heinz Stockinger, Kurt Stockinger, " Data Management in an International Data Grid Project", IEEE/ACM International Workshop on Grid Computing Grid'2000 - 17-20 December 2000 Bangalore, India

84. Bob Jones, "An Overview of the EGEE Project", 6th Thematic Workshop of the EU Network of Excellence DELOS, Cagliari, Italy, June 24-25, 2004.

85. Birger Koblitz, "Experiences with gLite", CHEP04, Interlaken, 29. September 2004.

86. Ian Bird, "LCG-1 deployment plan" , GridPP 7th Collaboration Meeting, 30 June - 2 July 2003

87. J. Novotny, S. Tuecke, and V. Welch. "An Online Credential Repository for the Grid: MyProxy", Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001, pages 104-111.

88. D. Kouril and J. Basney, "A Credential Renewal Service for Long-Running Jobs", Grid 2005 - 6th IEEE/ACM International Workshop on Grid Computing, Seattle, WA, November 13-14, 2005.

89. J. Basney, M. Humphrey, and V. Welch. "The MyProxy Online Credential Repository.", Software: Practice and Experience, Volume 35, Issue 9, July 2005, pages 801-816

90. OSG website, www.opensciencegrid.org , 2006.

91. TeraGrid website, http://www.teragrid.org , 2006.

92. World Wide Grid website, http://www.gridbus.org/ecogrid/wwg, 2006.

93. 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 Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, Ca, USA, March 2003

94. Nishandar, D. Levine, S. Jain, G. Garzoglio, I. Terekhov, "Extending the Cluster-Grid Interface Using Batch System Abstraction and Idealization", in Proceedings of Cluster Computing and Grid 2005 (CCGrid05), Cardiff, UK, May 2005

95. Gabriele Garzoglio et al, “The SAM-Grid/LCG interoperability system: A bridge between two grids”, CHEP 2006.

96. R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch, “A National-Scale Authentication Infrastructure” , IEEE Computer, 33(12):60-66, 2000.

97. Private communication from Dr. Yu, 2005

98. http://www-d0.fnal.gov/computing/reprocessing/ ,2006

BIOGRAPHICAL INFORMATION

Tummalapalli Sudhamsh Reddy joined the Masters program in Computer Science at the University of Texas at Arlington in the fall of 2003. He received his bachelor's degree from Vellore Engineering College (now Vellore Institute of Technology), Vellore, India, affiliated to Madras University, India. He worked at Fermi National Accelerator Laboratory, Batavia, IL as a software developer from May 2005 to May 2006. His academic and research interests include Grid Computing, Cluster Computing and scientific computing.