4-20-2011

A Cost-Benefit Analysis of a Campus Computing Grid

Preston M. Smith
Purdue University, psmith@purdue.edu

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Preston M. Smith

Entitled A Cost/Benefit Analysis of a Campus Computing Grid

For the degree of Master of Science

Is approved by the final examining committee:

Jeffrey J. Evans, Chair

Carol X. Song

Kevin C. Dittman

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Jeffrey J. Evans

Approved by: Gary R. Bertoline, Head of the Graduate Program — 4/13/2011, Date

Graduate School Form 20
(Revised 9/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:
A Cost-Benefit Analysis of a Campus Computing Grid

For the degree of Master of Science

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Preston M. Smith
Printed Name and Signature of Candidate

4/13/2011
Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

A COST-BENEFIT ANALYSIS OF A CAMPUS COMPUTING GRID

A Thesis

Submitted to the Faculty

of

Purdue University

by

Preston M. Smith

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2011

Purdue University

West Lafayette, Indiana

Dedicated to my wife, Cheryl, whose support and constant encouragement both got me through my return to school, and always kept me on task.

ACKNOWLEDGMENTS

I would like to gratefully acknowledge the following individuals for their support and encouragement throughout my graduate journey.

Gerry McCartney, whose creation of the ITaP scholarship program helped me throughout the duration of my graduate studies, and to all of my management in ITaP who allowed me the freedom to pursue this degree.

John Campbell, Carol Song and Sebastien Goasguen who, at various times, either motivated me to start this program or provided advice as my studies progressed. Kevin Dittman for both teaching several of the interesting classes during my coursework and being willing to serve on my committee.

Miron Livny and the Condor team, who created the powerful Condor system that both keeps me employed and provided such an interesting topic of research.

Donna Cumberland, who provided guidance in analyzing the financial side of high-performance computing.

Collaborators Adam Carlyle and Stephen Harrell, who wrote with me the cost comparison of Community Clusters and Amazon EC2 - thanks for the methodology that made much of this thesis possible.

And finally, Professor Jeffrey Evans, who was always willing to provide guidance for me reaching the professional and academic goals that I set for myself. I'm very grateful for your interest in always doing what is in the best interest of the student.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Smith, Preston M. M.S., Purdue University, May 2011. A Cost-Benefit Analysis of a Campus Computing Grid . Major Professors: Jeffrey J. Evans, Kevin C. Dittman, and Carol X. Song.

Any major research institution has a substantial number of computer systems on its campus, often in the scale of tens of thousands. Given that a large amount of scientific computing is appropriate for execution in an opportunistic environment, a campus grid is an inexpensive way to build a powerful computational resource. What is missing, though, is a model for making an informed decision on the cost-effectives of a campus grid. In this thesis, the author describes a model for measuring the costs and benefits of building a campus computing resource based on the institution's existing investment in computing hardware.

For this study, the author calculates the usable capacity of a campus environment, and based on TCO data for Purdue University's campus grid, presents a model for calculating the base cost for a core-hour of computation in the campus grid.

A campus grid does introduce some extra expense, which is then amortized among the available core-hours to determine the real additional per core-hour cost to operate a campus grid. With this additional cost added to a baseline, an institution can learn the total cost of a core-hour in its campus grid.

With a cost model, the author then analyzes the benefits gained from using the grid, based on the number of hours of delivered, number of computations completed, and the number of users and faculty members served.

At Purdue, a core-hour costs just over $.03, all of which is an already sunk cost. The campus grid only costs 3.66 tenths of a cent more per core-hour to run, and an average of only $364.57 per user each year.

CHAPTER 1. INTRODUCTION

Research in the 21st century is increasingly dependent on computational cyberinfrastructure for execution of science. For a university to serve its research mission, it must have some commitment to providing a cyberinfrastructure for its faculty. Two primary options exist for doing so - investing in a traditional supercomputing center, or creating a campus grid built upon the university's existing resources.

Purdue University operates one of the largest cycle recovery systems in existence in academia, based on the Condor workload management system (Thain, Tannenbaum, & Livny, 2005), which is described in greater detail in Section 1.4.1.1. This system represents a valuable and useful cyberinfrastructure (CI) resource supporting research and education for campus and national users. Officials at Purdue routinely look to the campus computing grid as a solution to realizing power and cost savings (Tally, 2008).

What is missing from reports that connect campus computing grids to power and cost savings is a detailed analysis supporting what are the specific costs; including power, staffing, hardware, etc. and the benefits gained by an institution that implements a computing grid.

1.1 Statement of the Problem

A higher education institution needs a model for making an informed decision on how to provide cyberinfrastructure to support research. In this thesis, I will define a model for measuring the costs and benefits of building a campus computing resource based on the institution's existing investment in computing hardware.

1.2 Significance of the Problem

Gopu, Repasky, and McCaulay (2007) found that over a two year span (2004-2006) on the NSF TeraGrid (Catlett, 2005), 66% of all jobs executed on all systems were single-core jobs, and 80% of all of these serial jobs ran executed for two hours or less.

From November 1, 2008 through October 25, 2010, Purdue University users ran 35.4 million single-core serial jobs, using a total of nearly 40 million hours. These high-throughput computing jobs had an average runtime of 1.35 hours, and were 21% of all HPC hours consumed at Purdue. Examples of this single-core work include building a database of hypothetical zeolite structures via simulation (Chemical Engineering) (Earl & Deem, 2006), understanding the structure of viruses (Structural Biology), (Jiang, Wen et al., 2008) and simulating the production of comets (Astrophysics). (Kaib & Quinn, 2009)

The NSF Teregrid data and the representative data from an R1 institution known for scientific discovery demonstrates that a significant class of work that is suitable to run in an opportunistic campus grid.

1.2.1 Size of Resource Available to Campus Grids

Purdue University's 2009 Data Digest publication (Purdue University, 2009) reports that there are 27,000 workstation computers are on Purdue's campus. Assuming a low average of two processor cores per workstation, that suggests that there are at least 54,000 processor cores on Purdue's campus. In addition, machines providing 30,000 cores are operated by the central research computing facility.

Of the approximately 84,000 processor cores around campus, Purdue's Campus Grid currently makes 40,000 available to researchers. To compare a campus grid to a traditional HPC cluster and provide a sense of scale, there are only 17 systems on the June 2010 Top 500 (J. Dongarra, Meuer, & Strohmaier, 1998) list with 40,000 or more cores. The theoretical peak capacity of Purdue's campus grid is over 200

Teraflops, which would place the Purdue Campus Grid near the top 20 on the June 2010 Top 500 list.

Efforts to reclaim unused cycles only make sense if the amount of potentially reclaimable resources are significant.

Assuming a conservative estimate that these workstations are idle from 11:00 P.M. to 8:00 AM, this would provide 9 processor hours per core every night - for a total of 486,000 processor hours every night. Over the course of a year, this is 177.4 million processor hours that could be used for computation to support science.

Mutka (1988) measured the overall available time of a group of workstations at the University of Wisconsin, and found that a workstation is idle and available for computation 70% of the time. Observations at Purdue HPC clusters match: community cluster owners use approximately 70% of the available wall clock time.

Using Mutka's availability observations to make an estimate, a campus the size of Purdue University has 331.1 million hours of computation time sitting unused.

In the economic climate of 2011, and the budget crises that go along with it at institutions of all types, extracting the maximum value from an investment in information technology is more important than ever.

1.2.2 Power Costs

Regardless of what a computer system is sitting idle or fully computing, its power cost is significant.

An average dual-core system consumes 111 watts idle, and 160 watts fully loaded. (Schmid, 2010) At this power consumption Purdue's 27,000 idle workstations consume a total 2.99 MW/hour, for a total of 26,253.7 MW per year.

At full load, again using Mutka's 70% availability value, it is estimated:

27,000 processors x 24 hours x 365 days x .70 availability x 160 W = 26,490.24 MW/year

27,000 processors x 24 hours x 365 days x .30 availability x 110 W = 7,876.11 MW/year

For a net increase of 7,875.76 MW (29.7%) per year.

At Purdue's cost of $.05 per kW/hour, that is only an estimated incremental cost of $393,805.80 per year. Therefore, there is little additional power expense incurred for the additional use of the computer systems, relative to the total power expense for a campus' IT environment.

1.3 Statement of the Purpose

This work explores the costs and benefits with building a campus grid as a resource for an institution's computing needs. Specifically:

- The costs of simply operating these resources for their primary purpose
- The additional cost of using existing resources in the campus grid
- The costs of building a campus-sized high-performance computing resource, as a comparison to the costs of a campus grid
- The impact of a campus grid on the delivery of science, to demonstrate the benefits gained from the campus grid.

1.4 Definitions

For this study, the terms "campus grid", "desktop grid", and "volunteer computing" are all used interchangeably in the literature.

1.4.1 Campus Grids

Campus grids link computing resources within universities and research institutes, often including geographically distributed sites. A campus grid may be comprised of dedicated computing resources, idle non-dedicated computing resources such as workstations or student labs.

Most importantly, campus grids build a shared computational resources out of an institutions existing investment in computer resources. A campus grid can be built a number of different ways: combining and sharing individual computer systems and community clusters (Purdue), combining and sharing homogeneous, distributed-owned collections of machines (GLOW), or combining and sharing individual clusters with middleware (FermiGrid).

Regardless of the architecture of a campus grid, the key characteristic is that the campus grid facilitates sharing of computing resources.

1.4.1.1. The Condor System

Condor (Thain et al., 2005) is a batch computing system developed at the University of Wisconsin, Madison, that provides job and resource management functions coupled with mechanisms to create and manage scheduling policies. One of the critical philosophical approaches that distinguish Condor from other scheduling systems is the concept of *high-throughput computing*, which emphasizes the delivery of "large amounts of processing capacity over very long periods of time" (Livny, Basney, Raman, & Tannenbaum, 1997). In contrast, the traditional approach in the high performance computing community has been to focus on metrics such as TeraFlops (TF), which cannot fully represent the end goal of operating a computing resource: enabling the completion of science.

Condor is designed to operate opportunistically to harvest unused computational resources wherever and whenever they are available, be it from a user's desktop computer or from a high-performance server in a data center.

1.4.2 Classifications of Campus Grids

I will describe three major campus grids in use today, each with different characteristics. Purdue, GLOW at the University of Wisconsin, and FermiGrid.

1.4.2.1. Purdue

Purdue University operates a large-scale campus computing grid distributed between its main West Lafayette campus, Calumet, Fort Wayne, and North Central campuses, the University of Notre Dame, and other institutions. (Purdue University, 2010b). The base of Purdue's campus grid is the *community clusters* (Shuey et al., 2005) program. A *community cluster* is a system purchased by a faculty group and centrally operated by an institution, maintained for the benefit of the many research groups that own the nodes in the cluster as well as the broader campus community. Community cluster customers gain peace of mind from the cluster's operation by professional IT staff; low overhead from centralized power, cooling, and data center space; and cost effectiveness from the combined purchasing power of all cluster owners and strategic sourcing of the cluster hardware. (Purdue University, 2010a)

As of 2010, Purdue operates two large community clusters in production: Steele, a 902-node (7216 core) Dell Xeon E5410 "Harpertown" cluster interconnected with Gigabit Ethernet; and Coates, a 993-node (7944 core) HP AMD Opteron 2380 "Shanghai" cluster interconnected with 10 Gigabit Ethernet.

Using Condor, the Purdue campus grid connects idle processors on the community cluster nodes, student lab machines, and individuals' desktops to provide a single general purpose computing resource to the campus.

1.4.2.2. GLOW at the University of Wisconsin, Madison

According to the University of Wisconsin, The Grid Laboratory of Wisconsin (GLOW) is a "campus-wide distributed computing environment designed to meet the scientific computing needs of the University of Wisconsin-Madison. It is built from autonomous sites from across the campus, each engineered to meet their own specific requirements, but cooperating to join with the other sites to serve the ever-growing computing needs of the entire campus." (University of Wisconsin, Madison, 2010) Like Purdue, GLOW is built with Condor.

1.4.2.3. FermiGrid

Like Purdue and GLOW, the campus of Fermi National Accelerator laboratory operates a grid sharing resources owned by different offices and experiments on its campus. FermiGrid (Fermi National Accelerator Lab, 2010) allows Fermilab experiments to get opportunistic access to resources owned by others, optimize the use of Fermilab resources, and provide a coherent means to connect Fermilab to the Open Science Grid. (Pordes et al., 2007) FermiGrid connects multple Condor clusters and PBS clusters, (Altair Corporation, 2010) with custom middleware and the Globus (Foster & Kesselman, 1997) toolkit.

1.5 Assumptions, Delimitations, and Limitations

There are some limitations on the scope of study for this thesis, described below:

1.5.1 Delimitations

This study will use only the West Lafayette Campus of Purdue University as an data source for the costs and benefits of implementing a campus computing grid.

Additionally, this study will only explore the costs and benefits of campus grid computation resources. The following topics will be outside the scope of this study.

- Costs and benefits of storage solutions on the campus grid
- Costs and benefits of a traditional supercomputing system. (For example, a cluster)

CHAPTER 2. REVIEW OF THE LITERATURE

2.1 Costs

Opitz, König, and Szamlewska (2008) assert that assumptions that grid computing does not incur any additional costs (Silverstein, 2005) are false. Opitz describes a detailed model that considers staffing costs, costs of important hardware components, network connectivity, mean time to failure of components, and power, based on experience with a corporate grid at the Novartis corporation.

This study is useful in that it outlines several items (power, staff, hardware), that need to be taken into account in any cost study. However, it does go into great detail on costing some items that do not apply to Purdue, such as grid software, external network connections, and the cost over a lifetime of a data center.

Microsoft Research's Gray (2008) described the economics of grid computing with a simple model, described in Table 2.1:

Table 2.1.

Grid Computing Costs, from Microsoft Research

Cost	Task
$1	1 GB WAN transfer
$1	8 hours CPU time
$1	1 GB disk space
$1	10M database accesses
$1	10 TB LAN bandwidth
$1	10 TB disk bandwidth

This model would suggest that an hour of CPU time is worth $.125.

The costs of cloud computing have been compared with Amazon's Elastic Compute Cloud (EC2) (Amazon, Inc, 2008) with volunteer computing grids. (Kondo, Javadi, Malecot, Cappello, & Anderson, 2009). This serves as an inspiration for this work to use EC2 as a frame of reference for computing costs.

Rather than comparing to a volunteer grid, Afgan and Bangalore (2007) compares the cost of using Sun Grid (now Sun Open Source Cloud) (Sun Microsystems, 2009) against a rough estimate of building and owning a dedicated cluster. This is the second of three sources that use a cloud computing offering as a cost comparison. Similar to Opitz et al., Afgan and Bangalore itemizes the total 5 year cost of owning a cluster, but the model is much more simplistic than what Opitz described.

Walker (2009) describes a model using net-present value (NPV) calculations to compute the real cost of a CPU hour and compare the cost of purchasing a computer system vs leasing. Walker calculates the per-cpu hour cost of the Ranger supercomputer at the Texas Advanced Computing Center as an example. Ranger's per-CPU hour cost is $0.045 cents per CPU hour.

Beck, Schwind, and Hinz (2008), describes the economics of a computing grid by comparing the provision and use of compute cycles to a market, complete with pricing to give a cost to allocation of cycles to users. A market is only useful, though, if there is more demand than supply. Purdue's campus grid currently has excess capacity, driving the economics of the grid down to zero.

Carlyle, Harrell, and Smith (2010) created an "EC2 equivalence" metric to compare the per node-hour costs of Amazon EC2 Cluster Compute Instances to Purdue University's Community Cluster program. This metric was developed to describe the per-share costs to a participant in the Community Cluster program, and can be adapted to describe the cost to an institution instead of a shareholder.

2.2 Benefits

To create a cost-benefit analysis, we must also be able to categorize the benefits of having a computing grid.

2.2.1 Measuring the Capacity of the Grid

Mutka (1988) measured the capacity of a department's workstations over a period of 5 months, and determined that a system is available for computation 70% of the time. This is one of the first works measuring the capacity of a volunteer computing environment.

Thain, Tannenbaum, and Livny (2006) measured the sizes, locations, OS and platform makeup, number of pools, software versions in use, and downloads for installations of the Condor software worldwide.

Anderson and Fedak (2006) described in the potential computational and storage capacity of a volunteer grid, using data from BOINC in 2006.

Toth and Finkel (2007) found that public computers used in a business are available to volunteer computing 73% of the time, while home and business desktops are available 27% of the time, and student desktops are available only 18% of the time.

Brevik, Nurmi, and Wolski measured traces from Condor at the University of Wisconsin, Madison, and the University of California, Santa Barbara and the University of California, Santa Cruz, with the goal of "a quantile estimation methodology that supports live predictions of availability so that schedulers (be they human or automatic scheduling programs) can make decisions dynamically." (Brevik et al., 2004). This study's models, when applied to a Condor grid, produced unusable availability estimates.

Nurmi, Brevik, and Wolski developed statistical models of desktop grids in general (2005), Studying a Condor pool in particular, Wolski, Nurmi, and Brevik (2007) found that no single statistical model is best for measuring availability in an

opportunistic desktop grid.

Finally, Acharya, Edjlali, and Saltz found that on average, 60-80% of workstations in an environment are available for computation, and that a substantialnumber (20-70%) of the systems are always available, and evaluated the utility of a volunteer computing environment for parallel computations. (Acharya et al., 1997) This method is attractive in that for one of the three traces analyzed, it simply uses Condor itself to report the usable capacity.

2.2.2 Measuring the Utility of the Grid

Kriebel and Raviv (1980) describes a framework for supplying computing services based on the demand in the "market". This framework can be used by management for analysis and planning. Kriebel and Raviv defined this work in the context of business IT in a time far before the grid, but notes even in 1980 that this model could be used in university computing centers.

Sassone (1988) surveyed methodologies for analyzing costs and benefits of information systems. Most of the methodologies discussed are more appropriate for analyzing whether or not to replace manual work with technology, but the "Cost Effectiveness Analysis" used to compare several potential choices is a general methodology applicable to this study.

Acharya et al. does some estimation of what benefits a user can expect from a workstation grid, but specifically focuses on parallel jobs. It uses the NAS Parallel Benchmarks (NPB) (Bailey et al., 1994) to benchmark the benefits of the grid.

Snir and Bader (2004) defines the productivity of a supercomputer as the:

> ... ratio between the utility of the output it computes to the cost of the system. This measure depends on the preferences of an agent, rather than being objectively defined. This is unavoidable: the value of the output of a supercomputer may vary from organization to organization; as long as there is no practical way to trade super- computer outputs, there is no way

> to directly compare the worth of a supercomputer for one organization to the worth of a supercomputer for another organization. (p. 418)

This could perhaps be summarized as "productivity is in the eye of the beholder". Such a vague definition of a supercomputer's utility illustrates the difficulty of defining it.

Sterling (2004) defines an HPC productivity model that takes into account useful work performed, divided by time, and then including program size and programmer effort. Other works (Kennedy, Koelbel, & Schreiber, 2004) focus much more on the productivity of the software and languages used in high-performance computing.

Kuck, (2004) discusses factors of HPC productivity, from architecture choice through software development. Kuck wisely describes a measure of *solutions* per year, rather than runs, but notes that such a measure is subjective and difficult to measure.

Streeter (1972) notes that with a scientific computing service, measures should be flexible and consider human costs and benefits, due to the wide variety of research projects making it difficult to define the "business" of research.

Kondo, Taufer, Brooks, Casanova, and Chien (2004) described a "cluster equivalence" metric, that compares the utility of a volunteer computing grid to a cluster. Additionally, the authors measured the length of intervals that a desktop grid system is available for computations, and the rate of task failure as a function of task size.

In table 2.2, Purdue University reports that its campus grid has delivered the following amount of time to science (Smith, Hacker, & Song, 2008), gathered from Purdue's usage metrics:

Such metrics are probably the most easily determined measures of productivity of the campus grid, from the institution's perspective.

Table 2.2.

Condor usage summary by year

Year	Pool size	Jobs	Hours Delivered
2004	1500	43,551	346,000
2005	4000	295,265	1.9 million
2006	6100	4.44 million	4.61 million
2007	7700	9.93 million	8.17 million
2008	22,000	14.9 million	16.6 million
2009	30,000	15.4 million	17.9 million
2010	42,000	15.2 million	18.6 million

2.2.3 Literature Review Summary

This literature review reveals many works studying the costs of high-performance computing as well as measuring the usable capacity of a volunteer computing system.

Opitz et al. (2008) suggests many things to consider when evaluating the cost of a computing grid, and Walker (2009) discusses the per core-hour costs of the Ranger supercomputer that we will find to be similar to resources at Purdue. Carlyle et al. presents a useful method for calculating a per node-hour or core-hour cost for an HPC resource, and relating it to the known cost of Amazon EC2.

The various studies about the capacity of a computing grid all use a methodology which simply queries the system (Condor or BOINC) and regularly samples the number of systems available for use.

In the area of describing the utility, or productivity of a computing grid, however, the literature is nowhere as complete. Some studies involve measuring the productivity of a business information system rather than a high-performance computing system, and others are nearly 25 years old.

Some studies (Kuck, 2004) note that solutions reached is an excellent metric with which to measure the output of a computing system, but notes that is a difficult

thing to measure.

As suggested by Snir and Bader, - "productivity is in the eye of the beholder" - I will define a set of quantifiable outputs with which to measure the utility of the Purdue campus grid, from the perspective of my particular institution. The specific metrics with which I will measure the grid's productivity are discussed in Section 3.1.

CHAPTER 3. PROCEDURES AND DATA COLLECTION

3.1 Methodology

This study included considerations described by Opitz et al., combined with values drawn Purdue's own IT cost models. This allowed the author estimate the costs to operate systems in the campus grid at Purdue, and identify the additional costs for the systems to participate in the campus grid.

With the costs of the grid identified, the next step was be to measure the capacity of the Purdue campus grid using methodologies like described by Toth and Finkel (2007) and Mutka (1988). For a two-week period of time, the Purdue Condor pool was sampled every 10 minutes, with each sample listing the state of a machine (Idle, Computing, Owner-used), and whether a previously-seen machine is online. No special instrumentation was be required to gather these traces, all of this information is available by querying the Condor system directly. This sampling simply queried Condor's own central accounting database (the *condor_collector*) currently used for usage reporting, and did not impact the production Condor pool. These scripts are included in the Appendix.

Each sample of data is approximately 79 bytes - generating only 5.6 KB of trace data each day.

With trace data collected, I created a system in Amazon EC2, installed Condor on it, and compared Condor's own internal benchmark metrics on that node (Kflops and Mips) to the average values in the Purdue Condor pool. The kflops internal metric is a single-CPU Linpack (J. J. Dongarra, Luszczek, & Petitet, 2003) benchmark, and Mips is created by running an implementation of Dhrystone (Weicker, 1984).

Then, to demonstrate the relative performance of a Condor node, I ran a CPU-bound, loosely-coupled benchmark on a sample of many nodes of Purdue's Condor Pool, one node of each of Purdue's community clusters, and an Amazon EC2 node, to obtain a factor with which to compare relative performance.

Using an adaptation of the equivalence metric developed in Carlyle et al., I then compared the costs of an hour of computation in the Purdue campus grid to both Purdue's Community Cluster program and to the Amazon EC2 cloud computing environment. The model described by Carlyle et al. measured the cost to each customer rather than the institution, so in this study, I will refine it to present the cost to the institution.

During development of the campus grid at Purdue, our driving forces were threefold:

- Provide computational time to facilitate research
- Serve new communities of users underserved by HPC
- Provide a solution with low time-to-science

Therefore, to measure the impact of the grid, I measured the following outcomes:

- Hours of computation delivered
- Number of computations completed
- Number of unique users served
- Number of disciplines served
- Mean time to result of computation

CHAPTER 4. PRESENTATION OF DATA AND FINDINGS

4.1 Presentation of the Data

After collecting all the data as described in Section 3.1, this chapter now presents the results of that methodology.

4.1.1 Grid Capacity

Over a 12-day period of time from Jan 22 through Feb 3, 2011, regular queries were sent to the Purdue Condor collector to establish the overall usable capacity of the Purdue Campus grid. In this case, a slot is "available" if it is not in use by its "owner". Examples of owners are students using a computer lab, a desktop user, or a cluster node in use by PBS. Both the overall available capacity, and the available capacity of desktop resources in particular were queried.

The averages of both the number of slots, and the percent of the slots available for user jobs calculated from the data collected is presented in Table 4.1, along with the number of slots that that percentage represents.

Table 4.1.

Purdue Campus Grid Usable Capacity

	Number of Slots	Percent Available	Slots available
Entire Grid	32,406	41.74%	13,526
Desktop Resources	4,509	90.13%	4,064

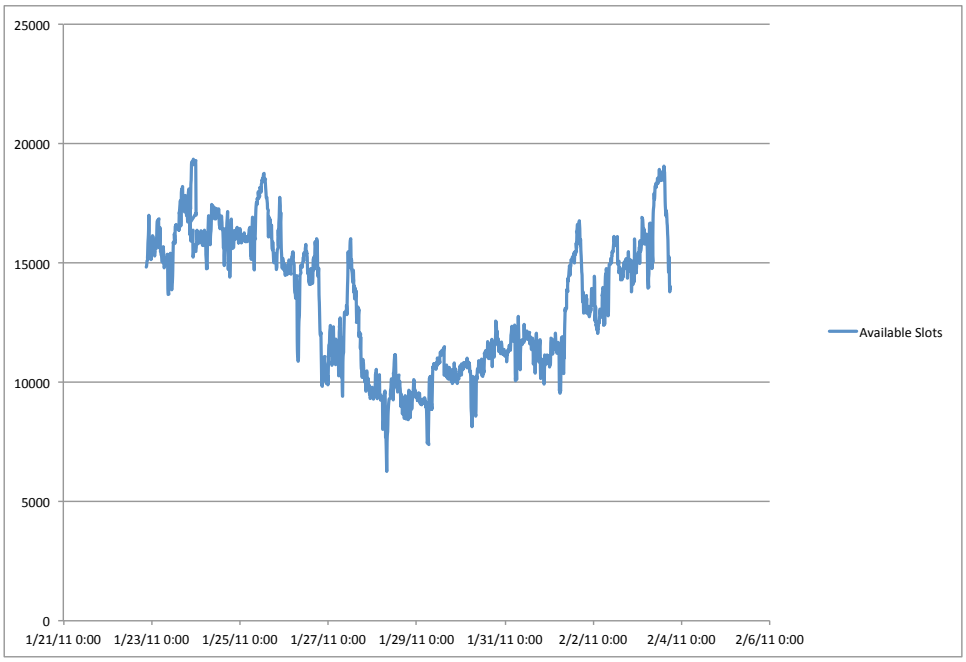

Figure 4.1. Usable Capacity of Purdue Grid Over Time

Figure 4.1 illustrates the usable capacity over the 12 days sampled. This table shows that, on average, 13,526 CPU cores are available to run user jobs. Over the course of a year, this totals 118,487,760 CPU hours.

4.1.2 Grid Performance

There are several potential metrics that may be analyzed to understand the performance of nodes in the campus grid.

4.1.2.1. Condor's Internal Benchmarks

By querying the central *condor_collector* daemon in the Purdue Condor pool (32579 CPU slots) I recorded the results of Condor's internal benchmarks for per-core performance: Kflops (Single-CPU LINPACK), and Mips (Dhrystone).

Table 4.2 reports the Kflops and Mips measurements for cores in Purdue's Community Clusters "Steele" (Dell 1950, Quad-Core Intel E5410); "Coates" (HP ProLiant DL165G5, Quad-Core AMD 2380); and "Rossmann" (HP ProLiant DL165G7, 12-core AMD Opteron 6172); the entire Purdue campus grid, and nodes from Amazon EC2.

Table 4.2.

Purdue Campus Grid Internal Core Benchmark Results

Cluster	Clock Speed		KFlops	Mips
Steele	2.33 GHz	Mean	1,547,375.79	7241.24
		Stddev	90,306.41	432.82
Coates	2.5 GHz	Mean	1,593,767.57	5708.17
		Stddev	30,949.49	337.34
Rossmann	2.1 GHz	Mean	1,338,529.38	4854.77
		Stddev	16,703.51	1,024.15
Condor Pool Avg	2.31 GHz	Mean	1,483,608.69	5833.40
		Stddev	232,144.96	1,156.13
Amazon EC2	2.66 GHz	Mean	1,256,329.7	8512.3
		Stddev	110,666.82	2,378.65

Figures 4.2 and 4.3 compare the mean Mips and Kflops measurements of cores from the Campus Grid, Purdue's Community Clusters, and Amazon EC2.

4.1.2.2. Real-world Benchmarking

Without further study, It is unclear if the LINPACK and Dhrystone numbers presented by Condor are useful as a benchmark with which to measure the relationship between the performance of a core in the campus grid and an core in an Amazon EC2

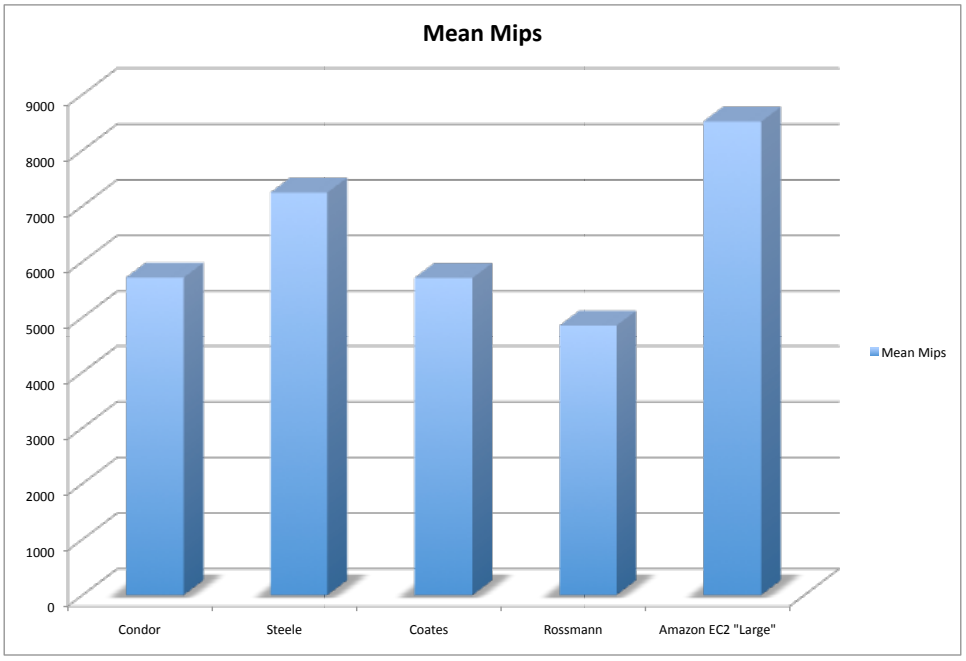

Figure 4.2. Mean Mips measurements

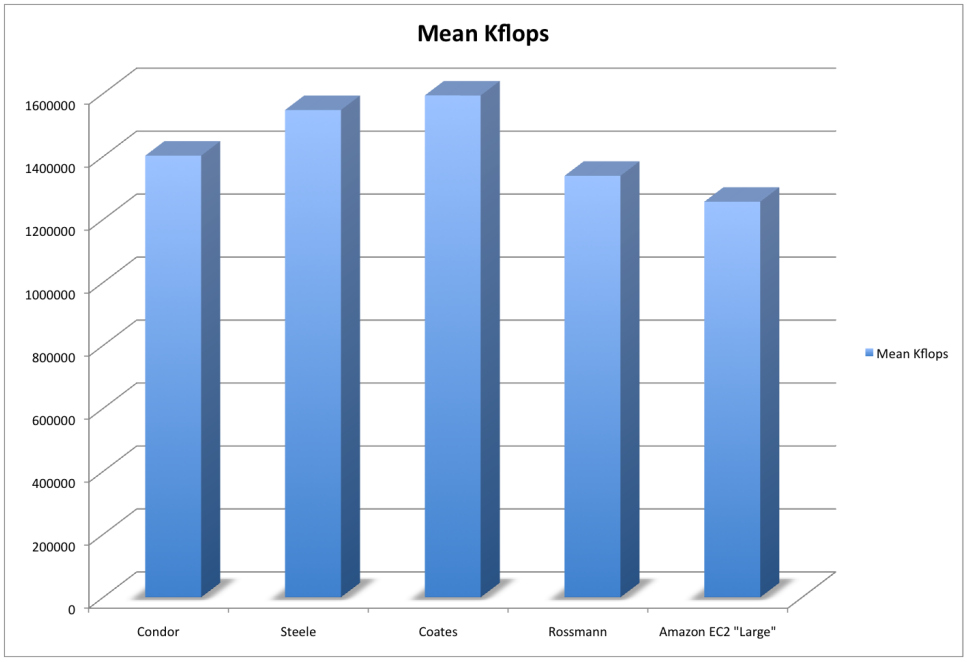

Figure 4.3. Mean Kflops measurements

node. Furthermore, other studies have documented weaknesses of Dhrystone (York, 2002) and it is not clear if LINPACK results have any relationship to the performance of a loosely-coupled campus grid computation.

Therefore, as did Carlyle et al. and Acharya et al., I have used a subset of the NAS Parallel Benchmarks (NPB) (Bailey et al., 1994) as a benchmark with which to normalize the performance of each class of resources. NPB is useful as it is a real-world type of application which "mimics the computation and data movement characteristics of large scale computational fluid dynamics applications". NPB is available in parallel versions suitable for large supercomputers, or a serial (single-CPU) version which lends itself well to execution in a campus grid. The serial NPB is what is run here.

Of the eleven potential benchmarks in NPB, I selected the "BT" (Block Tridiagonal) and "SP" (Scalar Pentadiagonal) benchmarks, both of which solve a system of nonlinear partial differential equations. These benchmarks are the same ones run by Carlyle et al. - selected because they solve nonlinear partial differential equation - a very common operation in many types of HPC codes. The BT benchmark has the added benefit of including an I/O intensive subtype. NPB benchmarks are available in a number of problem sizes ("classes") - class C is what I ran for this experiment - solving a 3D problem matrix of the size 162x162x162.

Using the Condor system, I ran 100 copies of each benchmark on campus grid nodes, and established an average performance for each benchmark. By comparing this result to the performance of the same benchmark on an Amazon EC2 "Large" instance, a factor can be obtained with which to compare the performance of campus grid cores to one on EC2. Table 4.3 and Figure 4.4 summarize the mean execution times of the NPB benchmarks.

These benchmarks show that, on average, the NPB benchmarks run at 17.1% faster on Amazon EC2 than they do on the campus grid. Therefore, an average campus grid core is 82.9% of the speed of a core on Amazon EC2.

Table 4.3.

NAS NPB Benchmark - Mean Execution Time (seconds)

	BT	SP
Campus Grid	2307.41	2310.36
Amazon EC2 "Large"	2025.74	1803.38
Difference	12.2%	21.9%
Average Speedup		17.1%
Scaling Factor		.829

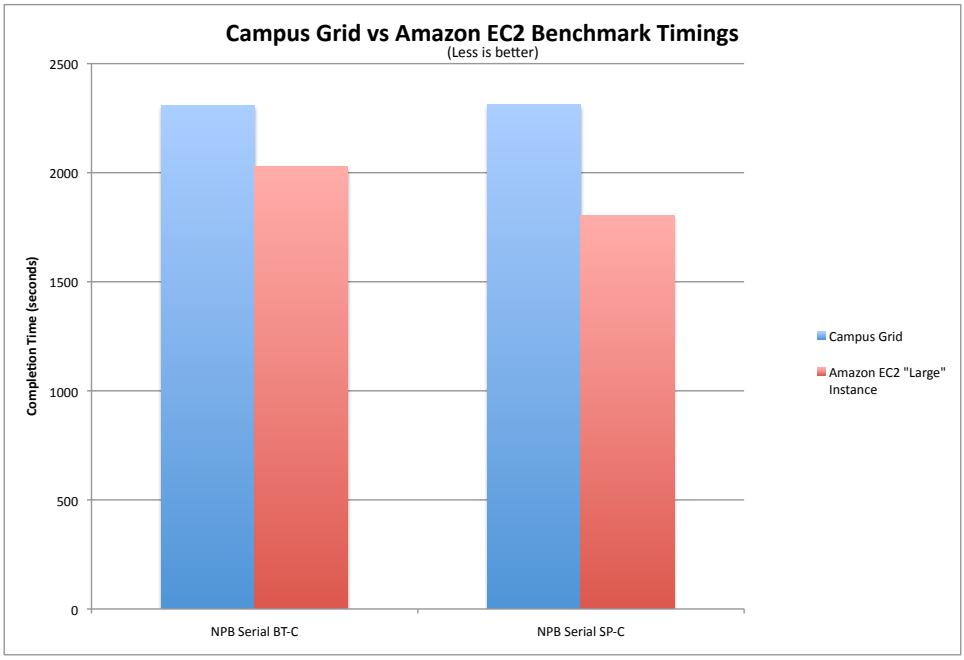

Figure 4.4. NPB Performance on Purdue Campus Grid and Amazon EC2

Figure 4.5 depicts the relationship between Condor's Kflops and MIPS metrics, and the results of the NPB benchmark runs described in Table 4.3 and Figure 4.4.

Condor's own Kflops metric shows that on average, Amazon EC2 nodes are 15.3% faster than nodes on the campus grid, compared to the 17.1% difference measured by the NPB runs. According to Table 4.2, the mean CPU speed of cores in the

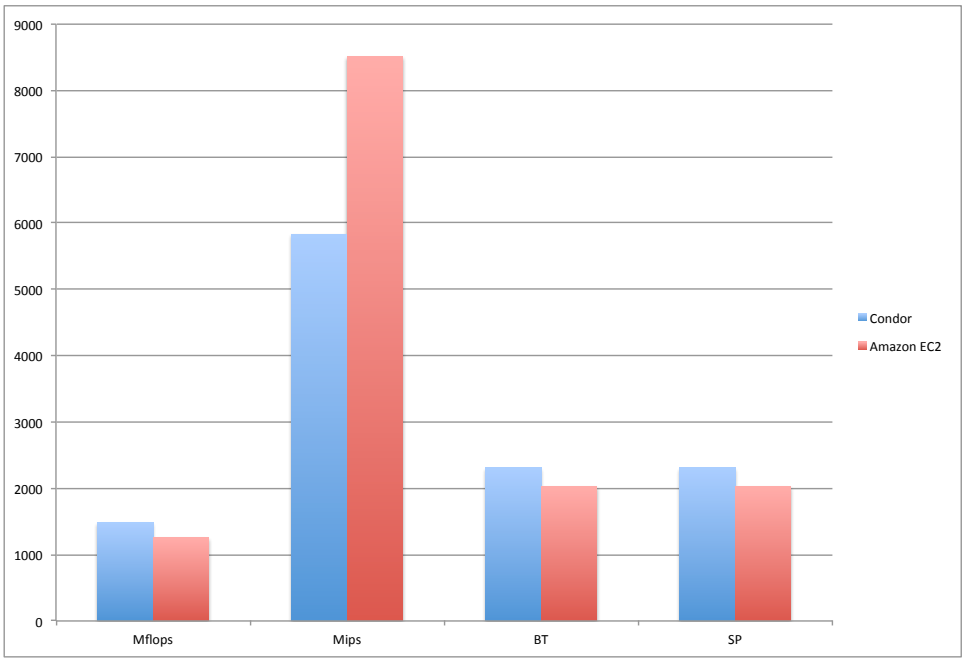

Figure 4.5. Relationship Between Condor Internal Benchmarks vs NPB Results

campus grid is 2.31 GHz, and 2.66 GHz for Amazon EC2 instances - a difference of 13.2%. This suggests a correlation between CPU clock speed, Kflops numbers, and NPB run time. Future work should further study this relationship.

This suggests that Condor's Kflops metric is potentially useful as an indicator of relative node performance for common CPU-intensive campus grid codes. The MIPS benchmark shows no obvious relation to the performance of NPB.

With this result from gathered from a real-world application, there is now a conversion factor (.829) with which to normalize the per-core cost of the campus grid and make an apples-to-apples comparison to Amazon EC2, or to community cluster results obtained by Carlyle et al.

4.1.3 Cost Comparisons

4.1.3.1. Description of the Model

Carlyle et al. described their EC2 equivalent cost methodology as follows:

$$\left(\frac{\text{\# months}}{\text{\# core-hrs}}\right)\left(\frac{\text{\# core}}{\text{1 node}}\right)\left(\frac{\text{1 hour (Cluster)}}{\text{hour (EC2)}}\right)\left(\frac{\text{Month cost}}{\text{1 month}}\right) = \text{Cost per node-hour} \quad (4.1)$$

Since all metrics and cost results are for a single core (not per-machine), and the fact that Condor represents an n-core node as quantity n single-core slots, I will treat each Condor "core" as a "node", (1:1) removing the factor of cores per node. Using 1 month in the first and last factors cancels out the "month" term. With these adjustments, a simplified model to determine the normalized cost per "node" (core) hour is presented in Equation 4.2:

$$C_n = \frac{C_h}{F_n}, \quad (4.2)$$

where C_h is the pre-normalized per core-hour cost of a core on the campus grid, and F_n is a constant representing the normalizing factor of one hour on the grid to 1 hour on EC2. The result, C_n, is the normalized per-core-hour cost.

4.1.3.2. Results of Model with TCO data from Purdue Campus Grid

Using the same cluster total cost of ownership (TCO) data used by Carlyle et al., I calculated the hourly per-core cost of the Purdue Community clusters. (D. Cumberland, personal communication, January 29, 2011) This data, along with TCO data for the other major component of the campus grid: Purdue's student instructional

labs (L. Theademan, personal communication, December, 2010), enabled calculation of the mean hourly per-core cost of machines in the Purdue campus grid. Data for the Rossmann cluster was not available for analysis, but due to its similarity to Coates it is expected that its TCO will be very similar to Coates. Pre-normalized costs are summarized in table 4.4, along with the cost of running a core-hour in Amazon EC2.

Table 4.4.

Summary of Per Core-Hour Cost of Computation Resources

Resource	Pre-Normalized Cost
Coates	.0237
Steele	.0218
Instructional Labs	.0445
Average Purdue Condor Cost	.0300
Amazon EC2 (Large)	.17

This table is the cost *to Purdue University* - the institution itself - to build a computation system on each of these resources. It should be noted that the dramatically higher cost of an Amazon EC2 core is due to the fact that this is a retail price, not the amount that it actually costs Amazon to run the infrastructure. The Amazon cost includes many unknown factors and an almost certainly substantial mark-up to make the service profitable.

Using the average per node-hour cost of the campus grid of $.03, obtained in Table 4.4, and a normalizing factor of 0.829 obtained from Table 4.3, these values are inserted into into the new equivalence formula previously described as Equation 4.2.

$$\begin{aligned} C_n &= \frac{C_h}{F_n} \\ &= \frac{.0300}{.829} \\ &= \$0.03619, \end{aligned}$$

where C_h is \$.0300 (from Table 4.4), and F_n is 0.829 (from Table 4.3).

Therefore, C_n - the normalized per-core-hour cost of an hour on the Purdue campus grid - is \$0.03619.

All of this expense is already absorbed by the University in the course of operating its Community Clusters and student computing labs.

4.1.3.3. Additional Expense to Operate Campus Grid

Missing from the per node-hour expense is additional staff and hardware not already used to operate the infrastructures comprising the campus grid. Table 4.5 summarizes all the additional expenses necessary to operate the Purdue campus grid.

Table 4.5.

Details of Additional Expense to Operate Campus Grid

Item	Total Yearly Cost
Systems Engineering (1 FTE)	\$73,810.00
Advanced User Support (.75 FTE)	\$55,357.50
Distributed Administrators (.1 FTE)	\$11,071.50
Additional Power Expense	\$290,295.01
Item	**Total Cost, Amortized over 5 years**
Dedicated Submit Nodes (3)	\$6360.00
Checkpoint Servers (4)	\$8480.00
Actual Yearly Expenses	\$433,502.01

These expenses include staffing for systems engineering necessary to maintain the additional servers that drive the campus grid, as well as the expense of those servers; advanced user support staff responsible for supporting researchers using the resource; staff expenses incurred by additional IT staff around campus participating

in the grid; and the additional cost of running the available workstations at full load instead of at idle. See Section 4.1.3.3.1 for full discussion of the additional cost of power.

The additional expense of operating the campus grid can be represented by Equation 4.3.

$$C_a = \frac{E_y * S_a}{H_y}, \tag{4.3}$$

where E_y is the total expenses of operating the grid per year, as calculated in Table 4.5, and S_a is the available slots calculated in Section 4.1.1. H_y is a constant representing the number of hours in a year. The result, C_a, is the additional per core-hour expense of operating the campus grid.

Solving Equation 4.3 calculates C_a:

$$\begin{aligned} C_a &= \frac{E_y * S_a}{H_y} \\ &= \frac{\$433,502.01 * 13,526}{8760} \\ &= \$0.003658623 \text{ (3.66 tenths of a cent).} \end{aligned}$$

C_a, the additional per core-hour expense of operating the campus grid, is \$0.003658623 (3.66 tenths of a cent).

4.1.3.3.1. Power Measurement

One substantial component of the additional expenses is the increased power cost of running machines at full power instead of idle. As discussed in the literature review, Schmid (2010) reported that an average desktop system consumes 111 watts idle, and 160 watts under full load, a ratio of 0.6938. A Purdue measurement of power consumption of a variety of 13 systems that can be found in the Campus Grid finds that the average ratio of idle to fully loaded wattage is 0.6603, which is

within one standard deviation (.0771) of Schmid's estimate. (P. Finnegan, personal communication, March 24, 2011)

The estimated additional power expense incurred based on Schmid's ratio of 0.6938 and a per kW-hour cost of $.05, is $290,295.01, depicted in Figure 4.6.

4.1.3.4. Total Cost

Equation 4.4 describes the computation of the total cost of a core hour.

$$C_t = C_n + C_a \tag{4.4}$$

Using the values for C_a and C_n from Section 4.1.3.2, I calculate C_t:

$$= \$0.03619 + \$0.003658623$$
$$= \$0.0398468012$$

The total cost per core hour delivered of a campus grid node is $0.0398468012. This value will be referred to as C_t.

4.1.4 Scientific Output

The following metrics in Table 4.6 were provided by the Purdue Rosen Center for Advanced Computing (G. Flint, personal communication, March 2, 2011), measuring the output enabled by the campus grid.

The mean time to result (that is, the time elapsed between submission and completion of the job) of all jobs ran in the campus grid from 2005-2010 is 13,986.6 seconds - that is, 3.88 hours. Table 4.7 summarizes the mean per-job run time, and the mean time to result for jobs in the Purdue Campus Grid. (G. Flint, personal communication, March 24, 2011)

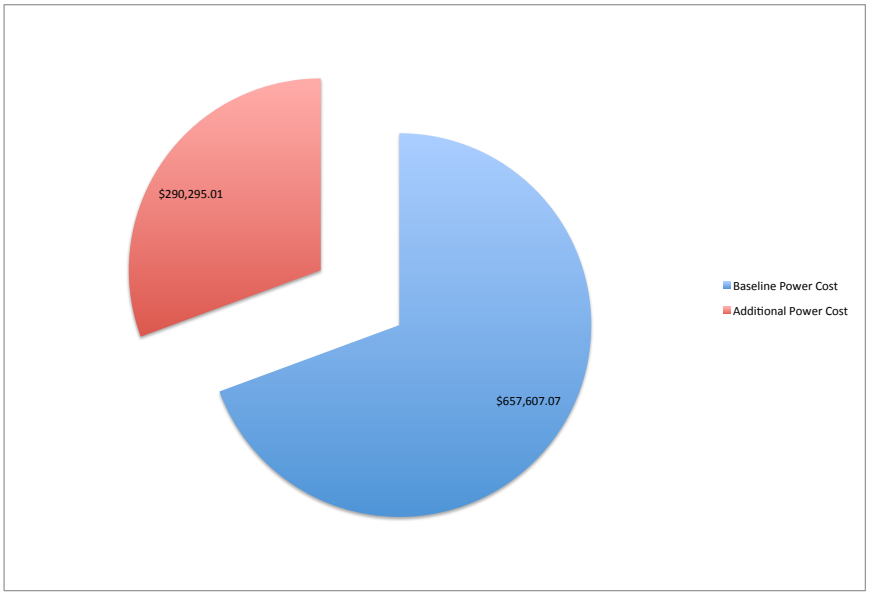

Figure 4.6. Additional Power Expense of Fully Loaded vs Idle Campus Grid

Table 4.6.

Summary of Usage Metrics for Purdue Campus Grid

Year	Unique Users	Unique PIs	Unique PI Departments	Unique Fields of Science	Jobs Completed	Hours Delivered
2005	25	8	5	4	295,265	1.9 million
2006	70	27	11	11	4.44 million	4.61 million
2007	115	50	16	19	9.93 million	8.17 million
2008	115	60	13	18	14.9 million	16.6 million
2009	163	85	18	16	15.4 million	17.9 million
2010	145	79	20	16	15.2 million	18.6 million

Table 4.7.

Summary of Time-to-Result of Purdue Campus Grid

Year	Avg Run Hours	Avg Time to Result
2005	7.49	57.20
2006	1.07	4.31
2007	0.83	2.42
2008	1.15	3.67
2009	1.19	3.35
2010	1.25	4.33
Average	2.16	3.88

Flint (personal communication, March 24, 2011) describes the results for 2005 as anomalous:

> .... because one user ran 36% of the jobs and each of his jobs averaged 14.5 run hours and 123.6 result hours. Another ran about 15% [of the jobs] and averaged 6.4 run hours and 14.8 result hours. With those two user entries removed, 2005 has 2.97 avg run hours and 19.9 avg result hours.

With the exception of 2005, the average runtime of a job in the Purdue grid remains remarkably consistent, varying a maximum of only approximately 15 minutes. Time to result, though, varies to a wider degree, due to a wide variety of outside factors including use of the machines in the grid by their primary owners. For example, a job that is preempted and restarted will exhibit a longer time to result - due to time being lost by a job terminating and restarting from the beginning.

The best metric possible would be the number of scientific publications in peer-reviewed journals that can be attributed to the use of the Purdue campus grid. Unfortunately, this information is not available today, and gathering it through a manual literature search is beyond the scope of this study.

4.2 Analysis of the Data

While Table 4.5 shows the details of the yearly full additional costs of running the campus grid, Table 4.8 and Figure 4.7 depict the totals of the additional per core-hour expense of the hours facilitated by the campus grid.

Table 4.8.

Total Additional per Core-Hour Costs for Use of Purdue Campus Grid, by Year

Year	Cost
2005	$7,118.67
2006	$16,878.91
2007	$29,884.52
2008	$60,739.12
2009	$65,846,19
2010	$68,082.94
Average	$41,425.06

4.2.1 Total Cost for Each Metric

However, these numbers do not gain their full value until considering what is gained by incurring each of these expenses.

Equation 4.5 provides a model for using the total per node-hour cost that was obtained in Section 4.1.3.2 ($0.03984680123) to calculate the per node-hour cost per unit of a usage metric:

$$C_{tot} = \frac{H_y * C_t}{M_u}, \tag{4.5}$$

where H_y is number of hours that the campus grid provided in the year, and C_t is the total cost of an hour of use in the campus grid, as discussed in Section 4.1.3.3. M_u is

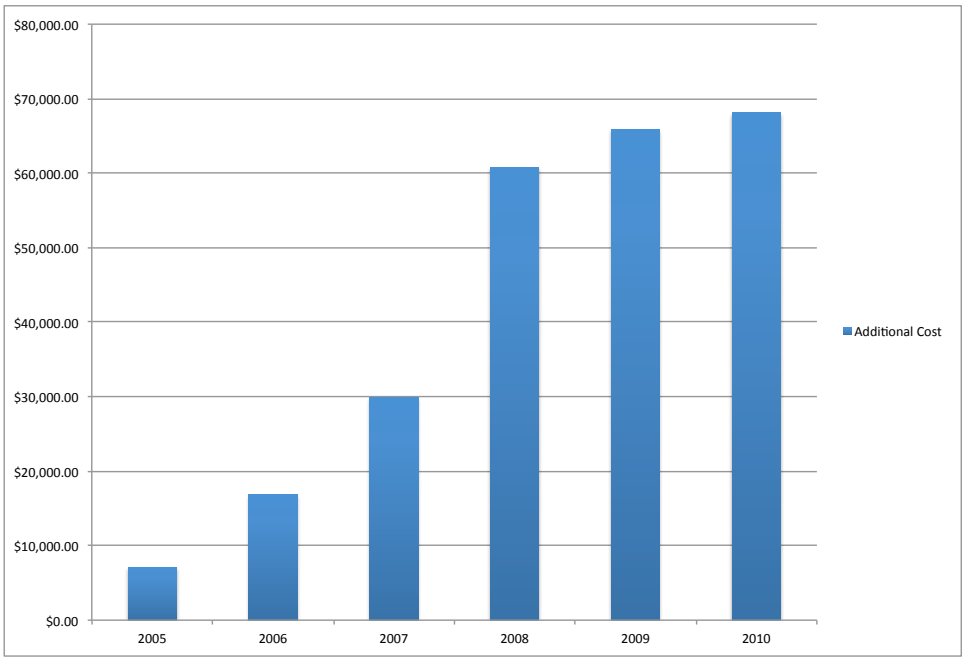

Figure 4.7. Total Additional per Core-Hour Costs for Use of Purdue Campus Grid, by Year

the metric of usage, as discussed in Section 4.1.4, such as users, PIs, or departments. The result, C_{tot}, is the total per-core-hour cost per unit of M_u.

For example, Equation 4.6 shows the results of computing C_{tot} for the number of unique users in 2005:

$$\begin{aligned} C_{tot} &= \frac{H_y * C_t}{M_u} \\ &= \frac{(1,945,723 \text{ hours} * \$0.0398468012)}{25 \text{ Unique Users}} \\ &= \$3,101.23 \text{ per user} \end{aligned} \tag{4.6}$$

Table 4.9 and Figure 4.8 summarize C_{tot} per user, principal investigator (PI), PI department, and field of science, as calculated by Equation 4.5

Table 4.9.

Summary of Per-Metric Costs for Hours Delivered by Purdue Campus Grid

Year	Per User	Per PI	Per PI Department	Per Field of Science
2005	$3,101.23	$9,691.35	$15,506.17	$19,382.71
2006	$2,626.17	$6,808.58	$16,711.97	$16,711.97
2007	$2,830.25	$6,509.57	$20,342.40	$17,130.44
2008	$5,752.37	$11,025.37	$50,886.31	$36,751.22
2009	$4,399.66	$8,436.99	$39,841.34	$44,821.51
2010	$5,113.83	$9,386.14	$37,075.26	$46,344.07
Average	$3,970.58	$8,643.00	$30,060.57	$30,190.32

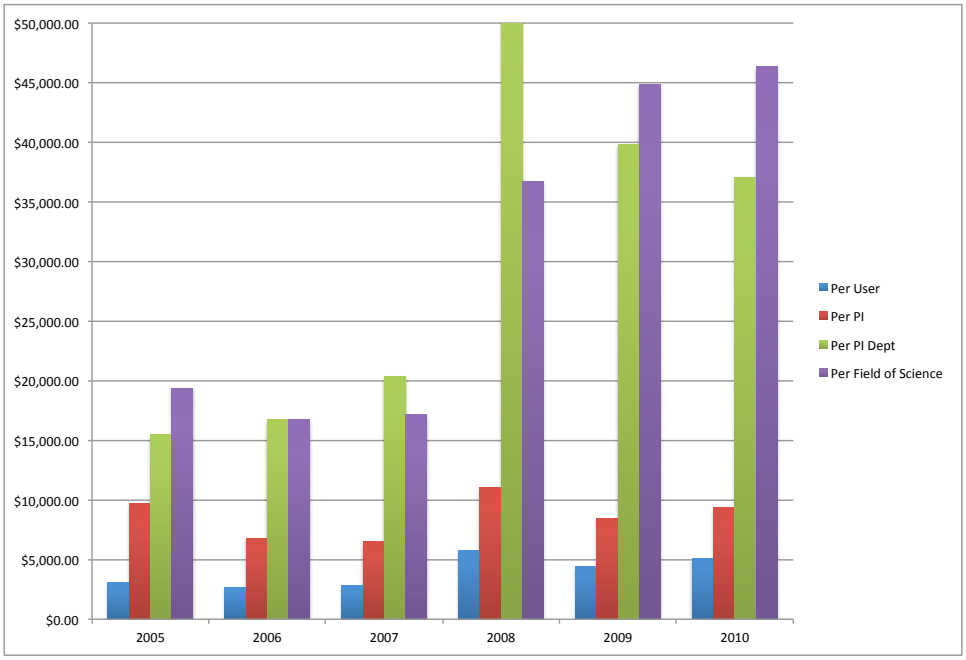

Figure 4.8. Cost Per Metric of Purdue Campus Grid

4.2.1.1. Notes on the Relationship Between Metric Costs

The reader may question why per-metric costs rise each year, along with usage instead of falling. This is due to the per core-hour cost model being a variable cost method of measuring the cost of the grid. Therefore, all metrics presented will rise with usage.

Additionally, the reader may question why Figure 4.8 shows the relationship between C_{tot} per User and per PI cost metrics remaining fairly constant from year to year, but per C_{tot}PI Department and per Field of Science C_{tot} spike dramatically in 2008-2010. Figure 4.9 illustrates the relationships between the metrics in Table 4.6. Figure 4.10 illustrates the relative growth of the hours and jobs provided on the Purdue campus grid.

It should be noted that the slope of unique users and PIs in Figure 4.9 closely matches that of jobs and hours delivered in Figure 4.10, while there is relatively flat growth in departments and fields of science. It may be concluded that due to the finite number of departments at an institution, and limited fields of science, that measuring costs per department or fields of science may not be a useful metric for the cost and benefits of a campus grid.

4.2.2 Additional Cost for Each Metric

Bearing in mind that a large portion of C_{tot} are already absorbed by the institution, the most important question to be answered is that of "what is the additional cost incurred for each metric", or C_{addtl}?

Similar to Equation 4.5, Equation 4.7 provides a model for using the additional per node-hour cost (C_a) that was obtained in Section 4.1.3.2 to calculate the additional per node-hour cost (C_{addtl}) per unit of a usage metric:

$$C_{addtl} = \frac{H_y * C_a}{M_u}, \tag{4.7}$$

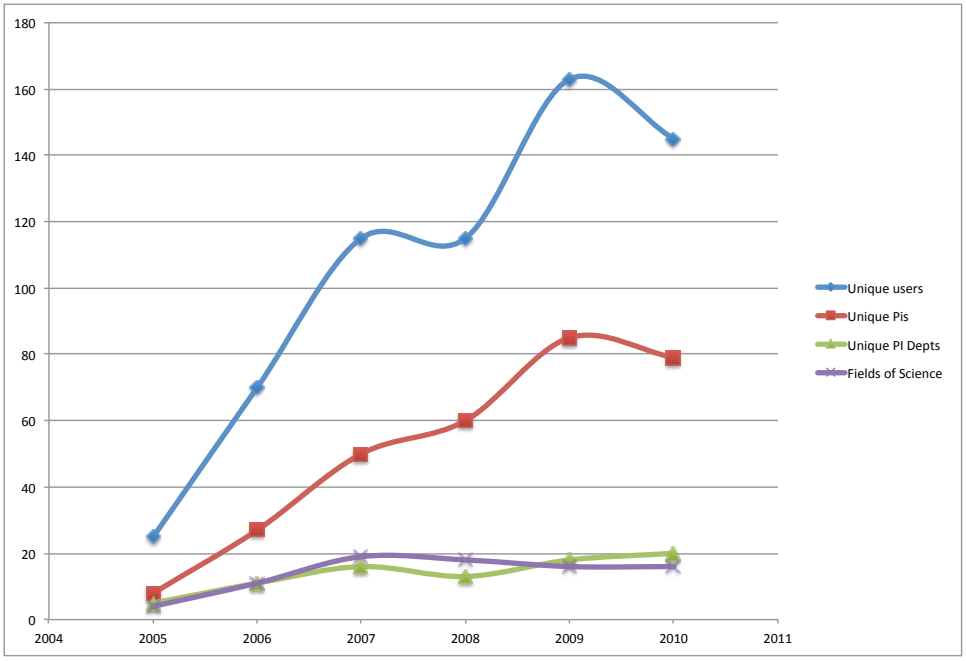

Figure 4.9. Relative Growth of Metrics of Purdue Campus Grid

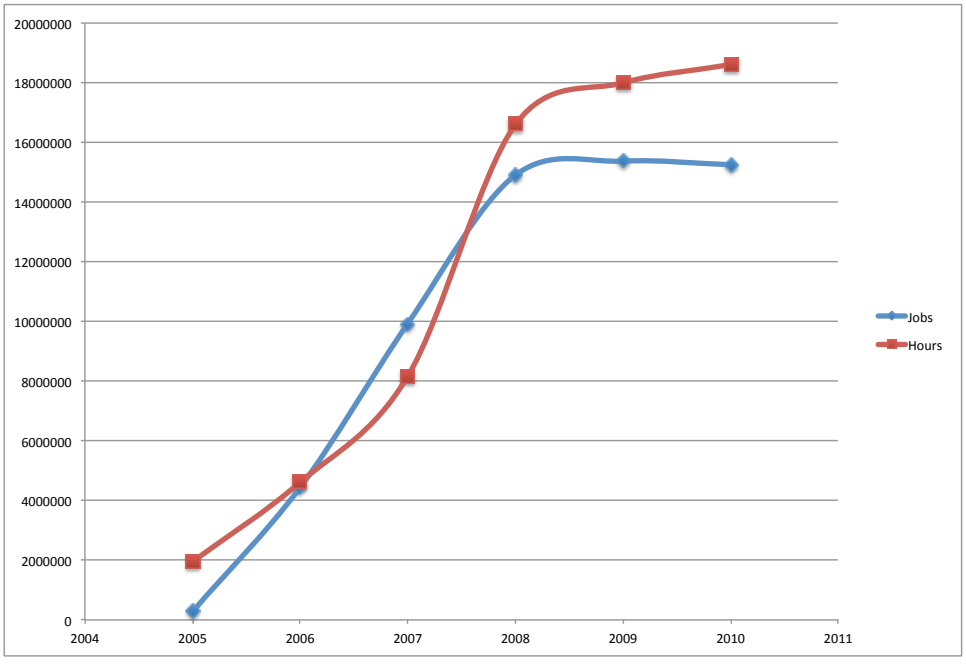

Figure 4.10. Relative Growth of Hours and Jobs of Purdue Campus Grid

where H_y is number of hours that the campus grid provided in the year, and C_a is the additional cost of an hour of use in the campus grid, as discussed in Section 4.1.3.3. M_u is the metric of usage, as discussed in Section 4.1.4, such as users, PIs, or departments. The result, C_{addtl}, is the total additional per-core-hour cost per unit of M_u.

For example, Equation 4.8 shows the results of computing C_{addtl} for the number of unique users in in 2005:

$$\begin{aligned} C_{addtl} &= \frac{H_y * C_a}{M_u} \\ &= \frac{(1,945,723 \text{ hours} * \$0.003658623)}{25 \text{ Unique Users}} \\ &= \$284.75 \text{ per user} \end{aligned} \tag{4.8}$$

Table 4.10 and Figure 4.11 summarize C_{addtl} incurred by the institution per user, principal investigator (PI), PI department, and field of science, as calculated by Equation 4.7.

As with C_{tot}, C_{addtl} exhibits the same increase in costs for PI Departments and Field of Science in relation to users and PIs. Again, this is due to flat growth in PI departments and Fields of Science in relation to users and PIs. Refer to Section 4.2.1.1 for explanation of this characteristic of the data.

Finally, as noted in Section 4.1.4, data on the number of publications attributable to the campus grid is unfortunately not available. Future work may investigate the number of publications made possible by the computing grid.

Table 4.10.

Summary of Additional Per-Metric Costs for Hours Delivered by Purdue Campus Grid

Year	Per User	Per PI	Per PI Department	Per Field of Science
2005	$284.75	$889.83	$1,423.73	$1,779.67
2006	$241.13	$625.14	$1,534.45	$1,534.45
2007	$259.87	$597.69	$1,867.78	$1,572.87
2008	$528.17	$1,012.32	$4,672.24	$3,374.40
2009	$403.96	$774.66	$3,658.12	$4,115.39
2010	$469.54	$861.81	$3,404.15	$4,255.18
Average	$364.57	$793.58	$2,760.08	$2,771.99

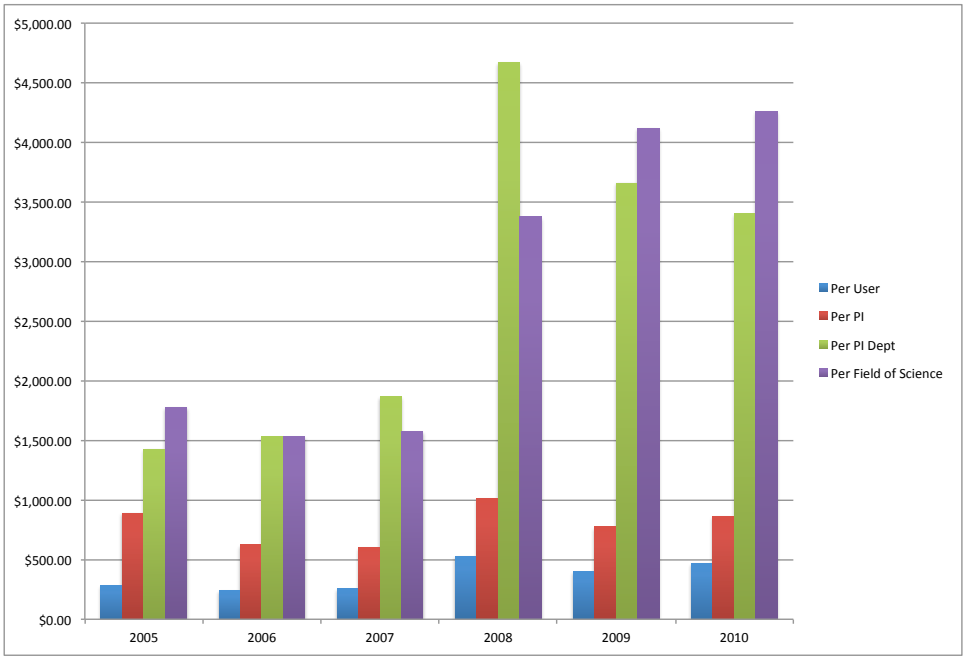

Figure 4.11. Additional Cost Per Metric of Purdue Campus Grid

CHAPTER 5. CONCLUSIONS, DISCUSSION AND RECOMMENDATIONS

5.1 Conclusions

In conclusion, we have established that a higher education institution needs a model for making an informed decision on how to provide cyberinfrastructure to support research. I proposed to define a model for measuring the costs and benefits of building a campus computing resource based on the institution's existing investment in computing hardware.

Furthermore, I evaluated this model with costs from the campus of Purdue and compared the results to a known price, Amazon EC2.

There are many studies that explore the costs of grid or high performance computing, such as (Opitz et al., 2008), (Gray, 2008), (Kondo et al., 2009), (Walker, 2009), (Beck et al., 2008), and (Carlyle et al., 2010).

A number of other works define methods for measuring the capacity of a grid, including (Mutka, 1988), (Thain et al., 2006), (Anderson & Fedak, 2006), (Toth & Finkel, 2007), (Brevik et al., 2004), (Wolski et al., 2007).

But when searching the literature for works about the utility, or productivity of a computing grid, there is much less work to draw from, as noted in Section 2.2.3.

Therefore, with this thesis I have set out to contribute a work that will fill this gap in the literature. The models refined or developed herein to meet this goal are summarized below:

5.1.0.1. The Cost Models

I adapted the model defined by Carlyle et al., to normalize the raw cost of a core-hour in relation to Amazon EC2. This model is described in Section 4.1.3.1, and summarized below:

$$C_n = \frac{C_h}{F_n},$$

where C_h is the pre-normalized per core-hour cost of a core on the campus grid, and F_n is a constant representing the normalizing factor of one hour on the grid to 1 hour on EC2. The result, C_n, is the normalized per-core-hour cost.

After identifying additional costs specific to the operation of the campus grid as described in Section 4.1.3.3, I determined that the following model will calculate the extra cost per core-hour of operating the campus grid.

$$C_a = \frac{E_y * S_a}{H_y}$$

Finally, with C_n and C_a calculated as described in Section 4.1.3.4, I identified a model to calculate C_t, the total per-core hour cost of a campus grid.

$$C_t = C_h + C_a$$

5.1.0.2. The Dollars per Measure of Productivity Model

Given a set of metrics such as those described in 4.6, I defined a model to compute the dollar cost per measure of productivity. Equation 4.5 describes a model to calculate the per node-hour cost per unit of a usage metric (C_{tot}), and Equation 4.7 describes a model to calculate the additional per node-hour cost per unit of a usage metric (C_{addtl}). Both models are summarized below:

$$C_{tot} = \frac{H_y * C_t}{M_u}$$

$$C_{addtl} = \frac{H_y * C_a}{M_u},$$

where H_y is number of hours that the campus grid provided in the year, C_t is the total cost of an hour of use in the campus grid, and C_a is the additional cost of an hour of use in the campus grid, both as discussed in Section 4.1.3.3. M_u is the metric of usage, as discussed in Section 4.1.4, such as users, PIs, or departments.

The results, C_{tot} or C_{addtl}, are the total or additional per-core-hour cost per unit, respectively, of M_u.

5.1.0.3. The Output of the Models

In this thesis, using cost data and usage metrics from Purdue University's campus grid as input, I utilized these models to determine C_n and C_a for that institution. C_n and C_a, when added together, yield C_t - the total per core-hour cost of the Purdue campus grid. These results are summarized in Table 5.1.

Table 5.1.

Summary of Cost Model Results for Purdue Campus Grid

Model	Cost
C_n	\$0.03619
C_a	\$0.003658623
C_t	\$0.0398468012

With C_n, C_a and C_t calculated, I was able to use usage metrics in Table 4.6 of the Purdue grid to compute C_{tot} and C_{addtl} for a number of usage metrics.

C_{tot} values for a variety of usage metrics are summarized in Table 4.9, and C_{addtl} values for the same metrics are summarized in Table 4.10.

5.2 Future Work

Future work will explore the relationships of runtime and time to results vs. the productivity metrics that were detailed in Table 4.6. Incorporating time to results as a part of a future cost model would be a useful topic to explore. Is it possible to determine a relationship between time to results and the cost of a computing resource?

Section 4.1.2.2 notes the relationship between CPU clock speed, Condor's Kflops metric, and NPB run times. Future work should further evaluate this relationship and study campus grid usage logs to determine the fraction of campus grid jobs that are CPU-intensive, I/O-intensive, etc.

Additionally, a study performing further investigation of the number and type of publications made possible by the campus grid would help create a new and meaningful measurement of one of the most important benefits made possible by the campus grid.

Finally, a study that executes a number of loosely-coupled codes ideal for execution in a campus grid to identify some ideal codes for use as benchmarks of a campus grid is another potential research area.

5.3 Summary

I have measured the capacity of a campus grid to see what fraction of it can reasonably be expected to be usable for computation. At Purdue, an average 13,526 processor cores are available at any given time.

I have benchmarked the campus grid to determine its performance relative to a community cluster and Amazon EC2. This provides a scaling factor with which to normalize the cost of a CPU hour, allowing the costs calculated to be compared with those presented by Carlyle et al.

With this factor, I have determined that the normalized cost to Purdue University for a core-hour in the campus grid is $0.03619. Adding in additional expense dedicated to the campus grid (extra staff, dedicated hardware, additional power load), I then found that the additional expense to run a campus grid on an institution's existing investment in computing equipment is $0.003658623 (3.66 tenths of a cent) per core-hour.

Therefore, the total cost of a core-hour is $0.039847.

As an example, in 2010, the additional expenses incurred to operate a campus grid and complete 15.2 million jobs using 18.6 million hours, all on hardware that would otherwise be idle, was $68,082.94. Each user utilizing this resource only cost Purdue an additional$469.54, and each faculty member (which may have more than one user associated with them) utilizing the grid costs the institution an additional $861.81.

This investment allows the creation of a very cost-effective high-performance computing resource. On average, it costs only $364.57 per user each year on top of the University's sunk investment in information technology to provide a campus grid service that is usable by and appropriate for a large portion of the researcher population. For example, if an institution were to build an equally-sized resource on Amazon EC2 (13,526 cores), it would cost $16,939.86 per user.

LIST OF REFERENCES

LIST OF REFERENCES

Acharya, A., Edjlali, G., & Saltz, J.(1997). The utility of exploiting idle workstations for parallel computation. In *Sigmetrics '97: Proceedings of the 1997 acm sigmetrics international conference on measurement and modeling of computer systems* (pp. 225–234). New York, NY, USA: ACM.

Afgan, E., & Bangalore, P.(2007). Computation cost in grid computing environments. In *Esc '07: Proceedings of the first international workshop on the economics of software and computation* (p. 9). Washington, DC, USA: IEEE Computer Society.

Altair Corporation. (2010). *Altair corporate - innovation intelligence.* (Retrieved from: http://www.altair.com)

Amazon, Inc. (2008). *Amazon elastic compute cloud (Amazon EC2).* http://aws.amazon.com/ec2/#pricing: Amazon Inc.

Anderson, D. P., & Fedak, G. (2006). The computational and storage potential of volunteer computing. In *Ccgrid '06: Proceedings of the sixth ieee international symposium on cluster computing and the grid* (pp. 73–80). Washington, DC, USA: IEEE Computer Society.

Bailey, H., David, N., Bailey, H., Barszcz, E., Barton, J., Browning, D., et al.(1994). *Title: The nas parallel benchmarks* (Tech. Rep.).

Beck, R., Schwind, M., & Hinz, O. (2008). Grid economics in departmentalized enterprises. *J. Grid Comput.*, *6*(3), 277-290.

Brevik, J., Nurmi, D., & Wolski, R. (2004). Automatic methods for predicting machine availability in desktop grid and peer-to-peer systems. In *Ccgrid '04: Proceedings of the 2004 ieee international symposium on cluster computing and the grid* (pp. 190–199). Washington, DC, USA: IEEE Computer Society.

Carlyle, A. G., Harrell, S. L., & Smith, P. M. (2010, December). Cost-effective HPC: The community or the cloud. *IEEE Conference on Cloud Computing Technology and Science*.

Catlett, C. E.(2005). TeraGrid: A foundation for US Cyberinfrastructure. In H. Jin, D. A. Reed, & W. Jiang (Eds.), *Npc* (Vol. 3779, p. 1). Springer.

Dongarra, J., Meuer, H., & Strohmaier, E.(1998). *Top500 supercomputer sites* (Tech. Rep.). Knoxville, TN, USA.

Dongarra, J. J., Luszczek, P., & Petitet, A. (2003). The linpack benchmark: Past, present, and future. concurrency and computation: Practice and experience. *Concurrency and Computation: Practice and Experience*, *15*, 2003.

Earl, D. J., & Deem, M. W.(2006). Toward a database of hypothetical zeolite structures. *Industrial & Engineering Chemistry Research*, *45*(16), 5449-5454.

Fermi National Accelerator Lab. (2010). *Fermigrid.* (Retrieved from: http://fermigrid.fnal.gov/)

Foster, I., & Kesselman, C.(1997). Globus: a Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications*, *11*(2), 115-128.

Gopu, A., Repasky, R., & McCaulay, S.(2007). *Survey of teragrid job distribution: Toward specialized serial machines as teragrid resources.*

Gray, J.(2008). Distributed computing economics. *ACM Queue*, *6*(3), 63–68.

Jiang, Wen, Baker, Matthew L., Jakana, Joanita, Weigele, Peter R., King, Jonathan, & Chiu, Wah.(2008, February 28). Backbone structure of the infectious epsilon15 virus capsid revealed by electron cryomicroscopy. *Nature*, *451*(7182), 1130–1134.

Kaib, N. A., & Quinn, T.(2009). Reassessing the Source of Long-Period Comets. *Science*, *325*(5945), 1234-1236.

Kennedy, K., Koelbel, C., & Schreiber, R.(2004). Defining and measuring the productivity of programming languages. *International Journal of High Performance Computing Applications*, *18*(4), 441-448.

Kondo, D., Javadi, B., Malecot, P., Cappello, F., & Anderson, D. P.(2009). Cost-benefit analysis of cloud computing versus desktop grids. In *Ieee international parallel & distributed processing symposium* (p. 1-12).

Kondo, D., Taufer, M., Brooks, C. L., Casanova, H., & Chien, A. A.(2004). Characterizing and evaluating desktop grids: An empirical study. In *Ieee international parallel & distributed processing symposium.*

Kriebel, C. H., & Raviv, A.(1980). An economics approach to modeling the productivity of computer systems. *Management Science*, *26*(3), 297 - 311.

Kuck, D. J.(2004, November). Productivity in high performance computing. *Int. J. High Perform. Comput. Appl.*, *18*, 489–504.

Livny, M., Basney, J., Raman, R., & Tannenbaum, T.(1997, June). Mechanisms for high throughput computing. *SPEEDUP Journal*, *11*(1).

Mutka, M. W.(1988). *Sharing in a privately owned workstation environment.* Unpublished doctoral dissertation, University of Wisconsin-Madison.

Nurmi, D., Brevik, J., & Wolski, R.(2005). Modeling machine availability in enterprise and wide-area distributed computing environments. In *Euro-par* (p. 432-441).

Opitz, A., König, H., & Szamlewska, S.(2008). What does grid computing cost? *J. Grid Comput.*, *6*(4), 385-397.

Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., et al.(2007). The open science grid. *Journal of Physics: Conference Series*, *78*, 012057 (15pp).

Purdue University. (2009). *Data digest.* (Retrieved from: http://www.purdue.edu/datadigest)

Purdue University. (2010a). *Community cluster program.* (Retrieved from: http://www.rcac.purdue.edu/clusterprogram.cfm)

Purdue University. (2010b, February). *Diagrid.* (Retrieved from: http://www.diagrid.org/)

Sassone, P. G. (1988). Cost benefit analysis of information systems: a survey of methodologies. In *Proceedings of the acm sigois and ieeecs tc-oa 1988 conference on office information systems* (pp. 126–133). New York, NY, USA: ACM.

Schmid, P. (2010, January). *System power consumption: Intel intros 3-series chipsets with fsb1333 and ddr3.* (Retrieved from: http://www.tomshardware.com/reviews/intel-intros-3-series-chipsets-fsb1333-ddr3,1607.html)

Shuey, M., Veldman, G., Baumbauer, C., Finnegan, P., McKay, D., & Goasguen, S. (2005). Local community clusters experiences with resource sharing in international and national grid infrastructure. In *2005 nsf cise/cns infrastructure experience workshops.*

Silverstein, J. (2005, December). (Retrieved from: http://abcnews.go.com/Technology/story?id=1410682)

Smith, P. M., Hacker, T. J., & Song, C. X.(2008). Implementing an industrial-strength academic cyberinfrastructure at Purdue University. In *Ieee international parallel & distributed processing symposium* (p. 1-7).

Snir, M., & Bader, D. A.(2004). A framework for measuring supercomputer productivity. *International Journal of High Performance Computing Applications*, *18*(4), 417-432.

Sterling, T.(2004). Productivity metrics and models for high performance computing. *International Journal of High Performance Computing Applications*, *18*(4), 433-440.

Streeter, D. N. (1972, September). Cost-benefit evaluation of scientific computing services. *IBM Syst. J.*, *11*, 219–233.

Sun Microsystems. (2009, December). (Retrieved from: http://www.sun.com/solutions/cloudcomputing/index.jsp)

Tally, S.(2008, September). *Sleeping computers and peripherals waste energy, Purdue CIO says.* (Retrieved from: http://news.uns.purdue.edu/x/2008b/080917T-McCartneySave.html)

Thain, D., Tannenbaum, T., & Livny, M.(2005). Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, *17*(2-4), 323-356.

Thain, D., Tannenbaum, T., & Livny, M.(2006). How to measure a large open-source distributed system: Research articles. *Concurr. Comput. : Pract. Exper.*, *18*(15), 1989–2019.

Toth, D., & Finkel, D. (2007). Characterizing resource availability for volunteer computing and its impact on task distribution methods. In *Sepads'07: Proceedings of the 6th wseas international conference on software engineering, parallel and distributed systems* (pp. 107–114). Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS).

University of Wisconsin, Madison. (2010). *Grid laboratory of wisconsin.* (Retrieved from: http://www.cs.wisc.edu/condor/glow)

Walker, E. (2009). The real cost of a cpu hour. *Computer*, *42*(4), 35 -41.

Weicker, R. P. (1984, October). Dhrystone: a synthetic systems programming benchmark. *Commun. ACM*, *27*, 1013–1030.

Wolski, R., Nurmi, D., & Brevik, J. (2007). An analysis of availability distributions in condor. In *Ieee international parallel & distributed processing symposium* (p. 1-6).

York, R. (2002). *Benchmarking in context: Dhrystone* (Tech. Rep.). ARM Ltd.

APPENDIX

APPENDIX: SCRIPTS

A.1 Condor Data Collection Script

```
#!/bin/sh

cd ~/thesis
stamp=‘date "+%F-%R"‘
stamp2=‘date "+%s"‘

cnd=‘/opt/condor/bin/condor_status -total | tail -1‘
echo $stamp2 $cnd  >> data.txt
cnd2=‘/opt/condor/bin/condor_status -total -pool egret \
| tail -1 ‘
echo $stamp2 $cnd2 >> desktop-data.txt

/opt/condor/bin/condor_status -format ’%s\n’ Machine | sort -u  \
> machines.txt.$stamp

gzip machines.txt.$stamp
```

A.2 Condor Data Analysis Script

```
#!/usr/bin/perl
use List::Util qw(sum);
```

```
$FILE=shift;
@percents;
@slots;

open FILE, $FILE || die "Can't open the data file";

while(<FILE>) {

 next if /Backfill/;
 chomp;
 if(/^\d/)  {
 ($stamp, $slots, $owner, $claimed, $unclaimed) = ($1, $2, $3, $4, $5) if
/(\d+)\s+Total\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)/;
 } else {
  ($slots, $owner, $claimed, $unclaimed) = ($1, $2, $3, $4) if
/\s+Total\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)/;
}

 #next if $slots < 10000;
 $free = $slots-$owner;
 $percent = sprintf("%.3f", $free/$slots);
 #print "Total Free Slots: $free of $slots (" . $percent*100 . " percent)\n";

 push(@percents, $percent);
 push(@slots, $slots);

}
```

```
$avgfree = sum(@percents)/@percents;
$avgslots = sum(@slots)/@slots;

print "Average percent of slots free: " .
$avgfree*100 . "% (of ". scalar @percents . " samples)\n";
print "\nAverage number of slots in grid: " .
$avgslots . " (of ". scalar @slots . " samples)\n";

close FILE;
```