8-2009

Deploying and Maintaining a Campus Grid at Clemson University

Dru Sepulveda
Clemson University, analog.ghost@gmail.com

Deploying and Maintaining a Campus Grid at Clemson University

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Masters
Computer Science

---

by
Dru Sepulveda
May 2009

Accepted by:
Sebastien Goasguen, Committee Chair
Mark Smotherman
Steven Stevenson

Abstract

Many institutions have all the tools needed to create a local grid that aggregates commodity compute resources into an accessible grid service, while simultaneously maintaining user satisfaction and system security. In this thesis, the author presents a three-tiered strategy used at Clemson University to deploy and maintain a grid infrastructure by making resources available to both local and federated remote users for scientific research. Using this approach virtually no compute cycles are wasted. Usage trends and power consumption statistics collected from the Clemson campus grid are used as a reference for best-practices. The loosely-coupled components that comprise the campus grid work together to form a highly cohesive infrastructure that not only meets the computing needs of local users, but also helps to fill the needs of the scientific community at large. Experience gained from the deployment and management of this system may be adapted to other grid sites, allowing for the development of campus-wide, grid-connected cyberinfrastructures.

Dedication

This thesis is dedicated to my fiance Heather and my parents whose love, support, and guidance have given me the momentum to pursue my dreams.

Acknowledgments

The author would like to acknowledge Nell Beaty Kennedy, Matt Rector, and John Mark Smotherman for help and technical support with software deployment for the Clemson School of Computing and the university public labs. Sebastien Goasguen is the author's advisor and mentor for this research. The author would also like to thank Michael Fenn and Mike Murphy for Apache server administration and general advice. The main Condor Team contact for troubleshooting Condor was Ben Burnett. This research was partially funded by an IBM Faculty Award, an NSF grant OCI-753335, and the Open Science Grid.

Figure 1: Goasguen and his team, clockwise from lower right: John Mark Smotherman, Matt Rector, Nell Kennedy, Sebastien Goasguen, and Dru Sepulveda (front).

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The act of pooling resources and sharing the computational weight of research around the world is central to the idea of cyberinfrastructure (CI) [17] and is the motivation for many grid programs at academic institutions [51]. Condor [32], a high throughput batch system, provides a way of pooling resources into a usable service without the cost of new machines and has the added benefit of being well-maintained and well-documented by the Condor Research Group at the University of Wisconsin at Madison. Condor provided more than enough computational power for research at Clemson University during its first year of use, but due to excess computational power machine cycles were left unused. To address the aforementioned idle time, Condor has been combined with the Berkley Open Infrastructure for Networked Computing (BOINC) by using Condor Backfill, which donates the compute cycles that Condor does not use to projects such as World Community Grid (WCG), Einstein@home, and LHC@home. In under six months Clemson University has become one of the top contributors to BOINC projects in the world. As faculty and students at Clemson University learned about the benefits of using High Throughput Computing (HTC) to accomplish their research goals the pools usage increased, sometimes exceeding the capacity of the Clemson Condor pool. A Globus Gatekeeper and Condor job_router were deployed on the campus infrastructure to send overflow jobs to other Condor pools on the Open Science Grid (OSG) and to better share our resources with others, respectively. Figure 1.1 shows how all of the middleware components work together at Clemson University to form the campus grid. The combination of these three systems into a functional grid is accomplished through sufficient staffing and open communication both locally between departments and globally between federated grid sites.

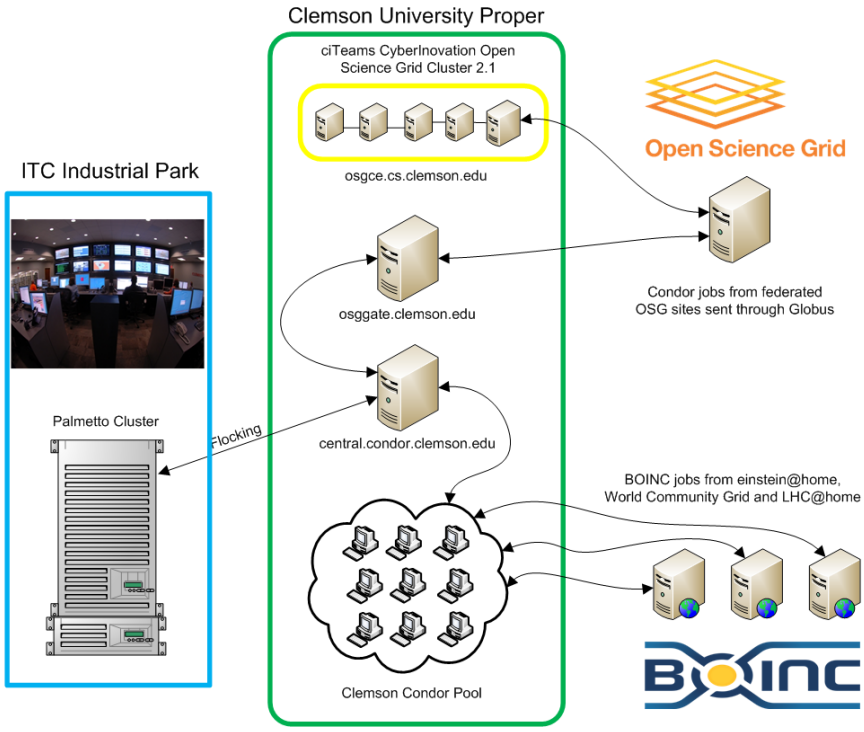


Figure 1.1: The Simplified Anatomy of the Campus Grid at Clemson University

Figure 1.1 shows the main components of the Clemson campus and how they interact with outside resources. Internally, Clemson has a central manager for the Condor pool (central.condor.clemson.edu) which receives Condor jobs from Globus through a local gatekeeper and compute element (osggate.clemson.edu) and local jobs from campus submit machines within the pool. The Globus Gatekeeper (osggate.clemson.edu) and the ciTeam test cluster osgce.cs.clemson.edu send and receive jobs from the Open Science Grid. BOINC is used to backfill Condor via a special Backfill command that is relatively new to Condor. Lastly the central manager and the Palmetto cluster, which is located at the Industrial Park, are set up to flock Condor jobs between the two resources depending on the operating system requirements of jobs. All of these components are explained in detail in the following sections of this chapter.

1.1 Research Question

The rapidly expanding field of cyberinfrastructure offers many research topics in grid computing and mechanisms used in distributed computing. The following two research questions can

be qualitatively and quantitatively compared to other grid solutions and have a direct impact on a diverse array of disciplines at Clemson University and other federated grid sites. The goal of this research is to find an effective vehicle for computation that enables Clemson to take the resources available through commodity hardware and accomplish more science. It is vital that a comprehensive solution maintain a positive end user experience and minimize the impact on the environment, while maximizing data throughput and hardware utilization.

How should a campus grid be built and maintained that will maximize utilization of available resources while maintaining a positive end-user experience, a secure campus network, and involve little administrative overhead?

Clemson University has well over 2500 cores in Windows XP and Vista lab machines throughout campus. Completely harnessing this compute resource for scientific research is accomplished by using Condor, BOINC, and a Globus interface. The following sections explain the research design and methods of building a campus grid with Condor as a foundation. Section 1.2 explains why research with Condor as a vehicle for campus grids is important, section 1.3 explains the end goal of this research with Condor, section 1.4 gives a short overview of Condor, section 1.5 explains why and how Condor is used to create a functional campus grid at Clemson, and section 1.6 concludes the chapter.

1.2 Motivation for the Study

Campus grids have been used at universities in other countries [49, 54, 56, 30, 53, 15] and at universities in the United States [51, 10] for a variety of diverse projects in multiple research domains. The afore mentioned grids provide a platform for parallel applications, job scheduling research, grid education, and are a possible indicator of the future model for computing around the world. [37] Clemson University has a large student base with many dedicated researchers whose research could benefit from a large campus grid. The research associated with the creation of the Clemson campus grid and use of BOINC backfill is important because it not only provides a stable base at Clemson University for distributed computing and grid education, but also creates a guide for other institutions that want to make a campus grid to support their own research while achieving nearly 100% utilization of resources.

1.3 Broader Impact of the Study

The purpose of this research is to find an effective vehicle for computation that enables us to take complete advantage of the existing resources we have, combine them with the resources of others, and accomplish more science all while giving back to the scientific community abroad. The broader impact of this research allows universities with smaller budgets and less funding to use community compute resources to make a valuable contribution to scientific research worldwide. The impact goes beyond the field computer science allowing for expedited cross disciplinary research in fields such as High Energy Physics (HEP), Geology, and climate change.

1.4 Introduction to Condor

Condor is a high throughput batch system that provides job management and data handling for distributed compute resources. It was developed at the University of Wisconsin at Madison nearly thirty years ago with the goal of recovering idle cycles from commodity machines such as lab machines and workstations. Condor centers its computing philosophy around high throughput computing versus high performance computing, which means that Condor is designed to reliably manage many computations over a long period of time [33]. Condor has been widely adopted by diverse Virtual Organizations (VOs) in the grid community such as the Open Science Grid [48] and the Teragrid [47] and is capable of integrating with grid technologies such as Globus.

Condor's basic architecture is a client/server model where there is a logical one to many correlation between the server and the clients. Both the server and the clients must have a condor_master daemon which spawns and monitors all of the other daemons. Figure 1.3 shows the client and the server as well as the resource available to the client through the server. A simple Condor server has three main daemons, the first of which is the afore mentioned condor_master which spawns and monitors the other two daemons. Condor uses plain text scripts called ads to communicate information between most Condor daemons. The condor_collector takes the machine ads generated by the worker nodes and maintains a dynamic list of available machines along with their specific hardware and software profiles. The condor_negotiator references this list when the client asks to procure a machine on wich to stage a job and returns the address of the available machine to the client. When the client receives the address of a compatible worker node it schedules the job directly on the worker node with no further direction from the server. Figure 1.3 shows that

there is a fourth daemon called the condor_procd which is responsible for writing the logs for all of the daemons, but this daemon is not integral to the running of the server.

The simple client, which is only capable of submitting a job and receiving the results of that job, has two main daemons. The expected condor_master daemon is present to spawn the condor_schedd or scheduler daemon which requests a machine with particular software and hardware capability from the server based on a condor submit file. A sample submit file is shown and explained in figure 1.2 In addition to submitting the job to the worker node the condor_schedd daemon also maintains a dynamic list of idle, running, and completed jobs. The client must remain online until the job is finished running because the worker node returns the results of a submitted job directly to the submitting node. This requirement makes persistent desktop computers good candidates for a submit machine, while transient laptops make poor choices for submit machines, because they may leave the pool unexpectedly before a job has completed.

The condor pool consists of machines running a condor_master along with a condor_startd which is responsible for negating in coming jobs, checking the status of the worker node, and marking the node as claimed by a job. The condor startd also reports back to the condor_collector with the status of the node, so that the collector can make informed decisions about matching a potential job to a worker node. The startd can be configured to run jobs on nodes based on a particular criteria, but generally the condor jobs are allowed to run when the worker node is idle. The worker node is normally (but not always) running Condor as a backfill system to a clustered system or a lab workstation.

It is worth noting that the actual Condor setup at Clemson has each Windows worker node running a schedd, so that each worker node can also submit jobs to the pool. Figure 1.2 shows a sample submit file. It has been discussed at Clemson that a system that has a restrictive API or limiting the submit machines to a select monitored few might allow finer control over what programs are submitted to the Condor pool. This configuration would reduce the number of badly written programs that enter the distributed system, preventing some of the security problems that have been encountered. While this view has value and is founded in common sense, it is contradictory to supporting an open environment where any researcher can easily use the Clemson campus grid to support their research without the intervention of programming specialists or direct assistance from an Administrator. Therefore because it is not practical for the staff at Clemson University to personally handle every submission to the pool, every worker node on Clemson's campus is also a

```
#Example Condor Submit File
#The type of job, can be java, mpi, etc
universe = vanilla

#Requirements tell the server what
#  qualities the job needs to execute
#  This includes architecture and
#  operating system, but can
#  include anything from memory and
#  specific software to custom
#  machine ads.
requirements = Arch == ``INTEL`` && OpSys == ``LINUX``

#The executable to be run
#  In this case the simple hostname
#  executable form bin
executable = /bin/hostname

#The location of the log, output, and
#  error output files for the job
log = example.log
output = example.out
error = example.err

#Assuming that there is no shared file system
#  these lines must be added to invoke
#  Condors native file transfer mechanism
should_transfer_files = YES
when_to_transfer_output = ON_EXIT


#The final command in any condor
#  submit file is the queue
#  which can be changed to schedule
#  different numbers if jobs,  here
#  5 jobs are scheduled
queue 5
```

Figure 1.2: Simple Submit File that is Submitted by the Client.

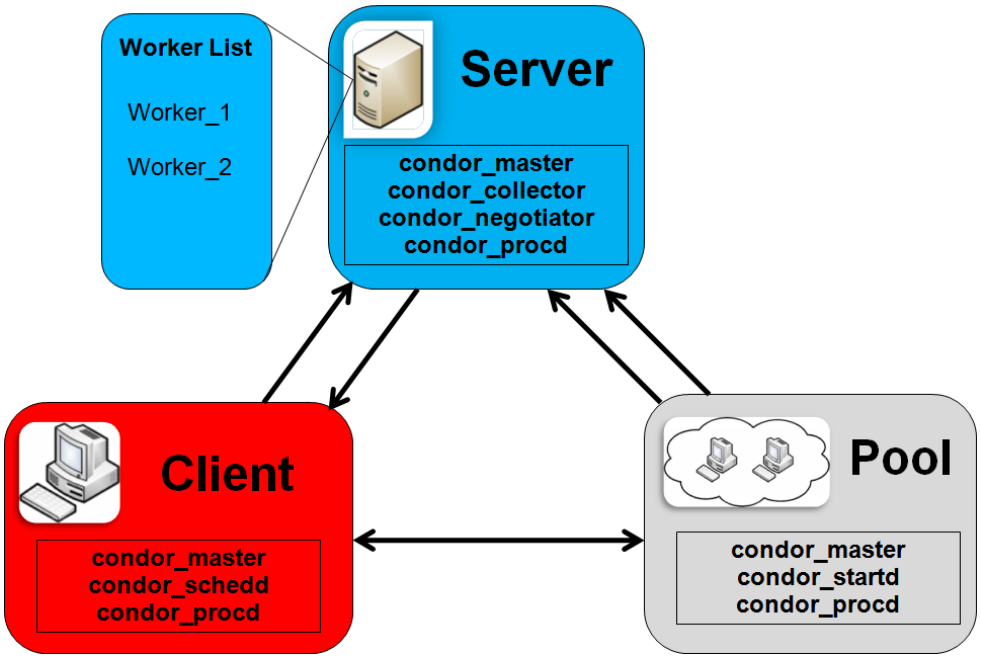


Figure 1.3: Simplified Architecture of a Condor Pool

submit machine.

Clemson University is using the Condor job_router daemon and the Backfill function in Condor to meet its computational needs. The job_router allows for Condor jobs to be sent out to other grid resources via Condor-G that support a common VO, such as Engagement. The participating pools can be found by querying the Resource Selection Service (ReSS) for the Open Science Grid (OSG) [23]. Condor's Backfill function allows other distributed batch systems to run under Condor's management; currently, the only supported Backfill system is BOINC. The following sections explain the basics of Condor, BOINC and the Open Science Grid as they relate to a Campus Grid. Section 1.5.1 explains the motivation for creating the Clemson Condor Pool, section 1.5.2 introduces BOINC, section 1.5.3 reviews how Condor and BOINC backfill are used to backfill the backfill system to achieve high resource usage, section 1.5.5 explains how the Open Science Grid is integrated into the campus grid to alleviate some computational demand during peek usage, and the final section 1.6 concludes the introduction.

	Total	Owner	Claimed	Unclaimed	Backfill
INTEL/LINUX	4	0	0	4	0
INTEL/WINNT51	946	280	3	375	288
INTEL/WINNT60	1357	193	0	4	1160
SUN4u/SOLARIS5.10	17	0	0	17	0
X86_64/LINUX	26	3	1	22	0
Total	2350	476	4	422	1448

Figure 1.4: The output of condor_status from a worker node on the Clemson Condor pool at noon on a class day.

1.5 Condor As a Way to Build a Campus Grid

Condor provides a comprehensive system for creating a campus grid, by managing distributed compute resources and providing remote Input/Output handling for staged jobs, as well as integrating with BOINC and OSG. While Condor can be used as a dedicated scheduler for a computing resource, it can only be a viable solution for maximizing utilization of commodity machines if there is some non-trivial amount of idle time. This key concern is discussed in the next section.

1.5.1 Motivation for the Clemson Condor Pool

Condor is only applicable if the amount of idle time on the machines is sufficiently great (for our purposes excluding Condor as a dedicated scheduler), so an informal survey was taken in early 2007. All the labs that were visited were Windows machines, so it was reasonable to assume that if no one was sitting at the computer, then it was idle. We found from this study that the labs were unused after midnight and over the weekends, indicating that the public lab machines were idle nearly 45% of the time. In addition to the informal study of public labs, we reviewed the schedule for teaching labs and found that those machines were idle more than 90% of the time (on average) justifying a system that would make use of these unused cycles. The amount of backfill shown in Figure 1.4 below supports our original hypothesis concerning unused cycles showing that of the 2350 machines 61% of them are in backfill and only 16% are idle, but without backfill 76% of the machines would be idle. Figure 1.5 shows that 92% of the machines would be idle, which again supports our original findings.

	Total	Owner	Claimed	Unclaimed	Backfill
INTEL/WINNT51	451	16	0	427	8
INTEL/WINNT60	2075	177	0	2	1896
SUN4u/SOLARIS210	21	2	0	19	0
SUN4u/SOLARIS5.10	1	1	0	0	0
Total	2548	196	0	448	1904

Figure 1.5: The output of condor_status from a worker node on the Clemson Condor pool at 3:00pm on a Sunday.

1.5.2 Introduction to BOINC

The "Berkeley Open Infrastructure for Network Computing (BOINC) is a platform for projects...that use millions of volunteer computers as a parallel supercomputer." [13] Institutions and individuals who use BOINC volunteer their free CPU cycles by joining BOINC projects. The BOINC client software is easy to install and will run with minimum administrative effort. In Windows it is as simple as downloading the mass-installer and updating the Condor configuration file. Institutions abroad have successfully used BOINC to create an institutional cyberinfrastructure [24], but Clemson has three requirements: a system that has more fine grained control over how idle CPU cycles were donated, was already known by some of the faculty at Clemson, and use a less complex API. Creating new BOINC projects involved using an API [8] which we felt would be restrictive given the lack of formal BOINC training among the Clemson faculty.

1.5.3 Why use Condor Backfill

In early 2007, Condor version 6.7 [52] added a feature called Backfill to Condor that allowed it to schedule a secondary job scheduler which would utilize the idle time that Condor could not use. BOINC was the first and (at the time of this writing) only officially supported backfill program that Condor could schedule. BOINC by itself could not be remotely administered in fine detail via internal components, but Condor, which can control BOINC in fine detail, can act as BOINC's manager. Condor is remotely administered by using enterprise administration solutions from Microsoft in the case of CCIT and has been administered in the School of Computing by dynConfig, a remote administration tool written by the author. The tool uses public key encryption to secure

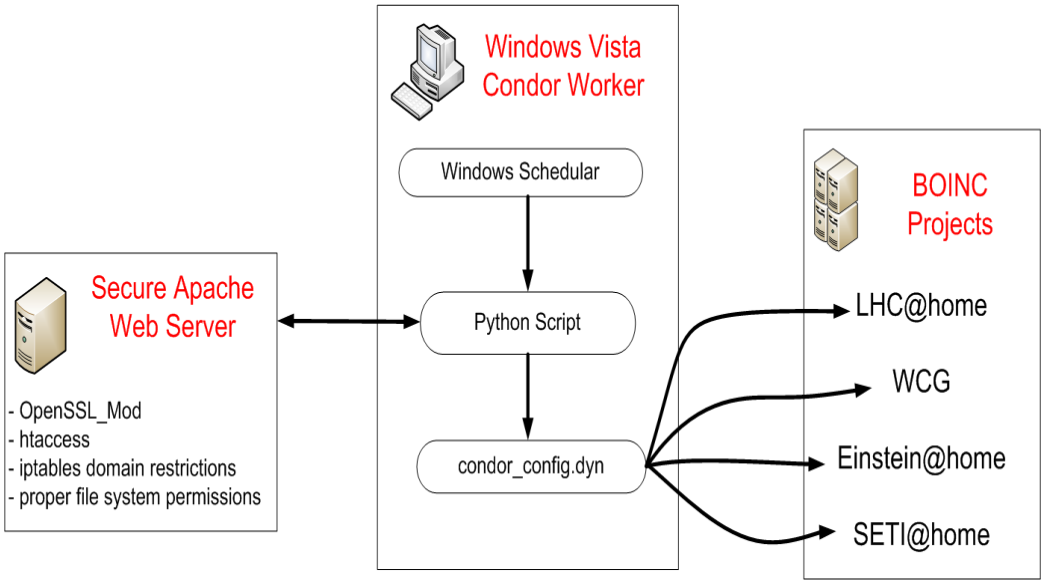


Figure 1.6: dynConfig Example

the connection between the client and the server to update Condor's configuration file at will, which is shown in Figure 1.6 in detail. The addition of the backfill functionality allowed Clemson to use Condor to manage both itself and BOINC with no negative interactions, essentially giving Clemson the ability to provide a usable resource for its faculty and students while donating CPU cycles to humanitarian causes.

1.5.4 Introduction to the Open Science Grid (OSG)

The Open Science Grid is a national grid composed of federated grid sites that work together to provide a production-scale computing resource. A grid is a collection of sites that work together to provide each of their users with the computing power, storage capabilities, and visualization resources that is equal to their combined capabilities through a common interface. The users of this grid form Virtual Organizations (VOs) so that they can easily share their work with people of like interests and be organized under a single administrative domain. The figure 1.9 shows what a medium grid site looks like. This configuration is the most common setup among the Engagement VO member sites. The explanation of this figure will be combined with the authors experience co-installing the CyberInovations OSG cluster (osgcs.cs.clemson.edu) which can be seen in figure 1.7.

A medium size OSG site is comprised of four main components each located on separate

machines: OSG Computing Element (CE), Grid User Management System (GUMS), OSG Storage Element (SE), and a resource which is commonly a cluster. The CE is usually the first machine to be installed and is the gateway through which all other sites and users communicate with the OSG site. All of the tools used to install the CE come from the Virtual Data Toolkit (VDT) which is downloaded via Packman package manager. At the time of installation the administrator can choose which type of OSG machine they are installing and have a clean and unconfigured site with no certificates. It is important to note that the approved list of architectures and operating systems should be examined before this installation because using the "masquerade" option, which enables a non-supported operating system to pretend to be a supported operating system does not work in the authors limited experience. It is also recommended that a site using Condor as a scheduler should pre-install and separately manage their own copy apart from the Condor that is installed through the VDT/Packman install. Many configuration files must be tailored to the individual site and each site and administrator must request a certificate from OSG with the consent of their VO representative. Other grid sites use the validity of the CE certificate to enforce grid security. Jobs submission is negotiated by the Grid Resource Allocation and Management system (GRAM), which is provided by the VDT along with CE Monitoring (CEmon) which allows independent testing of the finished OSG site.

Just before the CE install is finished a medium sized site will need to install a GUMS server on a separate machine from the CE. The GUMS server provides a dynamic list of federated grid users that can use the resources at a site based on VO membership and is exposed through a web services interface for convenient access. The GUMS server also replaces the static gridmap file, which provides the same information, but in a static text file with limited interactive capabilities.

The SE, which is not provided by Clemson's OSG install, is used at many sites that require extra storage space to stage large jobs, but is not required at a site as long as the CE and Cluster resource provide several expected directories through the shared file system. The shared file system, which in the CyberInovations OSG cluster is provided by Network File System (NFS), shares files between the CE, SE, and Cluster resource. Tools like GridFTP and Globus are also supplied by the VDT upon install and are used to move files to the site and authenticate and authorize users.

Figure 1.7: CyberInovations Test Cluster

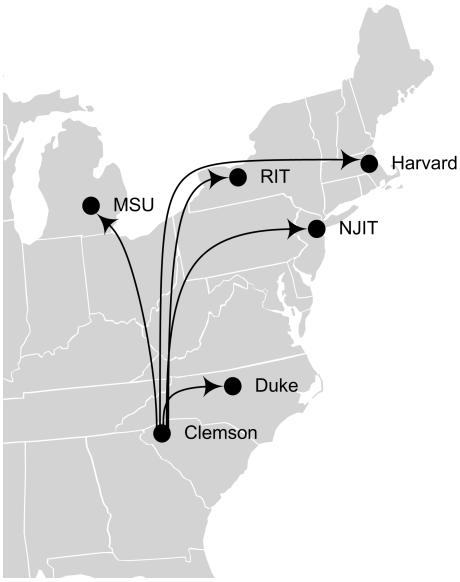


Figure 1.8: The sites directly affected by ciTeam on the east coast.

1.5.5 Open Science Grid Expands Campus Grids

For the grid model of HTC to emerge as the dominant model for the ever growing demand of computational power, there must be a concerted effort made to federate more grid sites. The Embedded Immersive Engagement for Cyberinfrastructure (EIE-4CI) project is part of an NSF grant awarded to the Renaissance Computing Institute (RENCI) and Clemson University. EIE-4CI's goal is to engage users and site administrators and train future CI professionals on how to install and maintain grid middleware such as Condor and the OSG Virtual Data Toolkit (VDT). Theend results is to create a grid site with new users that will mature as a grid site and move on to a VO that is tailored to their needs. Through this program, a five new grid sites have been added to the Open Science Grid in the Engagement VO, which has not only increased the total number of machines available for HTC with Condor at Clemson, but has also allowed Clemson to share its Condor pool with others. Figure 1.8 shows the sites that have been directly effected by the EIE-4CI program through ciTeam.

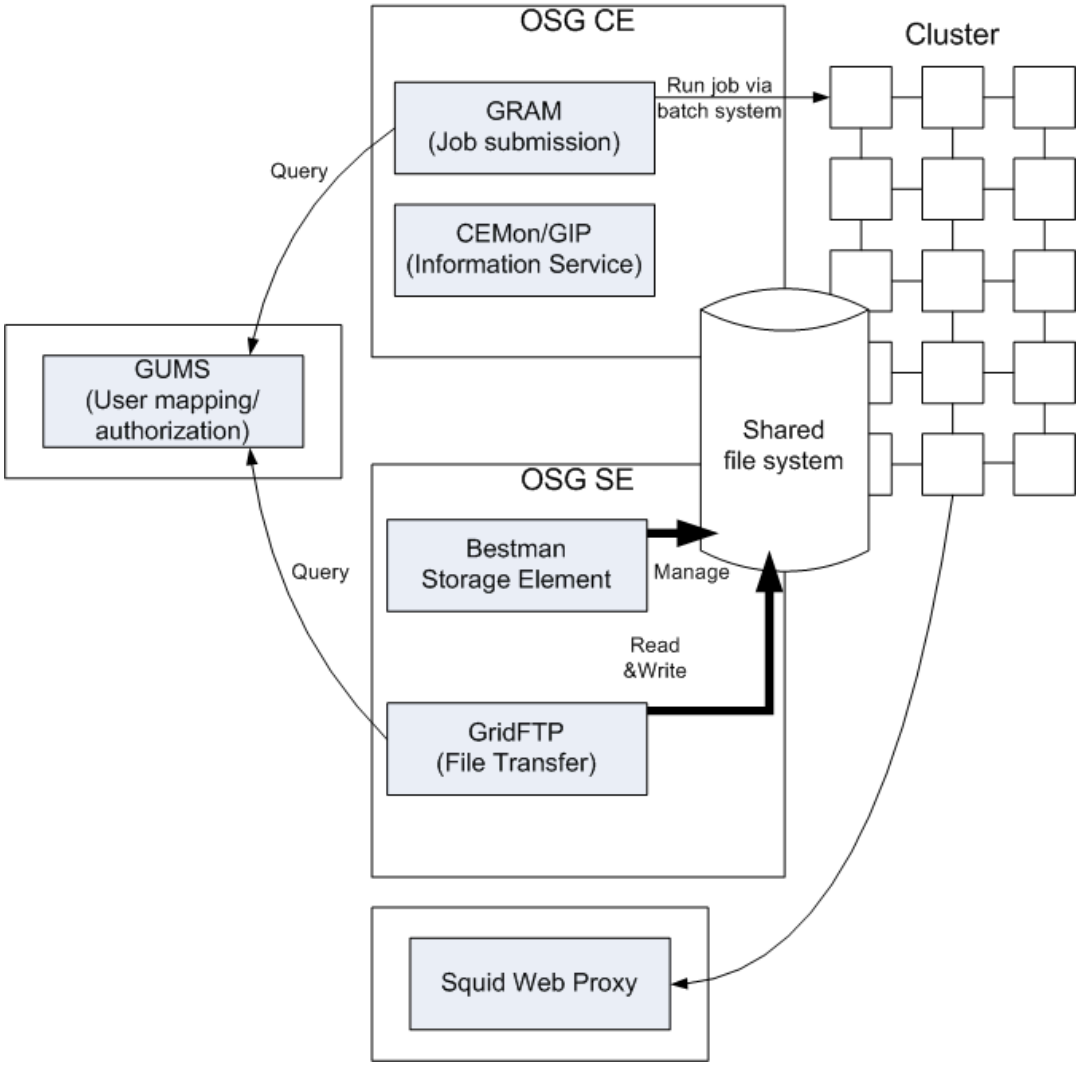


Figure 1.9: The Architecture of a Medium Open Science Grid Site. Diagram the OSG Twiki https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/SitePlanning

1.6 Summary of Methods

By harnessing the computing power of the commodity machines at Clemson University into a usable resource, providing a control device for BOINC to donate excess resources, and allowing grid enabled jobs to flow to and from the Clemson Condor pool, a comprehensive campus grid has been designed to feature almost 100% utilization. Local users can submit locally and run globally. Local cycles can be shared on an oppertunistic basis with collaborators. The three tiered approach using Condor, BOINC and the Open Science Grid will be expanded in the remaining sections of this Thesis. The following chapter will review the history of cyberinfrastructure and follow several modern Campus Grid projects which influenced the creation of the Campus Grid at Clemson University.

Chapter 2

Related Works: Grid Infrastructure

In the last decade there has been an explosion of new grids, both in the United States and abroad. A grid is a collection of independently administered sites that shares information, resources, and storage through a common interface with users who are partitioned into Virtual Organizations (VOs). These new grids are, in many cases, being formed from preexisting resources such as Beowulf clusters, dedicated rack mount server clusters, commodity workstations, distributed file systems, and NAS devices to name a few. The motley composition of these grids inspires researchers to write new grid middleware that takes advantage of the specific resources available at their site, while minimizing the common problems of too little disk space, user authentication and authorization, and throughput. In certain cases the software written for use in specific research grids, both national and academic, has become generic enough to be used in other grids. This software forms the foundation of many pervasive middleware software collections.

Academic grids or "Campus Grids" started to appear in the United States at the end of the 1990's and have grown into large scale, loosely coupled, cohesive organizations that share their resources with other such institutions and Virtual Organizations through the national grid infrastructure [18]. These campus grids will be discussed at length in this section, because they have set the groundwork for future Campus Grids by providing open source tools, first hand experiences, best practices, and benchmark results. Before we look at the modern grids, it is prudent to explore

the history of cyberinfrastructure that placed the groundwork for todays campus grids, therefore Section 2.1 explores the origins of the "grid" and an overview of cyberinfrastructure. The next four sections will focus on Campus Grids around the world, review what steps have been taken to create those grids, provide a basic overview of the problems that were solved, and explore common problems and practices between them. Section 2.2 will explore a two layer grid that was created in Warsaw Poland at the Warsaw University of Technology, section 2.3 explains how a functional Campus Grid was created at the University of Extremadura in Spain, and section 2.4 will conclude the academic grids with the "Academic Cyberinfrastructure" at Purdue University. Section 2.5 reviews the national grids of the United States and explains how they are used to connect the otherwise isolated Campus Grids.

2.1 Brief Grid Computing History

The concept of a computer grid is not a new idea and can be traced as far back as 1960 to John McCarthy who said that "computation may someday be organized as a public utility" [21]. This concept has been recently formalized in the form of Cloud Computing, which is an offshoot of Cyberinfrastructure. A viable batch scheduler which forms the heart of many grids was created by Miron Livny in 1988 and published in the 1988 paper "Condor: A Hunter of Idle Workstations" [32]. In 1997 Earle Ady and Christopher G. Stach II started Distributed.net [31] which began the volunteer computing movement [6] followed by the Search for Extra-Terrestrial Intelligence (SETI@home) with BOINC out of the University of California at Berkley in 1998 [29] with roots going back even further to 1994 [42]. More recently Ian Foster and Karl Kesselman coined the word "grid" and formalized its meaning in their 1998 book "The Grid" [28]. In the early 2000's the term grid became synonymous with the word "cyberinfrastructure". Today the word "cyberinfrastructure" is being interchanged with the word "cloud computing" [22] and is being spear headed under this new name by academic and commercial entities alike. According to Foster the evolution of cyberinfrastructure can be compared to other forms of infrastructure such as railroads, telephone lines, and banking systems [28]. Considering this comparison to be accurate the advent and expansion of cyberinfrastructure is therefore complicated and affected by many varying factors over a long period of time, but has some visible milestones that help to define a history of grid computing. The next section will move on to modern grids and describe in detail a modern Campus Grid at the University of Warsaw in Poland.

2.2 Two-Level Grid at the Warsaw University of Technology

The two-level grid design at the Warsaw University of Technology is inspired by the need to maximize computing power while minimizing cost [30]. Their grid consists of commodity machines from student instructional labs. At any point during the day a class could be held, which would temporarily decrease the amount of computing power available to grid users. Condor was considered as a vehicle for a Campus Grid, but concerns over complexity added by heterogeneous operating systems and complicated task allocation due to resource availability dissuaded its use. A two-level solution consisting of a lower clustering layer and a upper grid layer was used to create a campus grid, that meet the computational needs of the school while minimizing cost. Section 2.2.1 will explain how the cluster layer works, the following section 2.2.2 will explain the grid layer, and the last section 2.2.3 will conclude the two-level approach.

2.2.1 The Cluster Layer

The compute fabric of the cluster layer (lower layer) is composed of student lab machines that must be attached to the cluster when needed, but unattached when a class is in session. The process of "attaching" and "unattaching" is currently a manual process that requires administrator, teacher, or student intervention. The software that is used to connect all of lab machines together is MOSIX [9, 5], which provides a load-leveling middleware between machines that aggregates all of the machines in a room into a single virtual machine that is available to the upper grid layer. Unfortunately, MOSIX has several restrictions regarding ease of use, which includes only working with processors in the x86 family, needing a dedicated machine in each room, and needing a homogeneous operating system throughout the cluster. The latter is accomplished by the aforementioned manual rebooting of the machines to a CD or floppy disk. This approach is simple, but seeming inconvenient for the students and teachers using the lab. Inconvenient rebooting aside, this approach does create a heterogeneous computing environment which simplifies the end users interaction with the cluster and is an added feature that Condor does not have.

2.2.2 The Grid Layer

The grid layer of the campus grid is created using the Globus Toolkit [1, 19] and MPICH-G2 [40] a grid aware version of MPI. Globus provides uniform login via certificates to the multiple

student labs which have different security policies and account names, while MPICH-G2 provides communication between processes using Globus as a communication vector. Communicating with MPI through Globus is difficult, so to get the most from this type of cluster computations should be long with little communication. Globus provides many security features through proxy certificates, but a well maintained and monitored firewall is recommended. For testing purposes a small nine machine cluster was created using two separate four and five node clusters and the Google Page Rank Algorithm was run among them. The results generated by the cluster were an improvement over parallel code run on a single machine and over parallel code run with MPI alone. The favorable results support the overall design and usefulness of this campus grid.

2.2.3 Two Layer Overview

The Warsaw University Campus Grid is suitable for long running highly computational jobs that require little or no communication between processes. The original goals to have an effective homogeneous computing environment, save money, and provide an easy to use computing resource were met. Unfortunately this approach does not scale much beyond ten compute nodes because of the way the dedicated machines in each lab manage job distribution and communication between nodes, so this would not be a viable solution for use in Clemson University labs. There is no inherent mechanism in this grid that allows for fault tolerance in the case of a node failure during computation, so the individual code must be made fault tolorent. By definition a grid should "coordinate resources that are not subject to centralized control" [20], but this campus grid is said to explicitly be under one administrative domain, which implies centralized control.

2.3 BOINC Grid At The University of Extremadura, Spain

A novel campus grid using the volunteer computing software BOINC has been created at the University of Extremadura in Spain. [7] The philosophy behind this grid is that desktop PCs are overpowered for the menial tasks of writing papers, reading email, and surfing the web, and are thus underemployed. By using a middleware to aggregate all of these "underemployed" computing resources the computing potential of their workstations and lab machines may be realized. BOINC was selected over other middlewares such as Condor and Xtremweb [16] because it was "widely used and [has] great community support".[24] BOINC (the Berkley Open Infrastructure for Network

Computing) is designed as a server/client, where the server hosts the specially formatted BOINC project and the clients are (generally) the desktop PCs that have voluntarily joined a project. The model used by the Extremadura Grid removes the users from the project selection process and proposes a centralized control mechanism for managing the projects that institutionally owned computers join. Section 2.3.1 will explain the standard user based way of choosing projects along with using Account Managers, section 2.3.2 will explain how institutions may centrally manage a BOINC based Campus Grid, and section 2.3.3 will conclude the Extremadura Grid section with a proof of concept experiment and overall results.

2.3.1 User Based BOINC Model

The standard user model for BOINC depends on individual users with privately owned machines to volunteer their computer's idle time to projects. This model is completely dependent on good public relations through advertising interesting projects that hook potential users attention and compel them to donate their computer's time. This decision process is complicated because each project has a independent website and project URL, but has been simplified by several Account Manager projects that collect and summarize the available BOINC projects into documented lists. The two officially supported Account Managers are GridRepublic [2] and BOINCStats Account Manager (BAM). A fork of BOINC designed for academic/business users called SZTAKI Desktop Grid tries to take the global scope of BOINC and restrict it so that the BOINC system is more friendly for inter-institutional use. GridRepublic, BAM, and SZTAKI Desktop Grid all try to make the users experience better by managing projects and providing project information, but none of them try to manage Desktop Computers as a resource. [24]

Many times, users (employees) at an institution or company do not own their workstations or lab machines. In the case of institutionally owned machines, the user does not have a choice over which projects the machine they use will join, but for the institution to select a project each workstation must be updated by a administrator or a privileged user, which is inconvenient and unrealistic for large organizations. The following section explains the philosophy and tools that were used at the Extremadura Grid to realize the computing potential of institutionally owned workstations and lab machines.

2.3.2 Resource Based BOINC Model

The crux of this model consists of a two-part (server/client) tool that enables institutions to remotely manage groups of computers, assign projects to those groups, administer common BOINC tasks, and retrieve BOINC statistics. [24] Without this tool to facilitate central management BOINC alone would not be suitable for inter-institutional use. There are two time policies that the BOINC enabled workstations follow; the first policy is used when BOINC is going to run uninterrupted. A dedicated machine acts as an alarm clock for the other machines that have been programmed to Wake On LAN during nights and weekends so that BOINC can have uninterrupted runtime. Each machine is on a timer to shutdown after a set amount of time. The second time policy is used when students or employees are working on the machines, so that BOINC will not impact their user experience. The client software consists of two parts: a "mediator" part and a "BOINC core client". The BOINC core client is simply the BOINC executable without the GUI portion, but the mediator portion of the client software consists of a "launcher" to start and stop the BOINC core client and a "delegate" to submit statistics to the server and attach the BOINC core client to projects. The server side includes a two-part web interface for taking project connection requests and gathering information about client machines.

2.3.3 BOINC Model Overview

This design has many merits, including being able to successfully gather the computational power of the workstations and lab computers at an institution into a functional Campus Grid. BOINC can easily be extended outside of the university by virtue of the pervasive nature of volunteer computing systems. A proof of concept experiment which implements a genetic algorithm showed that the Campus Grid was capable of generating results many times faster than a single computer, thus validating the grid infrastructure as a worthwhile investment. BOINC is very easy to use from an end user point of view, but its main drawback is the complicated API that must be used to encapsulate project code. If a researcher who is unfamiliar with programming wanted to create a project for their research, a BOINC administrator or other third party would need to take what code they have and encapsulate it for them. The ease of use for the end user (who in this case does not exist) is easy, but this ease of use is paied for by the BOINC Administrator. BOINC does have other uses within a Campus Grid. It can be used as a backfill to a backfill system such as Condor,

so that as an end user with no projects, an institution can enjoy all of the benefits of a centrally controlled Campus Grid with the added benefit of donating free cycles to needy projects. This is further discussed in the Research Design and Methods Section of this thesis.

2.4 Campus Grid at Purdue University

Purdue has a large scale "Academic Cyberinfrastructure" program which uses Condor and Portable Batch System (PBS) as schedulers. [51] The grid at Purdue uses Condor to pick out the idle cycles that are left by PBS in their community clusters. [39] A community cluster is a cluster that has been bought by faculty and is maintained by professional IT staff, so that the faculty can concentrate on research and not be burdened with the day-to-day maintenance of a large cluster. A study at Purdue found that the community clusters were idle for nearly 30% of the time, but still used 50% of power that a fully utilized system uses. [51] The 30% idle time found by the study justified the use of a middleware to make use of this unused resource, so Condor was employed to handle the job, creating clusters that have greater than 95% utilization. The challenges and solutions as well as references to the Condor pool at Clemson University are made in section 2.4.1. Clemson is mentioned in this section because the implimentation at Purdue closely matches the Clemson implementation and comparing and contrasting the two systems reveals key differences in functionallity by design. Section 2.4.9 concludes this section.

2.4.1 Challenges with Condor at Purdue University

While maintaining the Campus Grid at Purdue many novel challenges have been identified and addressed. The lessons learned from these challenges are broad enough to be applicable to campus grids in general and should be considered by institutions who would like to deploy a campus grid of moderate to large size. The following challenges are reviewed briefly to give a flavor of what must be considered when running a large Campus Grid: Upgrades, Checkpoint Servers Scaling, multiple Central Managers and resulting pools, Submissions Hosts, Usage Tracking, Storage, Grid Education, Networking, and Staffing.

2.4.2 Upgrading Condor

The first of many challenges is to continuously keep Condor upgraded to the latest version because the Condor team is constantly making useful changes that increase throughput and reduce security risks. For example, a new version of Condor in 2007 contained an update to the schedd that increased the throughput by a factor of 25. [51] At Clemson University, upgrading and a monolithic management system was originally not a top priority, so there are currently eleven different version of Condor from 6.6.1 to 7.2.1. The variety of versions may be the cause of various problems including problems with the collector and the negotiator and has since become a priority. While the versions of Condor are starting to converge, it would have been a much easier process to upgrade and manage images if this had been considered from the beginning.

2.4.3 Checkpoint Servers

Checkpoint Servers are used in conjunction with Linux hosts to save the progress of jobs in the case of a significant failure or a job that is preempted. This process allows the job to start from the checkpoint instead of having to start from the beginning. While this saves time and reduces redundant computation, it is costly in memory. Purdue noted that "[a] simultaneous checkpoint operation across a few hundred standard universe jobs overwhelm[s] [a] single server" that has 200GB of available space. [51] To solve this problem, a small cluster of checkpoint servers was deployed with seven nodes with a total of 3.5TB of available space. The provided space has proven to be more than adequate to handle all of the checkpointing needs of Purdue University. Clemson University has a single checkpoint server that services the Palmetto Cluster, but is not available for the over 2500 Windows slots in the pool.

2.4.4 Multiple Pools

Originally a single central manager served the 1800 nodes at Purdue, but due to performance limitations the pool was broken up into logical partitions containing approximately 1000 nodes each. The pools were further set up to flock between each other, creating a pool that was large in depth but small in breadth. This produced several small but stable pools that can share the job workload between them, rather than having a monolithic unstable pool. Clemson University is now experiencing some of these same performance problems by using one collector for 2700 slots

distributed across the campus, but curiously there have been no scaling or performance problems with the 6000 node Palmetto Cluster which uses only one Collector and Negotiator and is set to flock Windows jobs to the campus pool and accept flocked Linux jobs from the campus pool. The consensus among the administrative staff at Clemson University is that the multiple architectures and operating systems, combined with inconsistent networking and Condor versions contributes to this phenomenon. It was also found that Condor version 7.2.* has a bug that causes the schedd to crash when there are more than 10,000 jobs submitted to the pool. The Condor team has been made aware of this problem and is working to fix it.

2.4.5 Logging and Tracking

For any large system logging and usage tracking is vitally important to maintaining a stable environment for users and gathering data for future research. Condor places the logs for each job run on the machine that submitted the job. This behavior is simple to implement but not optimal for gathering data and placing it into a central repository. Purdue has limited the number of submit hosts to a select a few machines which run Perl scripts that relay the job logs back to a POSTgreSQL database. The information recovered from these machines has uncovered some invalid wall time entries for jobs which have been reported to the Condor team. At Clemson University nearly every machine is a submit machine so logging job information from each of the machines would be an impossible task. When Condor was first set up at Clemson the administrators tried to promote its use by making it accessible. So far there have been no significant problems with this approach.

2.4.6 Storage

Transparent end user storage is import for any grid system to function properly. The space provided by the grid site allows the user to store and stage input files and executables before they start a job and collect their output after a job has finished running. Purdue found that space was not as much of a problem as how that space was presented to the user. The use of NFS to write out all of the users logs and output files would not scale to large numbers of nodes and would "reliably crash the NFS server"[51]. To tackle this problem they have trained the users to make use of Condor's file transfer device and have phased out the NFS system in favor of NAS devices. Clemson has not

experienced these problems with the Windows pool, mostly because there is no shared file system between all of the Windows machines in the public labs or on the employee workstations. The Palmetto cluster has not seen these problems because it uses NAS devices with the SAM/QFS file system.

2.4.7 Grid Education

Good grid education has become an important part of grid use and promotion, so Purdue has made an effort at the TeraGrid'07 conference to educate their own staff and staff from other institutions on common Condor user problems. Clemson has had many problems related to uninformed users and administrators, but has recently started a new push to further education about the Condor pool. The creation of a condor_admin mailing list and a new cancer research project has renewed interest in Condor among potential users and now the Condor pool is receiving more attention from top tier administration. This has created the opportunity for group meetings between researchers and administrators, so that small problems are solved before they become larger problems.

2.4.8 Staffing

Staffing is an important consideration for any large grid, and can be likened to provisioning cooling for the same system with under-staffing and under-cooling having a similar disastrous outcome. Purdue has come to the conclusion that a "single system administrator can support the upkeep and management of the enterprise-scale Condor infrastructure with approximately 25% of his effort." [51] They have found that as the number of users grows more support staff are needed to help the administrator with end user support. At Clemson University, we had a full time Condor Administrator who was supported by graduate students and CCIT personel.

2.4.9 Purdue Grid Overview

Purdue recognizes the potential for scientific computation, with minimal investment in staff and hardware for the Condor-based campus grid and has noted several best practices. They have found that it is important to follow the development of the campus grid closely and mitigate problems as they become apparent before they affect computers across the campus. A pool must be large

enough to be effective, so partitioning the available machines into pools that have about 1000 cores is a good practice. If people are going to use Condor they must be able to submit jobs to the pool, so therefore they must have access to schedds on submit machines. A shared file system is important to support jobs that have large disk needs, but for sufficiently large disk bound jobs an enterprise-scale storage solution may be necessary to facilitate computation. The overall effectiveness of Condor-based enterprise-wide grid-aware Campus Grids can not be denied, and will continue to be an effective vehicle for computation for the foreseeable future.

2.5 National Grids in the United States of America

National grid infrastructure connect smaller grid sites to each other via a common interface and serve as a trusted third party for federating trust among VOs. National grid sites use the same middleware that the smaller sites use to aggregate resources, but provide web portals and certificate based authentication to access all of the smaller grid sites that they represent. There has also been attempts to quantify the power of these grids [26], but none have resulted in a standard. The following sections explain the practices of a few of the larger grids in the United States and how they play into the larger role of cyberinfrastructure in the United States. Section 2.5.1 explores the New York State Grid which serves the distributed computing needs of the New York State area as well as national and international interests, section 2.5.2 explains the practices of the Teragrid, section 2.5.3 explains how the Open Science Grid operates, section 2.6 presents an overview for this section.

2.5.1 NYS Grid

New York State Grid [38], which is headed up by Millers's Cyberinfrastructure Laboratory (MCIL) [3], is an example of a regional grid that has ties to the national grid infrastructure via common interfaces such as Globus and GRAM. The Campus Grid encompasses a campus and perhaps outlying satellite campus (commonly referred to as a CAG or Campus Area Grid), whereas a regional grid may encompass several campus grids and commercial grids. A national grid encompasses many regional grids and Campus Grid. Finally, an international grid does not encompass national grids but links them together via national networking infrastructure. NYS Grid currently uses the Globus Toolkit to provide APIs and SDKs which simplify development of grid services, as well as

using GRAM and GridFTP to support the research done through their active web portal. The web portal is a major component of the infrastructure for the NYS Grid and facilitates a simplified user interface to access resources by creating a single sign in. A grid monitoring tool has been written in house to keep track of important statistics and monitor grid health.

2.5.2 Teragrid

Teragrid is an NSF funded project that originally linked four sites through a 40Gigabit fiber connection. Currently, the Teragrid "includes 25 platforms at 11 sites and provides access to more than a petaflop of computing power and petabytes of storage." [55] "TeraGrid is coordinated through the Grid Infrastructure Group (GIG) at the University of Chicago, working in partnership with the Resource Provider sites: Indiana University, the Louisiana Optical Network Initiative, the National Center for Atmospheric Research, National Center for Supercomputing Applications, the National Institute for Computational Sciences, Oak Ridge National Laboratory, Pittsburgh Supercomputing Center, Purdue University, San Diego Supercomputer Center, Texas Advanced Computing Center, and University of Chicago/Argonne National Laboratory, and the National Center for Atmospheric Research." [14]

Every site needs job negotiation and servicing capabilities and the Teragrid Science Gateways takes the place of the GRAM and Globus in the smaller grids and is used to manage users and resources. The idea behind Science Gateways is that each site would run a Science Gateway which is slightly different, but uses the same ubiquitous standards. Unlike some smaller grid projects cyberinfrastructure education is often built into the Science Gateway providing a convenient single point of entry and education. The web portal or client software that manages this single point of entry also manages a user account for each researcher that includes how much research they have done, what machines they have access to, and how much research they can do with their current allotment of CPU hours. An account on the Teragrid must be approved by an administrator and sent by US mail before a user is allowed to use the resources. [14]

Auditing the grid has been shown to be a top priority for many smaller grid sites so it is a logical assumption that macro grids require macro auditing tools, which is why the Teragrid developed GRAM Audit. GRAM audit is "an extension that lets gateway developers retrieve the amount of CPU hours a grid job consumes after it has completed." [55] In 2007 after an extended test, it was discovered that the load balancing systems were not able to perform as desired under

heavy load, so it was necessary to collaborate with Teragrid contributing sites to develop more fault tolerant applications across the grid. In addition to fault tolerance it was important to maintain scalability so that the wide mission of broadening Teragrid's user base and open goal of achieving compatibility and interoperability with peer grids was maintained.

Teragrid is used for a variety of cross platform applications such as Earthquake simulation, electromagnetic studies, biological simulation, and distributed rendering environments to name a few. [25, 47, 55] This diverse and challenging assortment of projects has been facilitated by the Science Gateways ease of use, but has also become self-propelling in that more research begets more research and more sites are pulled in to fill the demand. [11] In 2006 the Teragrid partnered with the Open Science Grid to increase their total computation power. [45] The following section 2.5.3 explains more about the Open Science Grid.

2.5.3 Open Science Grid

The Open Science Grid [4] is a collection of smaller grid sites which has a partnership between other large grids such as the Teragrid. The Open Science Grid has facilitated the development of several novel services such as the Virtual Data Toolkit (VDT) which allows grid sites to quickly install a software package containing all of the components they need to begin. They have also created the concept of a Compute Element (CE) and a Storage Element (SE) which are key components in OSG supported grid sites and have software stacks packaged by the VDT. ReSS (Resources Selection Service) allows transparent selection of resources for user's projects [23] was also developed for OSG sites along with other pervasive middleware [28]

Grid security has been a top priority of the Open Science Grid so a comprehensive group of tools has been developed and standardized through the VDT for authenticating users and federating trust. [34] Clemson University is a member of the Open Science Grid and uses a Globus Gatekeeper to manage and monitor incoming jobs to Clemson and outgoing jobs to OSG. Clemson is a member of the Open Science Grid Engagement VO which is supported by the NSF EIE-4CI project and is composed of grid professionals that offer help to startup OSG sites and provide remote assistance to OSG users. The topographical map in figure 2.1 below shows Clemson as a site on the Open Science Grid.

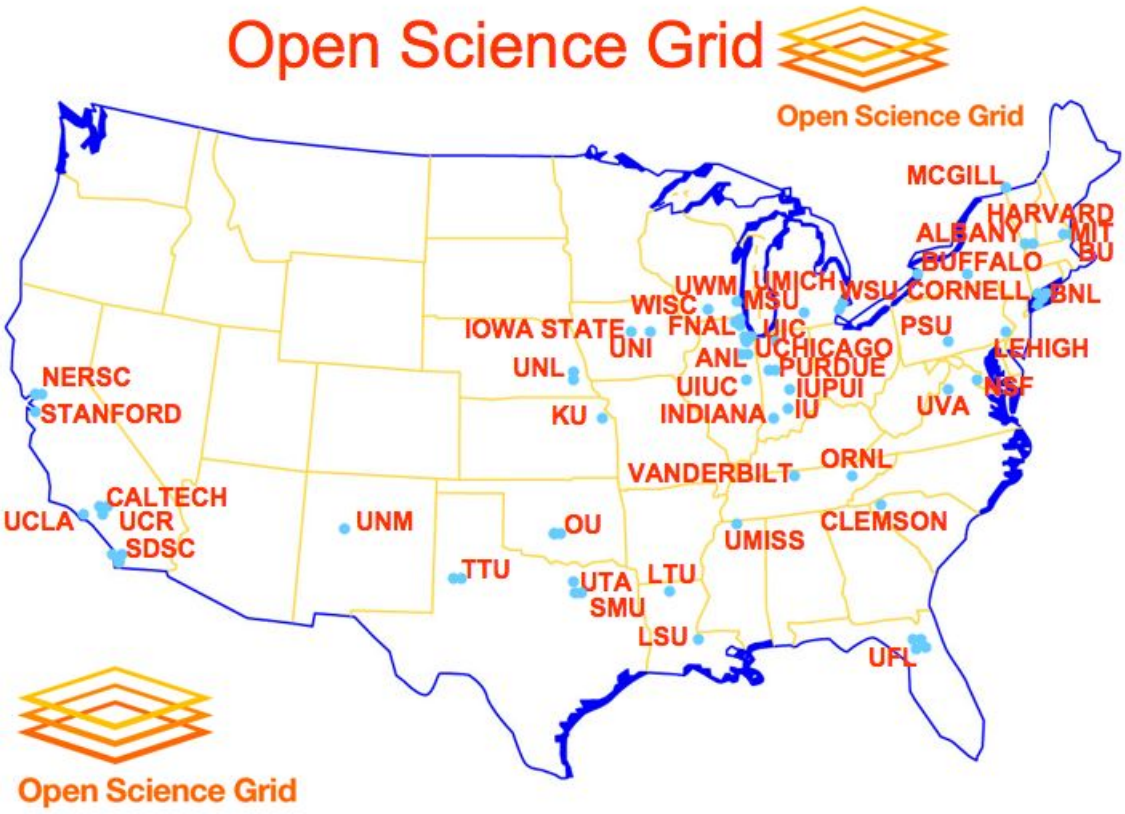


Figure 2.1: A topographical map showing the Open Science Grid sites. Clemson is the only site in SC. Picture Credit: http://www.uslhc.us/images/osgmap_aug08_720.jpg

2.6 Common Experiences in Grid Computing

There are many common problems and solutions among grid projects, as well as common experience among grid users. The first and most common problem experienced by grids is the need to monitor what goes on across the grid in realtime so that problems can be fixed before they cause widespread outages or loss of data and computation time. All of the grids mentioned in the previous sections have either used an open source auditing system from one of the national grids or they have created an in house solution for the problem. Using Condor as an example, logs are kept on the machines that submitted the jobs and that have run the jobs, but there is no centralized place that collects these logs for Condor. In a grid that has a topography that is not contained to one building, collecting these logs becomes a task and real time monitoring of job logs is not natively possible to the fine detail that is desired. Consider a grid that encompasses hundreds of sites distributed

over thousands of miles and the realtime monitoring problems become apparent. Many solutions to this problem have been proposed and several have become widespread. The Ganglia cluster monitoring software has been used at Clemson to monitor virtual and physical clusters with success and the Gratia system is used by the OSG site that serves the cyberinnovations lab. At this time a universal monitoring service does not exist but with some in house creativity and preexisting tools a satisfactory auditing system can be deployed for most organizations pursuing grid technology for use in a Campus Grid.

The second most common problem is grid security and federation of trust. This problem has applications in many domains, but is particularly difficult to solve with grids. Open Science Grid is an example of a large scale grid project with users from different institutions and countries, each with different ways of authenticating their own grid systems. First OSG must have something that a group of researchers would like to use, and next the new users must trust the Open Science Grid and accept them as a certificate authority (CA). The next step in federating trust requires that OSG trust the new users. This can be an involved process requiring several emails, a trusted third party, and possibly a physical visit from a OSG representative. Once the two groups agree that they can trust each other they must agree on some usage policies for each to follow. After all of the formalities of establishing and managing trust are achieved a mechanism must be used to allow the new trusted user to access OSGs resources and be authorized. It is at this point that many grids diverge and use a variety of mechanisms from web portals from certificates to individual user names and passwords for each machine at a site. Each mechanism has its pros and cons, but in general trade-offs will be made between the level of security and ease of use, so it is important for a mechanism to be chosen with qualities that fit the individual grid.

The third and final problem is the need for qualified staff to administer the grid and support users. All of the afore mentioned grids have discussed this need within their organization. Without the proper staff to manage a Campus Grid the system and the users suffer neglect. Is is vitally important that an administrator have proper maintenance and user support at the top of their everyday list. It is only through the diligence of an administrator that updates, problem prevention, and user satisfaction goals can be met. If a system does not operate correctly users will go to other sites to accomplish their computational goals, so it is not only important to the local grid that an administrator be assigned to grid management, but also vital to the overall expansion of the national and international grids.

Chapter 3

Scientific Application Running on the Grid

Scientific applications have parallel operations and, in the case of this particular experiment, have very little if any communication between nodes. The ciTeam from the Cyberinovations lab was approached by Logan Johnson and Prof. Mount from the Biology department to "Condor-ize" a deconvolution algorithm used to focus images from an experimental confocal laser-based microscope from Nikon, which can be seen in Figure 3.1. Our team has been a leading contributor to campus Condor maintenance and grid user education, but for the first time we have taken on a real world problem using Condor. After our first meeting we determined that the campus Condor pool will not be sufficient for this project due to the job's large disk needs and the lack of a shared file system between the nodes. The samqf NAS based shared file system that provides the shared file system for the Palmetto cluster is a perfect match for the many small files that must be staged for each job. Palmetto also has a fiber-based 1gig Ethernet connection between each node which also provides the connections to the samqf NAS based shared file system.

Palmetto is an example of a "Condo Cluster" where nodes are bought by investors who then have partial ownership of the cluster and a related guarantee on computing power available for their jobs. A Condo Cluster is a collection of nodes that is administered by an dedicated group of administrators inside of an organization. Most of the nodes are supplied by scientists who have a priority based on the number of nodes they have supplied to the cluster. The scientists are then

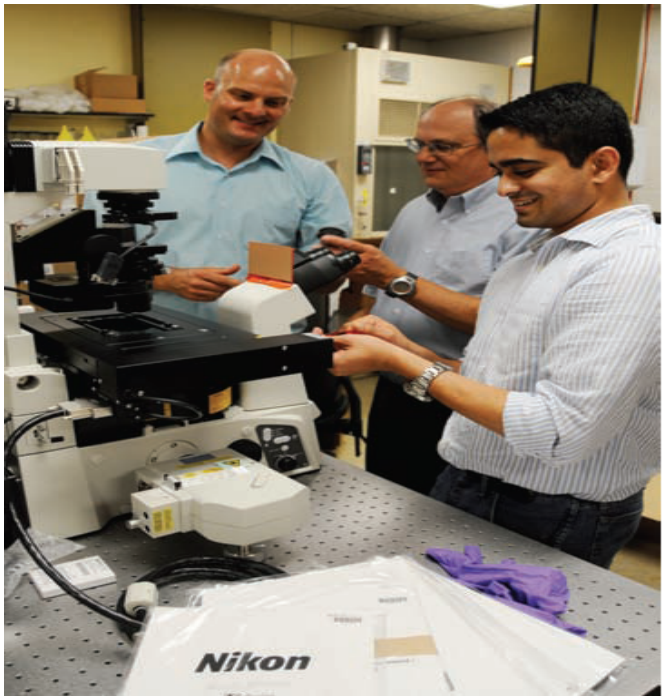
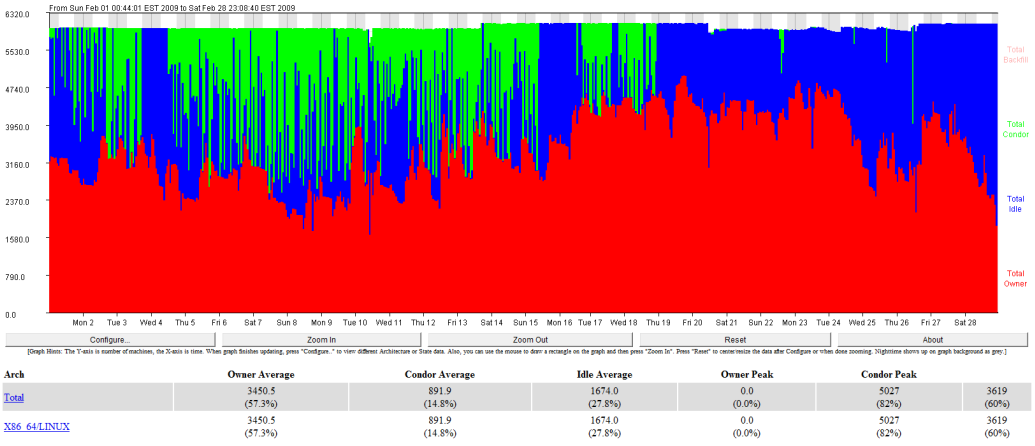


Figure 3.1: New Microscope from Nikon (Dr. Mount center), Photo provided by Nikon.

able to use a computing resource without the overhead of administrating the cluster by themselves, which gives them more time to do actual research. There have been examples of Condor clusters where only the scientists that contributed the node can use the node, but at Clemson a priority system is used and all contributers have the potential to use the entire cluster.

The main on-demand scheduling system for the Palmetto cluster is PBS, but Condor has been added to scavenge cycles taking the 57% utilization of the cluster to nearly 82%; this additional usage can be seen in Figure 3.2. Scavenged cycles from Condor do not cost anything to use, but are variable in availability and divided among the Condor users. This being said, there are usually between 500 and 1000 nodes avaliable for Condor out of the 6000 cores found in the Palmetto cluster. Section 3.1 explains how Condor and Matlab interact in general and how specifically they interact on the Palmetto cluster, Section 3.3 explains the setup and procedure for the experiment, Section 3.3.1 explains how sucessful the experiment was along with some statistics, Section 3.3.1 concludes, and Section 3.3.2 lists acknowledgements.



Arch	Owner Average	Condor Average	Idle Average	Owner Peak	Condor Peak	
Total	3450.5 (57.3%)	891.9 (14.8%)	1674.0 (27.8%)	0.0 (0.0%)	5027 (82%)	3619 (60%)
X86_64/LINUX	3450.5 (57.3%)	891.9 (14.8%)	1674.0 (27.8%)	0.0 (0.0%)	5027 (82%)	3619 (60%)

Figure 3.2: Usage On Palmetto

3.1 Introduction to Matlab and Condor

Matlab, which stands for Matrix Laboratory, is a high level language similar to Java that provides a platform for technical computing, and includes a graphing utility as well as a well defined and documented API for working with various data structures and algorithms. Matlab is widely used across many scientific and engineering domains because of its ease of use, but each license for installing Matlab is prohibitively expensive. For this reason Matlab is not installed on the nodes of the Palmetto cluster, rather Matlab libraries are installed in their place. This allows programs that are compiled by Matlab Compiler (MCC) to be run without having to open a time consuming Matlab shell or consuming a costly site license. There has been some research with Condor and Matlab [50], but we have found none that have combined Matlab and Condor to create a distributed deconvolution algorithm. This being found it was decided to delve into and setup the deconvolution experiment on the Palmetto cluster. The code from Figure 3.3 shows how Matlab is used to generate Condor submit files in an intermediate code section from the deconvolution project.

3.2 High Throughput Computing Vs High Performance Computing

There are two different schools of thought that are applied to grid computing. The first is High Performance Computing (HPC), which uses cutting edge hardware with massively parallel

```
    for i=1:layers
    string = strcat('layer', num2str(i), '.condor');
    fid = fopen(string, 'w');
    num = num2str(i);

    %add to dag
    write = ['Job ', 'Layer', num2str(i), ' ', string, '\n'];
    fprintf(dag, write);


    %write the submission file for each layer
    write = 'Universe = vanilla\n';
    fprintf(fid, write);
    write = 'Requirements = OpSys == "LINUX"\n';
    fprintf(fid, write);
    write = 'Executable = run.sh\n';
    fprintf(fid, write);
    write = ['Arguments = ', num, '\n'];
    fprintf(fid, write);
    write = ['Error = output/mlab', num, '.err\n'];
    fprintf(fid, write);
    write = ['Output = output/mlab', num, '.out\n'];
    fprintf(fid, write);
    write = 'Log = output/mlab.log\n';
    fprintf(fid, write);
    write = 'Getenv = true\n';
    fprintf(fid, write);
    write = 'should_transfer_files = YES\n';
    fprintf(fid, write);
    write = 'WhenToTransferOutput = ON_EXIT\n';
    fprintf(fid, write);
    write = 'transfer_input_files = img_dat.mat, psf_vol.mat, auto_weird.m, weird_mach
    fprintf(fid, write);
    write = 'Queue 1';
    fprintf(fid, write);

    fclose(fid);
end;
```

Figure 3.3: Sample Code from Logan Johnson that has been edited by Ben Sterrette in Matlab

algorithms to accomplish run times that dwarf linear execution of the same program by orders of magnitude. HPC closely follows theoretical computer science and is used when funding and resources are available. The second school of thought tends towards High Throughput Computing (HTC), which aggregates commodity hardware into a usable resource. HTC follows applied computer science and is used regardless of most funding problems, because it makes use of hardware and infrastructure already in existence.

3.3 Deconvolution Experiment: Setup and Procedure

There are three parts to each parallellizable part of the deconvolution problem. The first is the PSF (Point Spread Function) that is used on each image to account for the error introduced by the microscope in each sample. The second part is the image stack that is being deconvoluted. In these experiments it was the image being deconvoluted and the image directly above and below that image. The PSF is applied to the image stack and removes error from the middle image via a Fast Fourier Transform (FFT). The last part is the "wave" of images that is loaded into Condor and pushed out across the Palmetto cluster. In this case there are 250 images being sharpened, but each wave (which uses all 250 images) requires the previous set of 250 images to deconvolute the next 250 images. After several hundred iterations, the images are as sharp as the PSF and original quality of the images will permit. Image 3.4 and image 3.5 show a before and after image that has been deconvolved by the distributed algorithm.

This project started as two different ideas being implemented by separate programmers, with the idea that they would be compared in a future paper. The first of these ideas was that a single management submit file that used a Condor DAGman (Directed Acyclic Graph manager) would manage a layer of the distributed deconvolution algorithm across Palmetto. Each job would have its own submit file which was generated at the end of the computation, but be controlled by a DAGman job. This was juxtaposed by a plan that took each wave and made individual submit files as the program came across each image set without the use of a DAGman file and used scripts to wait for the first wave to finish. Eventually the two plans merged into a single program that most resembles the second non-DAGman plan. This is the program that has been submitted to the Palmetto cluster.

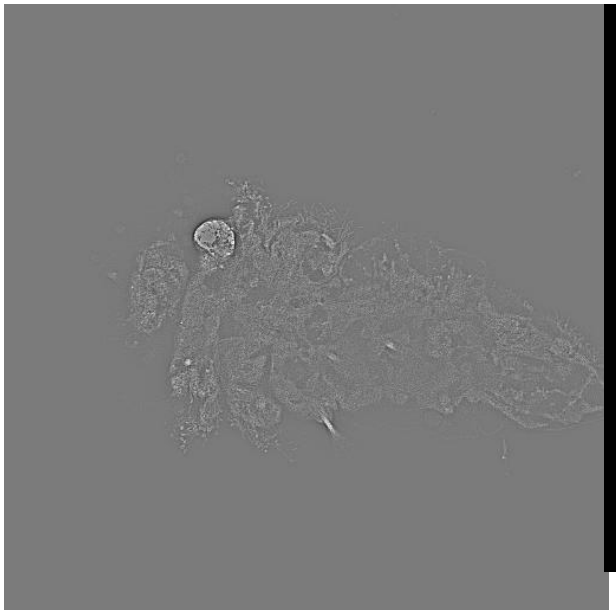


Figure 3.4: Image Before Deconvolution

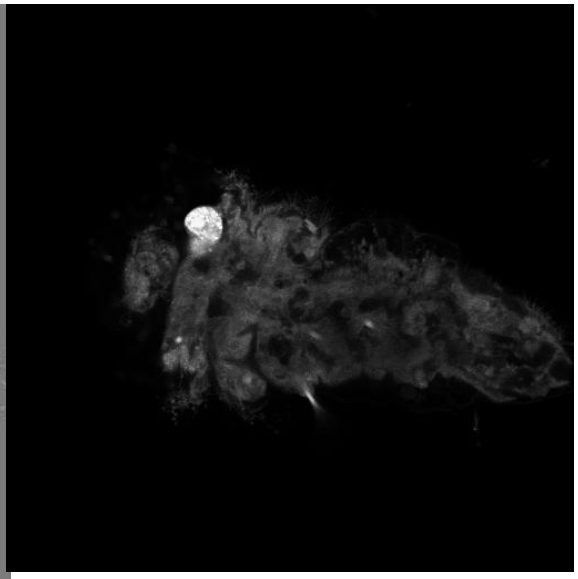


Figure 3.5: Usage After Deconvolution

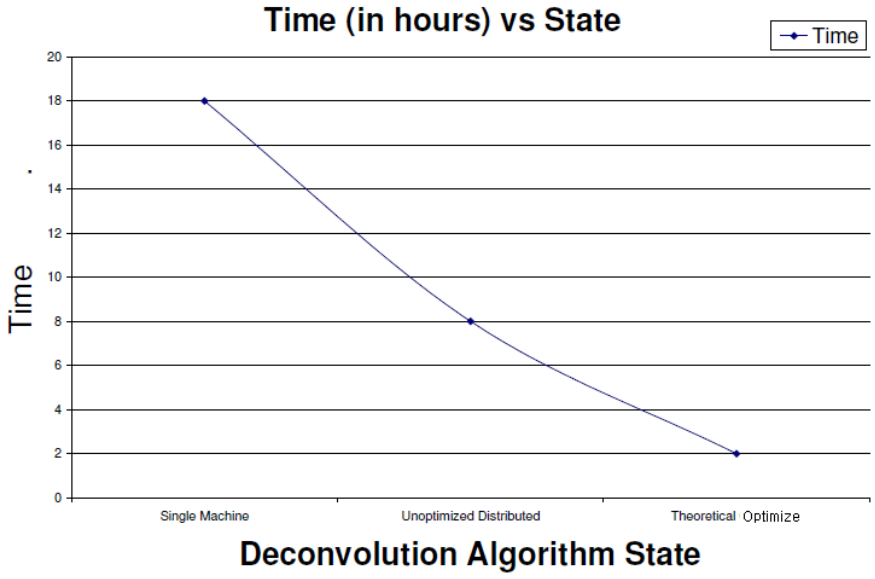


Figure 3.6: Results from Palmetto for the Distributed Deconvolution Algorithm

3.3.1 Results from Palmetto

It has been possible to take a deconvolution algorithm that has run on a quad core Intel based workstation in 18 hours and make it run in 8 hours, with a theoretical run time of 2 hours under optimal conditions. After the initial run it was discovered that there were a subset of nodes that took an order of magnitude longer to execute the code, so a script was written to find these outliers and remove them from the list of possible nodes to run on. There are several places where the FFT could be parallelized with threads as a parallelized sum. Threading the sum as well as minimizing job runtime by using the node selecting script would give us the theoretical runtime of 2 hours. Figure 3.6 shows the results from Palmetto. These results show that Condor can decrease the run time of an unoptimized distributed deconvolution algorithm by 66.6% with a possibility of gaining up to 89.9%.

3.3.2 Group Acknowledgments for the Deconvolution Project

Ben Sterrett was the main Matlab programmer, Linton Abraham provided MPI and C language consulting, Logan Johnson was the main Biology contact and wrote the original Matlab code that served as a base for the entire project. Dr. Mount is the main faculty member on this project, and Sebastien Goasguen is the secondary faculty contact from the School of Computing.

Chapter 4

Building a Campus Grid at Clemson

The cyberinfrastructure at Clemson has been successful at providing a compute resource for university research as well as giving back to the scientific community abroad. The following sections discuss the evolution of the Campus Grid, as well as collected statistics about the Clemson CI, and also present experiments which show a marked improvement in throughput with the job_router. The final section will review the Power consumption that Condor creates and what can be done to mitigate Condor's impact on the environment.

4.1 Condor Across Campus

Based on prior work the Clemson Condor pool was designed to be robust and grid aware. Figure 4.1 shows the basic structure of the Clemson Condor pool. The Central manager at Clemson (which is backfilled by BOINC) can flock to the Central manager of Palmetto (which is backfilled by the Open Science grid). The first full scale test run of Condor at Clemson University campus took place in the Fall of 2007 in one of the School of Computing's public labs, which was proceeded by a week of testing with a small number of old Dell 280s. When the initial testing with the small group of Dells and the single lab were complete CCIT was approached with placing Condor on each machine on campus. Special consideration for this pool had to be made because it consisted of

exclusively Windows machines (at that time) and had no shared file system between the computers. The lack of a shared file system presented a unique problem in and of itself, because it limited the types of jobs that could be sent to the pool; Condor on the Windows machines would have to rely on Condor's native file transfer mechanism. The pool started seeing widespread use within a few months of deployment, but there were long periods of idle time. The computers were still physically on during the idle time, thus using power but not working. The idle time study discussed in the introduction was reenacted for the Condor Pool, and it was found that large amounts of idle runtime still occurred, which can be viewed in Figure 4.2. This prompted the addition of BOINC to take up the additional idle time, which is shown in figure 4.3. Notice that the Windows XP machines have 98% with a 95% total usage for the pool that is thrown off by linux machines that are never targeted and that do not have BOINC installed on them, but even with the addition of the Linux machines the pool usage is up from 31% to 98% from Feburary 2007 in Figure 4.2 to July 2008 in figure 4.3. This significant change in usage will be explored in depth in the conclusion of this thesis.

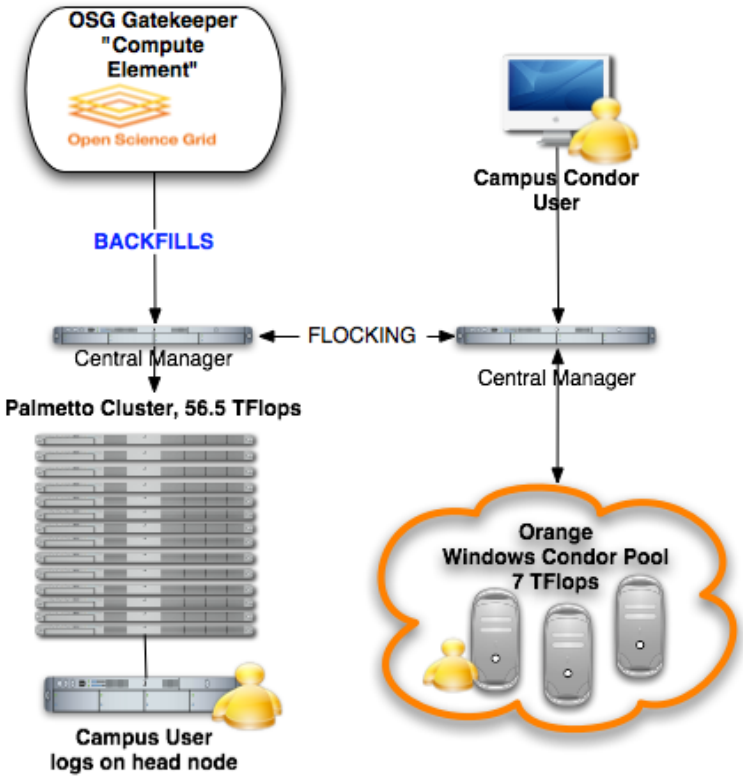
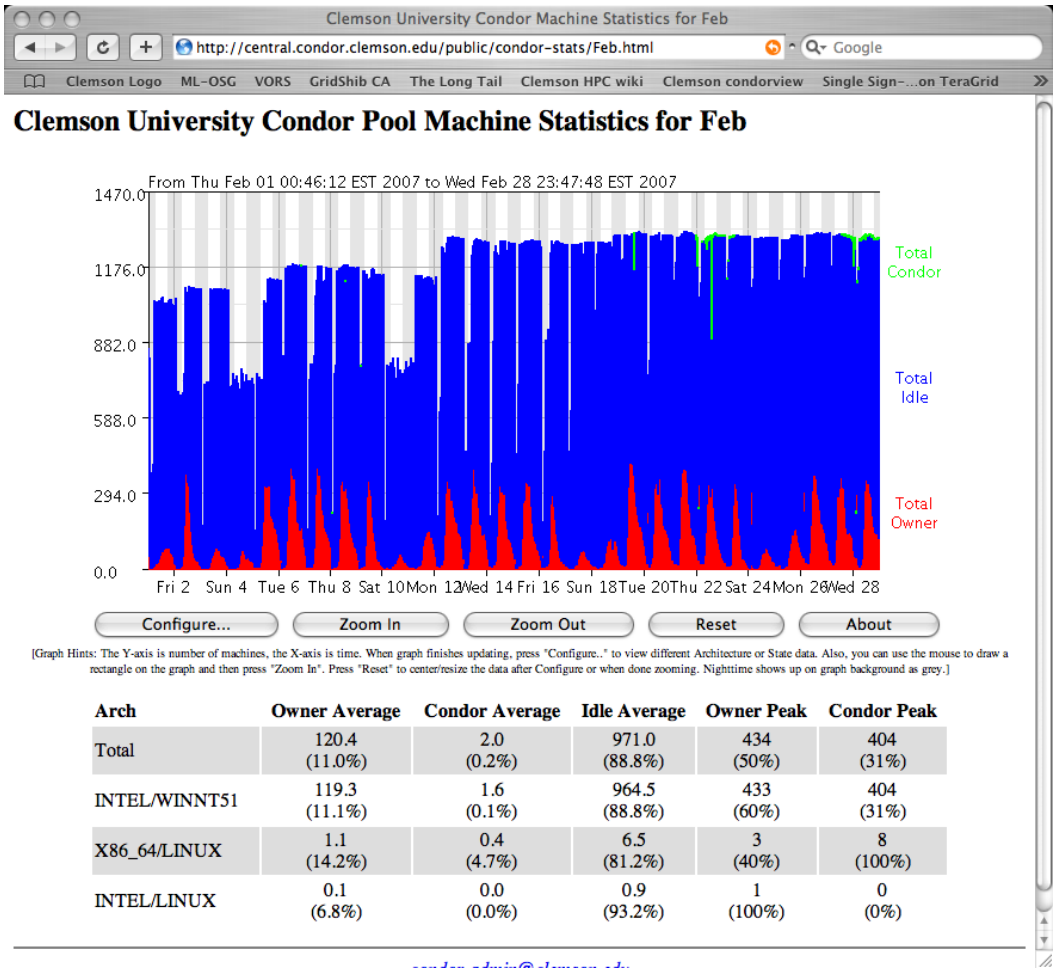
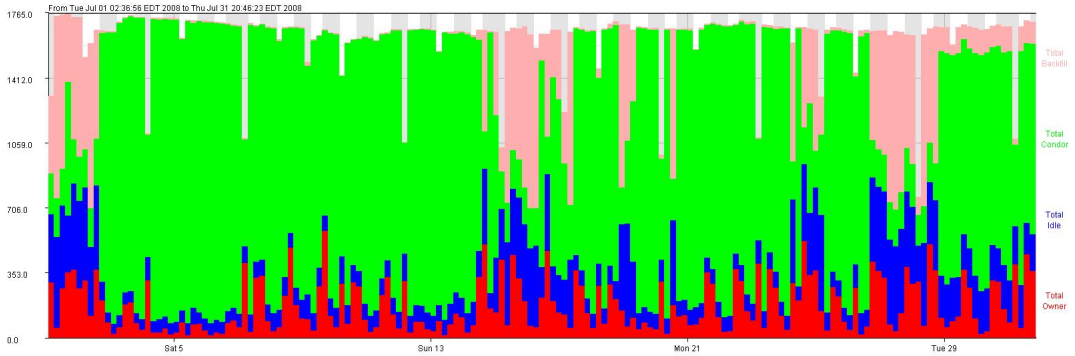


Figure 4.1: Clemson Condor Architecture, Diagram Source: Sebastien Goasguen



Arch	Owner Average	Condor Average	Idle Average	Owner Peak	Condor Peak
Total	120.4 (11.0%)	2.0 (0.2%)	971.0 (88.8%)	434 (50%)	404 (31%)
INTEL/WINNT51	119.3 (11.1%)	1.6 (0.1%)	964.5 (88.8%)	433 (60%)	404 (31%)
X86_64/LINUX	1.1 (14.2%)	0.4 (4.7%)	6.5 (81.2%)	3 (40%)	8 (100%)
INTEL/LINUX	0.1 (6.8%)	0.0 (0.0%)	0.9 (93.2%)	1 (100%)	0 (0%)

Figure 4.2: Output From Condor View for the month of February 2007



Arch	Owner Average	Condor Average	Idle Average	Owner Peak	Condor Peak	
INTEL/WINNT51	167.0 (11.0%)	1109.3 (69.1%)	140.2 (8.9%)	176.0 (11.0%)	583 (44%)	1658 (98%)
Total	170.7 (10.9%)	1112.8 (67.3%)	179.6 (11.1%)	177.1 (10.8%)	614 (42%)	1662 (95%)

Figure 4.3: Output From Condor View for the month of July 2008

4.2 Condor Pool Usage and Administration

The Clemson Condor pool started with a single dual core Windows XP machine two years ago and now consists of nearly a thousand dual and quad core Windows XP and Vista SP2 machines. The increase in size and use of the Condor pool made it necessary to have a researcher monitoring the Condor pool and preparing updates, as well as added part-time administrators to train and provide assistance to users. Figure 4.4 shows a sample command that can be run to check which users are submitting jobs to the pool. The Condor pool is open to the faculty and students by allowing them to create and submit their own code with no oversight procedure. This was taken to be an acceptable risk to promote the Condor pool's user. Figure 4.5 shows Condor use during the month of May, and clearly shows the bursty use of Condor when over 40000 jobs were queued onto the pool by a single user on the first of the month. The user who submitted these jobs happened to be an industrial engineer, but jobs come from a diverse group of domains that range from business process simulations to deconvolution algorithms in biology. The largest number of jobs ever successfully submitted to Condor at one time is 300,00. Job scaling problems occurred when five hundred thousand jobs were submitted to the pool and crashed the central manager. The central manager was rebooted and its scratch directory was deleted to rectify the problem. Node scaling problems occurred when the dual core machines were replaced with quad core machines last summer. The actual number of machines remained the same, but the number of cores and thus the number of slots screened by the collector daemon increased from 700 to over 2,600. To temporarily fix this problem, a larger port range has been opened on the central manager which roughly gives Condor two ports per slot; more machines have been requested to house collector daemons which will lighten the load on the one collector/negotiator for the pool.

Recently the Condor administrator left CCIT, which has caused several problems because there must be a minimum number of support staff for a site of this size. Many support issues have since arrisen and meetings have been called to try and fix pervasive problems brought up by users. There has been a break down of communication between the new part time administrators and the users who have felt that their concerns were not being answered. Condor has been disrupted several times because of a regression in version 7.2.* of the Condor scheduler which has prompted CCIT to search for a fulltime Condor administrator. This person's first job will be to make sure that Condor is running correctly. The research group that started to project believed that the Condor

```
dsepulv@gecko10$ condor_stats -userlist
squire2@gecko4.cs.clemson.edu/gecko4.cs.clemson.edu
kpoulse@frog3.cs.clemson.edu/frog3.cs.clemson.edu
shalikj@frog31.cs.clemson.edu/frog31.cs.clemson.edu
rahuls@gouraud2.cs.clemson.edu/gouraud2.cs.clemson.edu
vjain@frog31.cs.clemson.edu/frog31.cs.clemson.edu
asegars@gecko1.cs.clemson.edu/gecko1.cs.clemson.edu
qiz@clemson.edu/user001.palmetto.clemson.edu
mahakj@frog29.cs.clemson.edu/frog29.cs.clemson.edu
asegars@gecko5.cs.clemson.edu/gecko5.cs.clemson.edu
wolf@clemson.edu/wcnitc.clemson.edu
tjcurti@frog29.cs.clemson.edu/frog29.cs.clemson.edu
sjain@frog29.cs.clemson.edu/frog29.cs.clemson.edu
ankity@frog31.cs.clemson.edu/frog31.cs.clemson.edu
lstout@vector10.cs.clemson.edu/vector10.cs.clemson.edu
buml@gecko6.cs.clemson.edu/gecko6.cs.clemson.edu
lstout@frog3.cs.clemson.edu/frog3.cs.clemson.edu
eduffy@kamino/kamino
amcdoug@frog17.cs.clemson.edu/frog17.cs.clemson.edu
richard@gecko3.cs.clemson.edu/gecko3.cs.clemson.edu
shalikj@frog30.cs.clemson.edu/frog30.cs.clemson.edu
bsterr@clemson.edu/osggate.clemson.edu
amcdoug@frog24.cs.clemson.edu/frog24.cs.clemson.edu
Steve@Steve-PC/Steve-PC
```

Figure 4.4: Sample Administrative Command To Check Users

pool was still in Beta, but because real users were running code on the pool it was deemed to be a "production" system and administrative control was moved to CCIT, where the lack of staff has caused problems that are now being addressed.

4.3 BOINC

4.3.1 BOINC Time Donated

Clemson University has donated over 500 years to humanitarian projects through the World Community Grid (WCG) and has expanded project support to Einstein@home and LHC@home. Figure 4.8 shows how the backfill system has grown over the past year, starting from one test machine in March and moving to the entire university by August, and then slowly splitting more time between WCG, Einstein@home, and LHC. The impact of users returning from summer break is noticeable from August to September, but is a good indicator that Condor is correctly preempting BOINC for users. Clemson has been recognized in several national and international venues such as International Science Grid This Week [44] and the WCG News and Reports [41] as a top contributor to humanitarian projects as a college, for both the United States and the world. The Figure 4.6 shows Clemson University in the number 5 position for the world, which was gained in under one

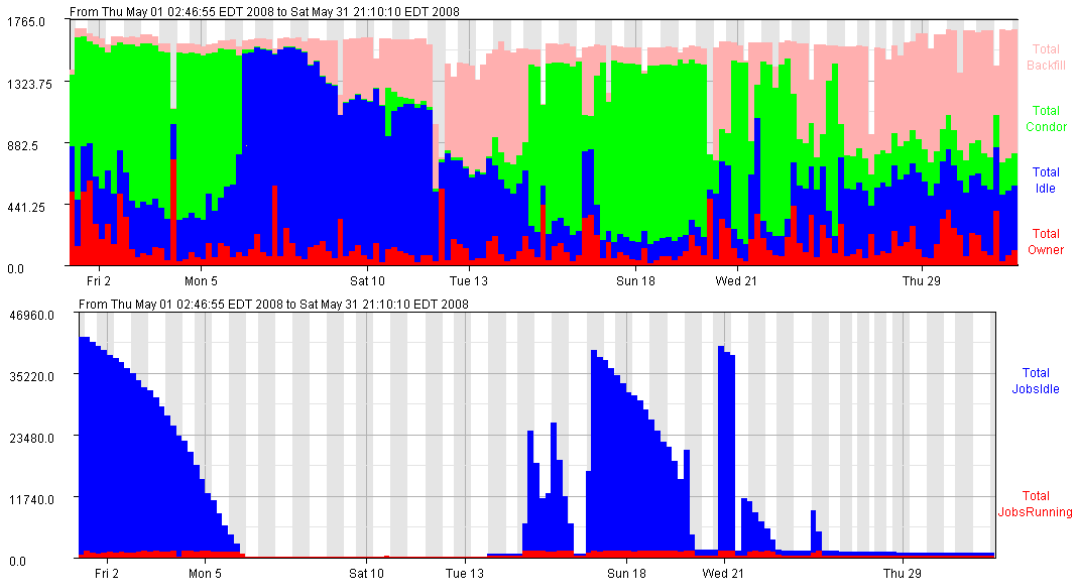


Figure 4.5: Output From Condor View for the month of February 2007

year's time. Figure 4.7 shows the total time donated to humanitarian projects through the World Community Grid, notice that is is over 500 years.

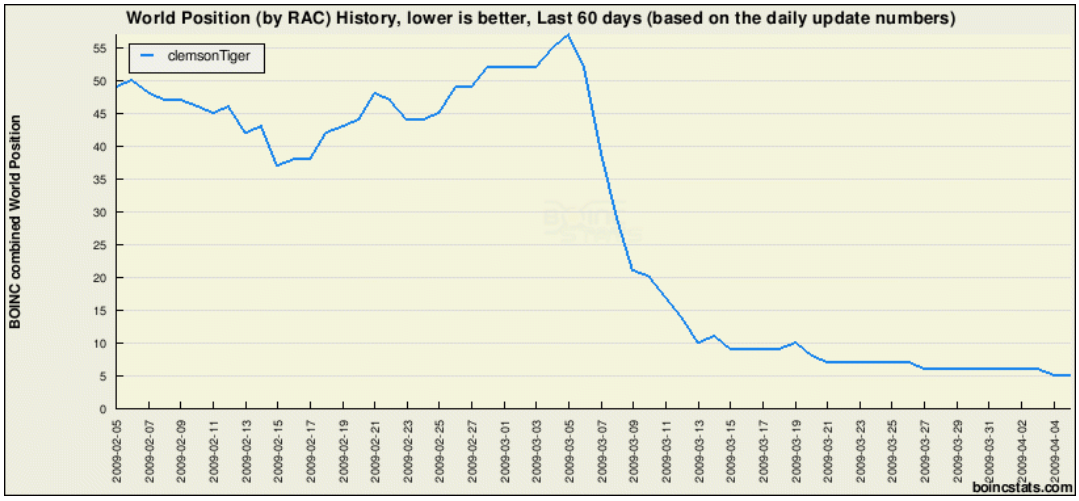


Figure 4.6: Current World Position of Clemsontiger

Welcome back **clemsontiger**

Registered Member Since: 2/14/08 19:19:38

My Statistics		My Team: Clemson School of Computing	
Total Run Time (y:d:h:m:s) (Rank)	576:330:05:23:58 (#13)	Total Run Time (y:d:h:m:s) (Rank)	597:310:00:08:46 (#34)
Points Generated (Rank)	543,788,599 (#7)	Points Generated (Rank)	561,617,950 (#19)
Results Returned (Rank)	834,807 (#11)	Results Returned (Rank)	860,778 (#31)

Figure 4.7: Statistics for World Community Grid

	Statistics Date	Total Run Time	Results Returned	
	11/11/08	2:283:21:19:32	3,661	
	10/19/08	3:150:17:17:06	4,612	
Split Proj. ->	9/11/08	3:138:01:40:35	5,200	
All Labs ->	8/25/08	4:191:01:58:08	7,446	<- Summer
	7/15/08	1:002:03:45:11	894	
	6/16/08	1:282:06:44:46	1,832	
	5/14/08	0:162:17:01:21	565	
	4/13/08	0:015:12:48:36	29	
Day One ->	3/29/08	0:000:13:06:38	1	

Figure 4.8: Sample Results for WCG, One Day A Month

4.4 Birdbath for Viewing Stats

The Simple Object Access Protocol is one of the standards for sharing formatted information over the Internet to and from web services. [35] Its purpose is to make information exchange between applications and web services transparent from end to end. [27] In the previous work section it was noted that NYS Grid has had a hand in the formation of a binary friendly version of SOAP called bSOAP and other solutions have been proposed to make SOAP perform better in certain situations [43], but the majority of SOAP is a simple XML tree designed to facilitate the exchange of information between resources on the web. The SOAP standard uses the W3C controlled schema for XML (http://www.w3.org/XML/Schema) and is highly contested as a 'good' choice for information exchange because it trades speed for interoperability.

Birdbath is the code name for the SOAP interface to Condor that was developed by the Condor team in 2007. [12] This program was originally designed to run through Mono, but a new .NET interface was designed so that clear interface could be designed for users that were not familiar

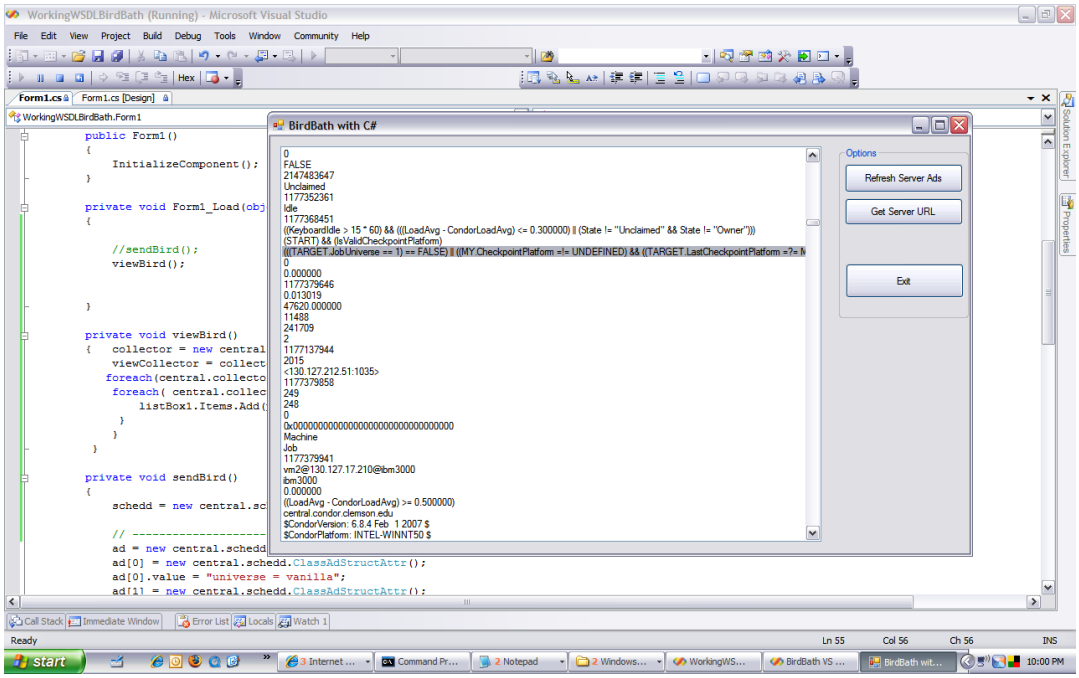


Figure 4.9: Snapshot of Early Birdbath Interface

with command consoles. Figure 4.9 shows a window with the output from condor_status which was taken from the soap interface. The below code shows how simple it is to gather information from the public WSDL located with the Condor collector and use it to interface with SOAP enabled Condor daemons. After several weeks of working with Birdbath it was found that job submission was not possible with the current libraries available from WISC, but data could be gathered and put into forms that are easy to read and manipulate via the .NET graphical programming environment.

```
namespace BirdBathV2
{    public partial class Form1 : Form
    {        // non condor Variables
        int hash;
        int searchType;
        /*  WSDL from http://central.condor.clemson.edu:9618/condorCollector.wsdl
         "condorCollector" Description
            Documentation
            gSOAP 2.7.6c generated service definition
            Methods
            getPlatformString ( ) As StringAndStatus
            Service definition of function condor__getPlatformString
```

```
getVersionString ( ) As StringAndStatus
Service definition of function condor__getVersionString
insertAd ( type As ClassAdType ,  ad As ClassAdStruct ) As Status
Service definition of function condor__insertAd
1 queryAnyAds ( constraint As string ) As ClassAdStructArray
Service definition of function condor__queryAnyAds
2 queryLicenseAds ( constraint As string ) As ClassAdStructArray
Service definition of function condor__queryLicenseAds
3 queryMasterAds ( constraint As string ) As ClassAdStructArray
Service definition of function condor__queryMasterAds
4 queryScheddAds ( constraint As string ) As ClassAdStructArray
Service definition of function condor__queryScheddAds
5 queryStartdAds ( constraint As string ) As ClassAdStructArray
Service definition of function condor__queryStartdAds
6 queryStorageAds ( constraint As string ) As ClassAdStructArray
Service definition of function condor__queryStorageAds
7 querySubmittorAds ( constraint As string ) As ClassAdStructArray
Service definition of function condor__querySubmittorAds
```

4.5 Condor job_router Speedup

The Condor job_router is a grid aware daemon that allows Condor jobs sent through a Globus interface to run on federated grid sites and allows those same sites to send jobs to be run on Clemson's condor pool. The job_router can take the load off of the local Condor pool during peak usage and give better total throughput to jobs sent at any time, by effectively adding more nodes to the pool from participating sites. The job_router is added to the pool by enabling the daemon in the condor_config and adding a list of available sites to the daemons configuration file, which in practice is kept separate from the local configuration file. The Resource Selection Service (ReSS) which is a service provided by participating grid sites can be used to automatically configure the list of available sites to keep the job_routers list up to date. ReSS alone is not sufficient though, because there is no quality control in the automatically generated list.

It was found that simple changes to the job_router routing table can make a large difference in throughput. The algorithm that the job_router uses to schedule the jobs begins with very high throughput but ends with a long tail of idle jobs. This asymptotic behavior makes sending a few small jobs impractical, so it is most useful for many long running jobs. The Figure 4.10 solid line

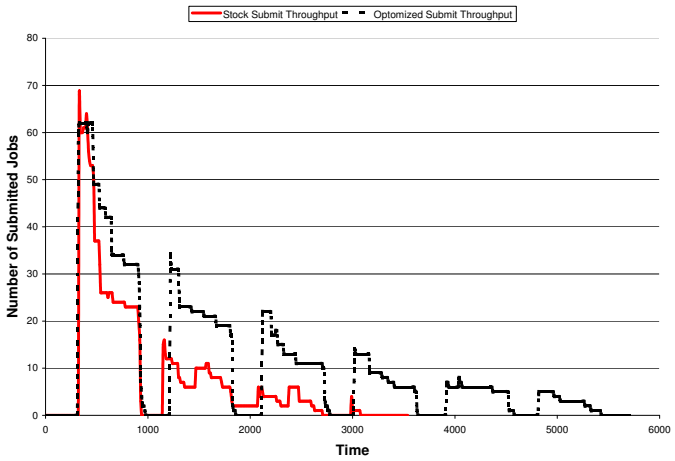


Figure 4.10: Stock vs Optimized Condor Job_Router

was created from information gathered by the Condor Router Log Parser (CROP) and shows the Condor job_router using the unaltered ReSS routing table to be unpredictable and ultimately results in a crash of the job_router daemon. Each site was tested by submitting Condor jobs through the job_router and reviewing the job logs to see which sites never ran jobs. After the sites were removed that would not run jobs and the sites that were slow, we increased the number of idle jobs that could be at each site and resent the jobs, the result was a uniform graph (Figure 4.10, broken line) that showed a marked improvement in submitted job throughput time as compared to other tests. The uniform graph indicates that jobs are being sent in waves, completed, and then returned without error. It appears that the optimized graph takes longer, but the unoptimized graph has a tail that extends past the end of the graph because it caused the schedd to crash. ReSS returns some non-functional sites because when the OSG software stack is installed on a submit machine it automatically joins ReSS regardless of whether Condor has been properly configured. The process of testing and removing faulty sites from ReSS could be automated into a QoS system that runs test jobs from time to time and then updates the routing table with good sites, thus ensuring that new sites were added to the table when they were properly configured.

4.6 Power Consumed by the Pool

Conserving power has become a top environmental concern due to rising fuel prices and a declining world economy. In response to the growing concern over power conservation and cost, we monitored the power usage of a teaching lab with stock Dell Optiplex 755's to determine how much extra power was being used by the systems running Condor and BOINC as opposed to the control systems that are not running either. We attached three randomly selected computers in the teaching lab to Kill-A-Watt power monitoring strips, which can be seen in Figure 4.11, and ran each in the following states for approximately one week: idle with no operating system, Windows Vista SP1 with Condor and BOINC, Windows Vista SP1 without Condor and BOINC, and Windows Vista SP1 isolated with no lab user contact. The results, which can been seen in Table 1, were disappointing but not unexpected when we considered that an unused machine without Condor and BOINC goes into sleep mode, which uses almost no power at all. The cost per kilowatt was extrapolated from a ITC bill, so if there is any error in this calculation the price shown in the following tables would be significantly different. Table 2 shows the extrapolated cost for the university for the entire year. Condor can be configured so that it only runs at particular times, which would limit power usage by BOINC, however we believe that the cost of running the backfill system in its current state is offset by the cross cultural impact of the humanitarian projects that benefit from the time Clemson donates to their causes. The following section will discuss several possible strategies that Clemson is considering to offset both the cost to the environment and the cost to the school.

4.6.1 Offsetting the Cost

The costs of running a backfill system can be measured in two ways: cost per kilowatt hour of backfill time minus cost per kilowatt hour of user time, and cost in carbon. The cost in carbon will be addressed in this Thesis. There have been studies that indicate storing carbon in trees may not be the best decision in the long run [36], but it can buy us time for now and is a viable way to offset the carbon generated by the production of electricity needed to run and cool the machines. It can be estimated that the carbon generated by the backfill system at Clemson University could be offset by planting 19 trees [46] per year.

Figure 4.11: Kill-A-Watt Power Strip Used In Clemson's labs

Table 4.1: Power Usage and Cost in a Teaching Lab at Clemson University per Computer

State	Avg. KwH/Day	Cost per year (at $0.14/KH)
BOINC&Condor	2.0534	$104.93
User Only	0.4116	$21.32
Machine Isolated	0.0552	$2.82

Table 4.2: Total Computer Power Usage and Cost at Clemson University with Approximately 2500 Machines

State	Avg. KwH/Day	Cost per year (at $0.14/KH)
BOINC&Condor	5133.5	$262,321.85
User Only	1029.0	$52,581.90
Machine Isolated	138.0	$7,051.80

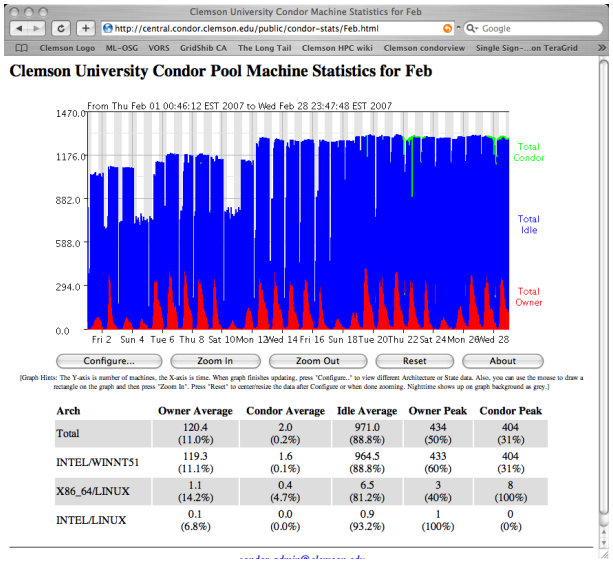
Chapter 5

Conclusions and Discussion

In this thesis the author has discussed how to use Condor with BOINC Backfill and OSG to aggregate commodity machines into a usable grid service and make sure that almost no CPU cycles are wasted. Proof of this progression from less than 7% usage to more than 98% usage can be seen in the following charts, the first of which is from Febuary of 2007 and the last of which is from July of 2008. A job_router was added to the pool to give Clemson a computational cushion during peak need bursts. Statistics and experiments were presented to show where Clemson Campus Grid excels and where and how it can be improved. Strategies were used and identified that private institutions and companies as well as public institutions can use to help offset the cost of using a backfill system. The Condor soap interface was explored to find new ways of displaying data to users. An OSG site was created to test the VDT and provide a small resource for Engagement VO members. With the proper motivation and social networking an institution can use preexisting technologies to create a fully functional Campus Grid that will not only meet the needs of the local users, but also provide an overhead of CPU time that can be donated to humanitarian projects around the world for the advancement of science and the benefit of mankind.

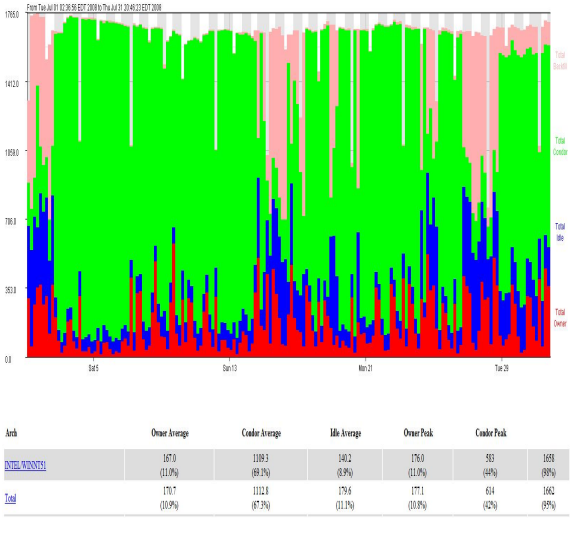
5.1 Answering the Research Question

The results of this research have shown that it is possible to come very close to 100% (98% according to Figure 5.2) utilization of avaliable resources, but at a cost in power and administration. The high utilization is the result of using Condor, BOINC, and the Open Science Grid through



Arch	Owner Average	Condor Average	Idle Average	Owner Peak	Condor Peak
Total	120.4 (11.0%)	2.0 (0.2%)	971.0 (88.8%)	434 (50%)	404 (31%)
INTEL/WINNT51	119.3 (11.1%)	1.6 (0.1%)	964.5 (88.8%)	433 (60%)	404 (31%)
X86_64/LINUX	1.1 (14.2%)	0.4 (4.7%)	6.5 (81.2%)	3 (40%)	8 (100%)
INTEL/LINUX	0.1 (6.8%)	0.0 (0.0%)	0.9 (93.2%)	1 (100%)	0 (0%)

Figure 5.1: Usage Before Backfill



Arch	Owner Average	Condor Average	Idle Average	Owner Peak	Condor Peak
INTEL/WINNT51	167.0 (11.0%)	1109.3 (69.1%)	140.2 (8.9%)	176.0 (11.0%)	583 (44%)	1658 (69%)
Total	170.7 (10.9%)	1112.8 (67.3%)	179.6 (11.1%)	177.1 (10.8%)	634 (42%)	1662 (69%)

Figure 5.2: Usage After Backfill

Globus to schedule both local and remote jobs on Clemson's Condor pool. With the small amounts of overhead allowed for transitions between Condor states it does not seem likely that a higher percent usage could be easily attained. The Condor pool has been built by a graduate student with the help of School of Computing and CCIT administrators using tips and techniques recomended by other institutions who already have functional Campus Grids. By not repeating the mistakes noted by others and given the clean track record of the Condor pool, it can be argued that the subjective goal of creating the "best designed" campus grid has been meet.

5.2 Future Work

As advanced as the Clemson Campus Grid is there can always be improvements to user ease of use, security, and administration. A web portal should be created to support Clemson's Condor pool with a graphical interface and a single sign on system that integrates with the Open Science Grid. This would facilitate cross disciplinary use of the resources available at Clemson University. The web portal would address added security, but hiring a dedicated administrator whose first priority was to make sure the Clemson Condor pool was functioning correctly would be the best security upgrade. A innovative shared file system that the Windows machines could use would add more usability to the pool because Condor would not have to invoke its own file transfer mechanism.

Appendices

Appendix A XP BOINC and Condor How-To Guide

HOWTO: Condor with BOINC Backfill Windows XP SP2

Written by Dru Sepulveda

Thursday, 08 January 2009 13:49 - Last Updated Friday, 16 January 2009 14:31

A tutorial on how to install Condor and BOINC for a backfill system using Windows XP SP2

History:

Created: 2/1/08

Edited: 2/4/08, Added step five for cert authentication

Edited: 2/8/08, Changed attached config file and section Step 4 to reflect new BOINC_Owner macro

Edited: 3/7/08, Added new client install process with custom client from WCG, specifically for WCG projects, altered latest versions and addressed the unregistered machines problem, and many other small changes.

Edited: 3/31/08, Changed condor_config.local to account for unexplained problems with the machinebuisy macro.

Edited: 4/11/08, Included a fix for the Cant connect to local master error found by Matt Rector, along with some minor updates.

Edited: 4/14/08, Specified OS in title, updated contact email, added new Testing area, noted the config_file specification, added disclamer, added future work

Introduction and Acknowledgements:

Using the backfill option on Condor allows BOINC to run jobs when there are no Condor Jobs currently running and the machine is sitting idle. This allows for maximum CPU usage on a given workstation. This process was derived from (http://www.cs.wisc.edu/condor/manual/v6.7/3_13Setting_Up.html) and conversations with Prof. Sebastien Goasguen and Ben Code Monkey Burnett from WISC. Experiences with Matthew Rector and John Mark Smotherman also influenced this tutorial.

Disclaimer:

Use at your own risk.

Process:

Step 1: Install BOINC

a. Download the latest self-registering BOINC client from WCG as outlined in http://www.worldcommunitygrid.org/bg/BOINCMassInstall.pdf from http://boinc.berkeley.edu/dl/wcg_mass_boinc_5.10.21_windows_intelx86.exe and follow the

HOWTO: Condor with BOINC Backfill Windows XP SP2

Written by Dru Sepulveda

directions from section 5, which I have outlined in steps b through f.
b. Install BOINC using the Shared Installation so that all users have access to executables.
c. During the installation process deselect the option to make BOINC the default screen saver and deselect the option to run on start-up or log-in.
d. Do not launch BOINC at the conclusion of the installation
e. Do not alter the cc_config.xml, this causes a race condition with condor and makes massively large file with non-standard character sets that windows can not delete. To fix this problem, restart windows and let windows run scan disk.
f. Do not alter the account_www.worldcommunitygrid.org.xml file, Condor will generate a new one in the directory defined in the condor_config.local file.
g. Be sure to set your default profile at www.worldcommunitygrid.org so that when machines join WCG they use memory, CPU and backfill within parameters.
Notes: BOINC for Windows has two .exe files, boinc.exe is the client and boinccmd.exe is the manager tool. For all operations you want to point Condor at boinc.exe. Also, when using BOINC on Windows, BOINC_InitialDir will not be respected fully. That is to say, you must pass the BOINC home directory directly to the BOINC application, via the BOINC_Arguments line. The implications of this will be explained in the configuration section. The uninstall file that comes with the WCG mass installer does not work correctly, so if uninstalling is necessary the windows uninstaller from control panel should be used. If, at the end of step 6 in this paper, there are multiple BOINC.exes running under the defined user account, the Service Instillation was chosen over the Shared Instillation option. To fix this problem the current client must be uninstalled and reinstalled with the proper selections.
Step 2: Install Condor
a. Download the latest Condor Client which is 7.0.1
b. Set the Central Controller to: central.condor.clemson.edu
c. Do not start condor at the end of the install
Step 3: Configure Condor for backfill jobs.
a. Take the contents from the attached config file and place them into condor_config.local (this includes the fix for the above note)
b. YOU MUST alter the condor_config.local to meet your needs (file is well documented)
c. The key provided for WCG is not a real key. A real key may be obtained from Prof. Goasguen for the clemsonTiger account.
d. Never use old condor_config files from previous versions of Condor, always copy your custom settings from your old condor_config to the new one.
e. Questions associated with each of the variables in the config file may be resolved by going to http://www.cs.wisc.edu/condor/manual/v6.7/3_13Setting_Up.html#SECTION
00413900000000000000

Step 4: Make Files
a. Make the directory condor/boinc or D:Data (specific to our system), this file must survive a reimage.
b. Make the files boinc.err and boinc.out with permissions granted for read, write, and execute for the selected user account.
c. Remove the read only attribute
d. Enable advanced file view if not already enabled
e. Set the file condor/boincs group to everyone and give all controle
f. Save settings and close
Notes: Future versions will have a Boinc_User variable that allows Condor to run BOINC jobs as a specific user and will mitigate steps c through f of step 4.
Step 5: Set Certificates:
a. Move the certificate ca-bundle.crt, which was generated at the time BOINC was installed in C:Program FilesBOINC to ../condor/boinc, or the file specified in your BOINC_HOME variable. This allows BOINC client to authenticate itself to projects.
Step 6: Start Condor using Net commands
a. Version 7.* of condor supports new commands and will sometimes fail when using condor_on to start the deamon. Instead, net start condor should be used to get condor going.
b. For good measure also do a condor_on and condor_reconfig after condor is started.
c. (To stop condor use condor_off followed by net stop condor if deamons are still running)
d. It may be necessary for a Condor credential to be added for the user account that BOINC backfill is running as. This may be done (the simple way) as logging in as the backfill account, opening the cmd, and typing condor_store_cred add, then follow the on screen instructions. This account MUST have a password associated with it. If it does not, add one.
e. ERROR: Because Condor was not started by the installer, no Condor_Master is currently running to accept your commands or you did not start condor using the Net commands as stated in part a of this step. This means that running a condor_q or a condor_store_cred will cause the Can not connect to local master error. To fix this, go to ../condor/sbin and run the ./Condor_Master.exe program, then ./condor_on to spawn all the deamons.
Step 7: Testing:
a. There are several good ways to test your setup; the first is to check your logs. These logs are

located at ../condor/log and the location that you set BOINC_HOME to in your condor_local

configuration file. SchedLog and MasterLog have good information about how condor is doing. If you are having problems, look here first.

b. The second log to be aware of is StarterLog.boinc, which as the name implies, has information about how the backfill setup is working. If you are having problems with backfill or BOINC, this is where to look.

If all is correct the log file condor/log/StarterLog.boinc will look like:

```
2/1 12:50:42 ******************************************************
2/1 12:50:42 ** condor_starter (CONDOR_STARTER) STARTING UP
2/1 12:50:42 ** C:condorbincondor_starter.exe
2/1 12:50:42 ** $CondorVersion: 7.0.1 Jan 31 2008 BuildID: none PRE-RELEASEUWCS
$
2/1 12:50:42 ** $CondorPlatform: INTEL-WINNT50 $
2/1 12:50:42 ** PID = 508
2/1 12:50:42 ** Log last touched 2/1 12:50:38
2/1 12:50:42 ******************************************************
2/1 12:50:42 Using config source: C:condorcondor_config
2/1 12:50:42 Using local config sources:
2/1 12:50:42 C:condor/condor_config.local
2/1 12:50:42 DaemonCore: Command Socket at
2/1 12:50:42 Setting resource limits not implemented!
2/1 12:50:42 Starter running a local job with no shadow
2/1 12:50:42 Getting job ClassAd from config file with keyword: "boinc"
2/1 12:50:42 "boinc_proc" not found in config file
2/1 12:50:42 setting the orig job name in starter
2/1 12:50:42 setting the orig job iwd in starter
2/1 12:50:43 Job 1.0 set to execute immediately
2/1 12:50:43 Starting a VANILLA universe job with ID: 1.0
2/1 12:50:43 Tracking process family by login "condor-reuse-slot1"
2/1 12:50:43 IWD: C:condorboinc
2/1 12:50:43 Output file: C:condorboincboinc.out
2/1 12:50:43 Error file: C:condorboincboinc.err
2/1 12:50:43 Renice expr "10" evaluated to 10
2/1 12:50:43 About to exec C:PROGRA~1BOINCboinc.exe --dir C:condorboinc --
attach_project http://www.worldcommunitygrid.org/ cbf9d0enot realcode035b4b2
2/1 12:50:43 Create_Process succeeded, pid=4036
```

and condor/boinc (or D:Data) will be filled with files pertaining to jobs.

c. The third set of logs is located in the BOINC_HOME directory. boinc.out and boinc.err will tell you what BOINC is doing independent of Condor. Check here if you are unsure if BOINC is connecting to WCG or is not restarting jobs after preemption.

d. If you are having problems with console commands, try adding a d argument. This causes the error response from condor to be printed to your screen, which can be handy for diagnosing

problems.

e. Try running condor_q from the ../condor/bin directory to make sure the IP displayed is the IP of the machine. In rare instances Condor has connected itself to the loopback Ethernet interface.

f. If you would like to dynamically check if your backfill is working with out disturbing the machines you may run condor_status | grep Backfill from ../condor/bin to see if your machines are running (I am assuming you have Windows Tools for Unix installed).

g. If you want to make sure that your backfill settings allow condor to preempt BOINC for a in-house job, give your backfill machine a custom machine ad and send it a job, making note of the time. Check the log StarterLog.boinc to see if BOINC was killed around the time you sent the job.

h. If you want confirmation fast, set condor to run jobs all the time (explained in the condor manual or config file) and check the task manager. Two startds, BOINC as the specified user, and one or more WCG_* threads will be running if all is correct.

Future Documents:

Hopefully there will be a guide on how to set Condor backfill up under Windows Vista in the near future.

Sample Configuration

```
##
# Sample BOINC configuration
##
# Turn on backfill functionality, and use BOINC
ENABLE_BACKFILL = TRUE
BACKFILL_SYSTEM = BOINC
# Spawn a backfill job if we've been Unclaimed for more than 5
# minutes
START_BACKFILL = $(StateTimer) > (5 * $(MINUTE))
# Evict a backfill job if the machine is busy (based on keyboard
# activity or cpu load)
# EVICT_BACKFILL = $(MachineBusy) # this macro does not work as expected
EVICT_BACKFILL = $(KeyboardBuisy)
# Define a shared macro that can be used to define other settings.
# This directory must be manually created before attempting to run
```

```
# any backfill jobs. This should be read/writable by all users.
BOINC_HOME = $(LOCAL_DIR)/boinc
# *NIX:
5 / 6
HOWTO: Condor with BOINC Backfill Windows XP SP2
Written by Dru Sepulveda
Thursday, 08 January 2009 13:49 - Last Updated Friday, 16 January 2009 14:31
# Path to the boinc_client to use, and required universe setting
#BOINC_Executable = /usr/local/bin/boinc_client
#BOINC_Universe = vanilla
#BOINC_Arguments = --attach_project http://einstein.phys.uwm.edu/ [key]
# WINDOWS: (note the use of the "--dir" option to boinc.exe, this is required)
# Path to the boinc_client to use, and required universe setting
BOINC_Executable = C:PROGRA~1BOINCboinc.exe
BOINC_Universe = vanilla
#BOINC_Arguments = --dir $(BOINC_HOME) --attach_project http://einstein.phys.uwm.edu/
[key]
BOINC_Arguments = --dir $(BOINC_HOME) --attach_project
http://www.worldcommunitygrid.org/ cbf9dNOTAREALKEYGETYOUROWN035b4b2
# What initial working directory should BOINC use?
BOINC_InitialDir = $(BOINC_HOME)
# Save STDOUT and STDERR
BOINC_Output = $(BOINC_HOME)/boinc.out
BOINC_Error = $(BOINC_HOME)/boinc.err
# What user the boinc client program should be run as. This
# macro is only used if the Condor daemons are running as root.
# In this case, the condor_starter must be told what user
# identity to switch to before spawning the boinc client. This
# can be any valid user on the local system, but it must have
# write permission in whatever directory is specified in
# BOINC_InitialDir).
BOINC_Owner = "DOMAIN""USERNAME"
STARTER_ALLOW_RUNAS_OWNER = TRUE
6 / 6
```

Appendix B Condor Dynamic Configuration Example

```
 ############################################################
## Warning this file is updated every 30-40 minutes
## and should NEVER be altered by hand.  For configuration
## information see cs.clemson.edu/~dsepulv/dynamic.html
############################################################

## Created by CirgCron
## Version: 755VistaLab
## Machine Location: McAdams 123, Conf, Daniel 315, 304
## Release: 0.1
## Date of Release: 10.10.08
## Written By Dru Sepulveda (dsepulv@g.clemson.edu)

###################################
## Changes to condor_config
##
###################################

## For unknown reasons Condor is picking the loopback
## on Windows for the first time, so...

#NETWORK_INTERFACE = 130.127.*.*

###################################
## Backfill Config

## Turn on the Condor backfill option and set it to use
## BOINC as the backfill system.

#ENABLE_BACKFILL = TRUE
#BACKFILL_SYSTEM = BOINC

## Spawn a backfill job if the machine is unclaimed for
## more than X minutes where X=0 in this case.

#START_BACKFILL = $(StateTimer) > (0 * $(MINUTE))
```

```
## Evict a backfill job if the machine is busy (based on keyboard
## activity or cpu load) If this setting is not set Condor will
## complain in the StartLog, but this is not a major problem.
## Setting this variable may cause condor to stop BOINC jobs
## when a BOINC job ramps up the cpu.

#EVICT_BACKFILL = $(KeyboardBusy)

## Define a shared macro that can be used to define other settings.
## This directory must be manually created before attempting to run
## any backfill jobs.  This should be read/writable by all users.

#BOINC_HOME = D:\Data

## Point Condor to the BOINC executable and tell condor that BOINC
## jobs will be from the vanilla universe.

#BOINC_Executable = C:\condor\BOINC\boinc.exe
#BOINC_Universe = vanilla

## Set the project, clemson is a member of WCG, Einstein, LHC, SETI

#BOINC_Arguments = --dir $(BOINC_HOME) \
--attach_project http://www.worldcommunitygrid.org/ 1eae6b1ad44490d99490fbc0ef4f544d
#BOINC_Arguments = --dir $(BOINC_HOME) \
--attach_project http://einstein.phys.uwm.edu/ d1cdcffc16d2668a77f0adf6d78b2bfd
#BOINC_Arguments = --dir $(BOINC_HOME) \
--attach_project http://lhcathome.cern.ch/lhcathome/ 014b8ef0d4ce74f68fbac5613ce13d45
#BOINC_Arguments = --dir $(BOINC_HOME) \
--attach_project http://setiathome.berkeley.edu 5d084d86b97185c72474c75e5fbdee56

## Set the working directory for BOINC, this will be the directory
## where the error and out BOINC files will be placed, as well as the
## cert pack.

#BOINC_InitialDir = $(BOINC_HOME)

## These files must be created before hand and must be readable by
```

```
## all users

#BOINC_Output = $(BOINC_HOME)/boinc.out
#BOINC_Error = $(BOINC_HOME)/boinc.err

## What user the boinc client program should be run as. This
## macro is only used if the Condor daemons are running as root.
## In this case, the condor_starter must be told what user
## identity to switch to before spawning the boinc client. This
## can be any valid user on the local system, but it must have
## write permission in whatever directory is specified in
## BOINC_InitialDir).

#BOINC_Owner = m55????\condoruser
#STARTER_ALLOW_RUNAS_OWNER = TRUE

## Identify the type of machine with a custome machine ad

#MACHINE_TYPE = "csm55"
#STARTD_ATTRS = MACHINE_TYPE
```

Appendix C Python Code: Condor Dynamic Configuration

```
{

import httplib
import os
import time

#Dru Sepulveda
#DynConfig

try:
    CERTFILE = 'C:\condor\https_config\dyncondorclient.crt'
    KEYFILE = 'C:\condor\https_config\dyncondorclient.key'
    HOSTNAME = 'cirg.cs.clemson.edu:443'

    conn = httplib.HTTPSConnection(
HOSTNAME,
key_file = KEYFILE,
cert_file = CERTFILE
    )
    conn.putrequest('GET', '/dynconfig/condor_config.cs280')
    conn.endheaders()
    response = conn.getresponse()

    # HTTP Basic Authentication: first you have to do the initial connection
    # (above), then make a *second* GET request with the Authorization header
    # set to "Basic ARG"
    # where ARG is the base64 encoding of username:password (including the colon)
    conn.putrequest('GET', '/dynconfig/condor_config.cs280')
    conn.putheader('Authorization', 'Basic Y29uZG9yOmJyZWFrc3R1ZmYK')
    conn.endheaders()
    response = conn.getresponse()
    response = response.read()

    localConfig = open('C:/condor/condor_config.local.dyn', 'r')
    local = localConfig.read()
    localConfig.close()
```

```
    print int(cmp(response, local))

    if int(cmp(response, local)) != 0 :
        print "Files different, writing new config to condor_config.local.dyn"
        dynCondorConfig = open('C:/condor/condor_config.local.dyn', 'w')
        dynCondorConfig.write(response)
        dynCondorConfig.close()
        os.system('C:/Windows/System32/net.exe stop condor')
        time.sleep(10)
        #os.system('C:\condor\bin\condor_restart')
        os.system('C:/Windows/System32/net.exe start condor')

except:
   exit(1)

exit(0)


}
```

Appendix D Condor Router Log Parser (CROP) Code Sample

```
#! /usr/bin/env python

_name_='icrop.py'

import lex
import writeTable

# 1. Read Raw File into Data Structure
# 2. Find start and end time
# 3. Convert all dates into Times

def populateTable():
        i=0
        lineNo=0
        datetime=[]
        table=[]
        setx = []
        state=''
        states = ["Shadow", "exception", "submitted", "executing", "terminated.", \
         "aborted", "held.", "exception!", "disconnected", "reconnection", "failed", \
         "size", "suspended.", "unsuspended.", "evected.", "aborted", "Up", "Down", \
         "Globus", "Grid", "resource" ] # Globus, Grid, resource
        jobRouterList = []
        while lex.eof_state():
                lineNo=lineNo+1
                token = lex.passToken()
                if token == "...":
                        random = lex.passToken()
                        name = lex.passToken()
                        date = lex.passToken()
                        time = lex.passToken()
                        tempState = lex.passToken()
                        state = state + " " + str(tempState)
                        tempState = lex.passToken()
                        state = state + " " + str(tempState)
```

```
                tempState = lex.passToken()
                state = state + " " + str(tempState)
                if name != None and date != None and time != None: # and info != None:
                        table = table + [[name, date, time, state]]
        state =''
globals()['jobTable']=table

return table

print populateTable()
```

Appendix E Condor Router Log Parser (CROP) Screen Shot

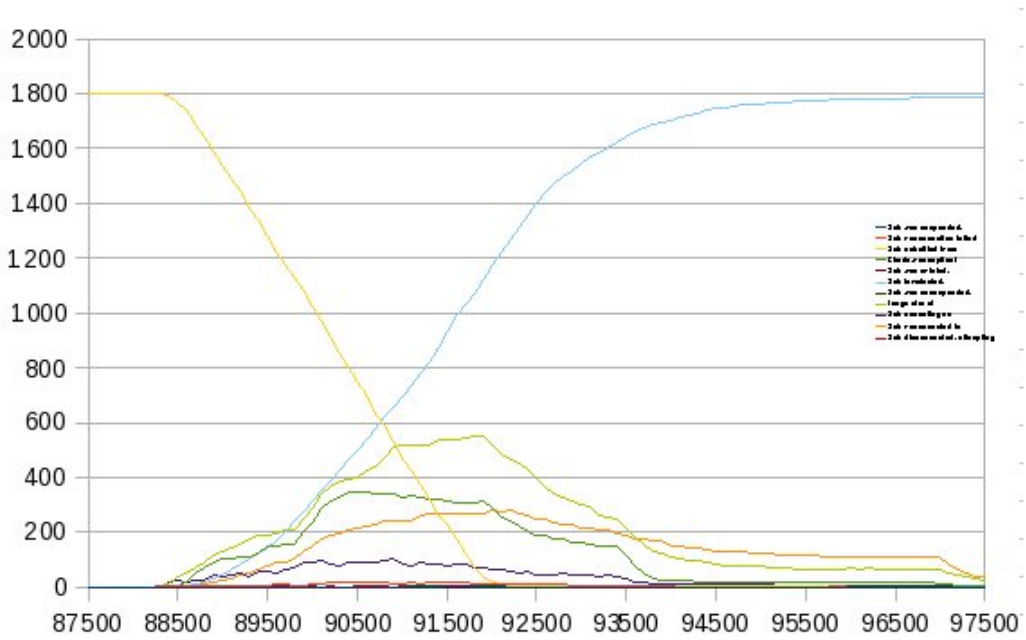


Figure 3: Example of a graph generated in Excel from data read by CROP

Bibliography

[1] The globus project home page. http://www.globus.org, January 2009.

[2] Gridrepublic's public website. http://www.gridrepublic.org/, January 2009.

[3] Miller's cyberinfrastructure labroratory. www.ces.buffalo.edu/faculty/miller/CI, March 2009.

[4] Open science grid home page. http://www.opensciencegrid.org, January 2009.

[5] O. La'adan A. Barak and A. Shiloh. Scaliable cluster computing with mosix and linux. *Proc. 5-th Annual Linux Expo*, pages 95–100, May 1999.

[6] Earle Ady and Christopher G. Stach II. Distributed.net home page. http://www.distributed.net/, March 2009.

[7] D. P. Anderson. Boinc: a system for public-resource computing and storage. In *Proc. Fifth IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 8 Nov. 2004.

[8] D.P. Anderson. Boinc: A system for public-resource computing and storage. *Proc. Fifth IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004.

[9] A. Barak and O. La'adan. "the mosix multicomputer operating system for high performance clister computing". *Journal of Future Generation Computer Systems*, 13:361–372, March 1998.

[10] J. R. Boisseau. Ut grid: a comprehensive campus cyberinfrastructure. In *Proc. 13th IEEE International Symposium on High performance Distributed Computing*, pages 274–275, 4–6 June 2004.

[11] et al. C. Hung. Integrating cagrid and teragrid. *Proc. 3rd Ann. TeraGrid Conf.*, 2008.

[12] Wolfgang Emmerich Matthew Farrellee Todd Tannenbaum Miron Livny Mark Calleja Clovis Chapman, Charaka Goonatilake and Martin Dove. Condor birdbath. *Unknown*, 2006.

[13] BOINC Community. The boinc manual wiki. http://boinc.berkeley.edu/wiki/User_manual, 2008.

[14] Teragrid Community. Teragrid home page. http://www.teragrid.org/, March 2009.

[15] E. Etzion and Y. Benhammou. Cinderella: Integration of classroom computers into the grid infrastructure. In *IEEE Nuclear Science Symposium Conference Record NSS '07*, volume 1, pages 918–920, Oct. 26 2007–Nov. 3 2007.

[16] G. Fedak, C. Germain, V. Neri, and F. Cappello. Xtremweb: a generic global computing system. In *Proc. First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 582–587, 15–18 May 2001.

[17] I. Foster. The anatomy of the grid: enabling scalable virtual organizations. In *Proc. First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 6–7, 15–18 May 2001.

[18] I. Foster. A new era in computing: Moving services onto grid. In *Proc. 4th International Symposium on Parallel and Distributed Computing ISPDC 2005*, pages 3–3, 04–06 July 2005.

[19] I. Foster and C. Kesselman. The globus project: a status report. In *Proc. Seventh Heterogeneous Computing Workshop (HCW 98)*, pages 4–18, 30 March 1998.

[20] Ian Foster. What is the grid? a three point checklist. Argon National Labratory and University of Chicago, July 20, 2002, July, 20 2002.

[21] Ian Foster. Ian foster's type pad page on cloud computing. http://ianfoster.typepad.com/blog/2008/01/theres-grid-in.html, March 2009.

[22] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Proc. Grid Computing Environments Workshop GCE '08*, pages 1–10, 12–16 Nov. 2008.

[23] Tanya Levshina Parag Mhashilkar Garzoglio, Gabriele and Steve Timm. Ress: A resource selection service for the open science grid. *Grid Computing International Symposium on Grid Computing (ISGC)*, 2007.

[24] D. L. Gonzalez, G. G. Gil, F. F. de Vega, and B. Segal. Centralized boinc resources manager for institutional networks. In *Proc. IEEE International Symposium on Parallel and Distributed Processing IPDPS 2008*, pages 1–8, 14–18 April 2008.

[25] S. L. Gooding, L. Arns, P. Smith, and J. Tillotson. Implementation of a distributed rendering environment for the teragrid. In *Proc. IEEE Challenges of Large Applications in Distributed Environments*, pages 13–21, 2006.

[26] A. Iosup, C. Dumitrescu, D. Epema, Hui Li, and L. Wolters. How are real grids used? the analysis of four grid traces and its implications. In *Proc. th IEEE/ACM International Conference on Grid Computing*, pages 262–269, 28–29 Sept. 2006.

[27] T. Jepsen. Soap cleans up interoperability problems on the web. *IT Professional*, 3(1):52–55, Jan.–Feb. 2001.

[28] Carl Kesselman and Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998.

[29] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky. Seti@home-massively distributed computing for seti. *Computing in Science & Engineering*, 3(1):78–83, Jan.–Feb. 2001.

[30] A. Kozakiewicz and A. Karbowskr. A two-level approach to building a campus grid. In *Proc. International Symposium on Parallel Computing in Electrical Engineering PAR ELEC 2006*, pages 121–126, 13–17 Sept. 2006.

[31] G. Lawton. Distributed net applications create virtual supercomputers. *Computer*, 33(6):16–20, June 2000.

[32] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor-a hunter of idle workstations. In *Proc. th International Conference on Distributed Computing Systems*, pages 104–111, 13–17 June 1988.

[33] Basney J. Raman R. Livny, M. and T. Tannenbaum. Mechanisms for high throughput computing. *SPEEDUP*, 11(1), 1997.

[34] M. Lorch, D. Kafura, I. Fisk, K. Keahey, G. Carcassi, T. Freeman, T. Peremutov, and A. S. Rana. Authorization and account management in the open science grid. In *Proc. 6th IEEE/ACM International Workshop on Grid Computing*, page 8pp., 13–14 Nov. 2005.

[35] P. Louridas. Soap and web services. 23(6):62–67, Nov.–Dec. 2006.

[36] Gregg Marland and Scott Marland. Should we store carbon in trees? *Water, Air, and Soil Pollution*, 64:18:l–195, 1992.

[37] et al. Michael Armburst. Above the clouds: A berkley view of cloud computing. *Electrocal Engineering and Computer Science University of Calfornia at Berkley Technical Report*, UCB/EES-2009-28:1–23, 2009.

[38] R. Miller, J. J. Bednasz, K. Chiu, S. M. Gallo, and M. Govindaraju. Grid-based research, development, and deployment in new york state. In *Proc. IEEE International Symposium on Parallel and Distributed Processing IPDPS 2008*, pages 1–8, 14–18 April 2008.

[39] C. Baumbauer P. Finnegan D. McKay M.Shuley, G. Veldman and S. Goasguen. Local community clusters experience with resource sharing in international and national grid infrastructure. http://www.cs.uiuc.edu/events/expwork2005/Sebastien_Goasguen_Abstract.pdf, 2005.

[40] BToonen N. Karonis and I. Foster. Mpich-g2: A grid-engbled implimentation of the message passing interface. *Journal of Parallel and Distributed Computing in Science & Engineering*, 63, No. 5:551–563, May 2003.

[41] News and Media. World community grid interviews the clemson school of computing's dr. sebastien goasguen. http://www.worldcommunitygrid.org/newsletter/08Q4/viewClemson.do, 2008.

[42] B. M. Oliver. Seti [search for extra-terrestrial intelligence]. 13(3):51–54, Aug–Sep 1994.

[43] Khoi Ahn Phan, Z. Tari, and P. Bertok. Similarity-based soap multicast protocol to reduce bandwith and latency in web services. *IEEE Transactions on Services Computing*, 1(2):88–103, April–June 2008.

[44] Suzan Polowczuk and Anne Heavey . No excuse for under-utilization: Clemson back-fills with boinc. http://www.isgtw.org/?pid=1001404, 2008.

[45] Ruth Pordes. Teragrid and open science grid. http://osg-docdb.opensciencegrid.org/0000/000027/002/GIG-OSG.pdf, 2004.

[46] Non profit Community. Carbon calculator. http://www.treefolks.org/prog_calculator.asp, 2008.

[47] D. A. Reed. Grids, the teragrid and beyond. *Computer*, 36:62–68, 2003.

[48] Igor Sfiligoi, Greg Quinn, Chris Green, and Greg Thain. Pilot job accounting and auditing in open science grid. *Proc. 9th IEEE/ACM International Conference on Grid Computing*, pages 112–117, 2008.

[49] Zheng Shijue, Shu Wanneng, and Chen Guangdong. A load balanced method based on campus grid. In *Proc. IEEE International Symposium on Communications and Information Technology ISCIT 2005*, volume 2, pages 1516–1519, 12–14 Oct. 2005.

[50] J. B. Sibarita, H. Magnin, and J. R. De Mey. Ultra-fast 4d microscopy and high throughput distributed deconvolution. In *Proc. IEEE International Symposium on Biomedical Imaging*, pages 769–772, 7–10 July 2002.

[51] P. M. Smith, T. J. Hacker, and C. X. Song. Implementing an industrial-strength academic cyberinfrastructure at purdue university. In *Proc. IEEE International Symposium on Parallel and Distributed Processing IPDPS 2008*, pages 1–7, 14–18 April 2008.

[52] The Condor Team. The condor manual version 6.8/7.0. http://www.cs.wisc.edu/Condor/manual/, 2008.

[53] D. Wallom, I. Stewart, and J. Wakelin. The university of bristol grid, a production campus grid. In *Proc. 14th IEEE International Symposium on HPDC-14 High Performance Distributed Computing*, pages 320–321, 24–27 July 2005.

[54] Jing Wang, Zhimin Yang, and Weili Kou. A solution for building campus grid. In *Proc. First IEEE International Symposium on Information Technologies and Applications in Education ISITAE '07*, pages 545–548, 23–25 Nov. 2007.

[55] N. Wilkins-Diehr, D. Gannon, G. Klimeck, S. Oster, and S. Pamidighantam. Teragrid science gateways and their impact on science. *Computer*, 41(11):32–41, Nov. 2008.

[56] Xingshe Zhou, Zhengxiong Hou, Yunlan Wang, Qiurang Liu, and Tao Wang. Campus computational grid oriented resource optimizing management. In *Proc. Fifth International Conference on Grid and Cooperative Computing Workshops GCCW '06*, pages 54–57, Oct. 2006.