

Implementing an Industrial-Strength Academic Cyberinfrastructure at Purdue University

Preston M. Smith ^{‡*}, Thomas J. Hacker ^{*‡}, Carol X. Song [†]

[†]Rosen Center for Advanced Computing,

^{*}Computer & Information Technology, College of Technology

[‡]Discovery Park Cyber Center

Purdue University, West Lafayette, IN

Abstract

Purdue University operates one of the largest cycle recovery systems in existence in academia based on the Condor workload management system. This system represents a valuable and useful cyberinfrastructure (CI) resource supporting research and education for campus and national users. During the construction and operation of this CI, we encountered many unforeseen challenges and benefits unique to an actively used infrastructure of this size. The most significant problems were integrating Condor with existing campus HPC resources, managing resource and user growth, coping with the distributed ownership of compute resources around campus, and integrating this CI with the TeraGrid and Open Science Grid. In this paper, we describe some of our experiences and establish some best practices, which we believe will be valuable and useful to other academic institutions seeking to operate a production campus cyberinfrastructure of a similar scale and utility.

1. Introduction

1.1. Condor

Condor [20] is a high-throughput batch computing system that provides job and resource management functionalities coupled with mechanisms to create and manage scheduling policies. Condor is a product of the Condor Research Project at the University of Wisconsin Computer Science Department, led by Professor Miron Livny. 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 ca-

capacity over very long periods of time" [11]. 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 operational realities of providing a production research cyberinfrastructure.

A unique and critical functionality of Condor that we rely on is cycle recovery, or cycle scavenging. 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. Our Condor installation primarily provides recovered cycles from UNIX based systems, since the main platform of our science and engineering researchers is commodity Linux cluster systems.

1.2. Condor at Purdue: Motivation

The Rosen Center for Advanced Computing (RCAC) is a division of Information Technology at Purdue (ITaP), the central computing and telecommunications organization on Purdue's main campus in West Lafayette, IN. RCAC offers a wide variety of computational resources for Purdue researchers: a medium-sized IBM SP, five 24-processor, large-memory Sun F6800 systems, a 128-CPU SGI Altix, and numerous Linux clusters.

The most powerful Linux cluster resources operated by RCAC are *community clusters* [16]. A *community cluster* is a system purchased by a faculty group and operated by RCAC, and is maintained for the benefit of the research group that owns the cluster as well as the broad campus user community. The driving concept motivating the community cluster model at Purdue is to exploit the economies of scale from centralizing hardware selection, operations, and support staff within

one organization, while maintaining faculty autonomy in the selection and use of high performance computing resources. Community cluster users benefit from centrally provided maintenance of both hardware and software by a professional IT staff, which frees researchers and graduate students in their laboratories to focus on scientific investigations [10].

The scheduling system on the community clusters at Purdue follows a two-tier model. The first tier is controlled by the primary scheduler, which has the highest scheduling priority on all of the nodes, and has the right to preempt any computational job running on any node that it did not deploy. The second tier is the cycle recovery service, which seeks to find and put into use idle computational nodes. The primary scheduler used at Purdue is the Portable Batch System (PBS) from Altair Engineering [1]. Condor and PBS are configured to share information about their respective states such that an individual node will only be willing to run a Condor job when there are no current PBS scheduled jobs running on that node. As described in Figure 1, when PBS indicates that a node is idle, Condor is permitted to schedule a job on that node. A running Condor job will be subject to preemption by incoming PBS jobs, similar to the case in which a Condor job running on a desktop workstation may be preempted by keyboard or mouse activity.

Efforts to reclaim unused cycles only make sense if the amount of potentially reclaimable resources are significant. Mutka [13] measured the overall available time of a group of workstations at the University of Wisconsin, and found that only 71% of resources were used, leaving nearly a third unused. At Purdue, we determined that the community cluster owners use approximately 70% of the available wall clock time. Leaving these systems idle 30% of the time made little sense. Idle computers consume a full 50% of the power draw of a fully utilized system. At this rate of power consumption, idle systems still require significant amounts of cooling. We decided that the hidden costs of permitting idle systems to take up precious data center space and resources greatly exceeded the costs of operating a cycle recovery service. In response, the Rosen Center turned to Condor to provide cluster scheduling backfill to allow Purdue to harvest these unused cycles on Linux clusters.

2. Deployment

In three years of operation, Condor has proven to be very popular, and has facilitated an increase in the use of cluster systems to an average in excess of 95%

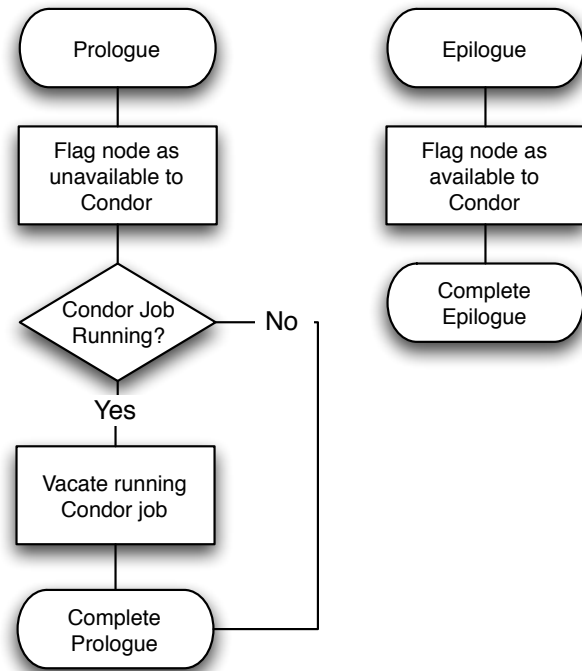


Figure 1. Flowchart of Condor/PBS interaction

utilization in 2007.¹ Opportunistic use by the campus Condor community occasionally exceeds the fraction of time the system is utilized by cluster owners. Foremost, the overall utilization (98.2% for the "Lear" cluster in September, 2007) far exceeds the fraction of time the system lies idle, which significantly increases the value of the cluster hardware for the entire Purdue research enterprise.

The value delivered by Condor to the Purdue research enterprise is summarized in Table 2.

Year	Pool size	Jobs	Hours Delivered
2004	1500	43,551	.346 million
2005	4000	210,717	1.695 million
2006	6100	4,251,981	5.527 million
2007	7700	9,611,813	9.524 million

Table 1. Condor usage summary by year

By late 2005, the groups operating the Condor infrastructure found that as the installation had grown over time it had difficulties keeping up with the rate of growth of clusters in RCAC, and needed a major

¹For an illustration of cluster utilization, visit <http://web.rcac.purdue.edu/condorview>

overhaul. The next section will describe several of the challenges we faced in scaling Condor to thousands of nodes and many active users.

3. Challenges

While taking the leap to an enterprise-scale Condor system, we uncovered several problems. The Condor installation needed to be updated; checkpoint infrastructure was insufficient; too many execute hosts used a single central manager, and too few submit hosts were available to users. With expansion came new challenges in usage reporting, storage, and networking. And finally, expanding the Purdue Condor pool outside of the Rosen Center to include resources and users across campus and other Indiana universities increased the need for support and administration, specifically user engagement and IT staff training.

3.1. Upgrading Condor

The first step in running a leading-edge Condor campus grid is to keep the Condor installation current. In late 2005, the Condor installation at RCAC was upgraded from a May 2004-released installation of Condor 6.6.5 to Condor 6.6.11, the latest stable version at the time. Subsequently, we've found that it is often preferable to actually use the *development* series of Condor to gain early access to performance improvements and new features.

For example, in 2006, [19] the negotiation protocol was dramatically sped up by the addition of *SIGNIFICANT_ATTRIBUTES* functionality, changing TCP connections to the *startd* to be non-blocking, and implementing caching on the *negotiator*, among others. This dramatically improves the ability of the Condor *negotiator* to match jobs to machines, giving improved throughput to users in a busy pool such as ours.

In 2007, in addition to further improvements to the *collector* and *negotiator*, performance improvements were made to the *schedd*, the Condor daemon responsible for queueing and job submission, [18] which made the *schedd* able to submit run 5 times the number of jobs in 1/5 of the time, and not block while trying to both schedule jobs and service *condor_q* requests.

These critical improvements were released into Condor's development series, and the ability to quickly adopt and deploy new versions allowed us to make Condor more responsive and useful to our users.

3.2. Checkpoint Servers

Many computational tasks scheduled through Condor can utilize the checkpoint and recovery functionalities provided by Condor. Since the Purdue Condor installation relies on cycle recovery and preemption, we found that the lack of adequate checkpointing services to be a problem.

We began in 2005 with a single checkpoint server containing 200GB of space providing checkpointing space for 1800 Condor nodes. A simultaneous checkpoint operation across a few hundred standard universe jobs overwhelmed this single server. A single mass checkpoint could easily fill the disk on the checkpoint server and saturate its single gigabit Ethernet interface.

To overcome this problem, we planned a small cluster of checkpoint servers, with enough aggregate disk space and bandwidth to be able to withstand the simultaneous checkpointing of 2/3 of the aggregate memory of all of the machines in the pools. In 2005, there was 3.6 TB of memory in the condor pools, which would require 2.4TB of disk space to hold that quantity of checkpoint images.

We deployed a set of seven checkpoint servers, each with 500GB of available space, which Condor nodes select from on a random basis, using the `$RANDOM_CHOICE` expression in the *condor_config* file.

In practice, this architecture has proved to be much more than adequate to service all of the checkpointing requirements at Purdue.

3.3. Central Manager

In 2005, a single central manager served 1800 Condor nodes, and due to performance limitations present in the Condor *negotiator* and *collector* at the time, the single host was unable to scale well enough to provide matchmaking services for all 1800 nodes. At that time, Thain [21] reported that the largest Condor pools in operation in the world contained approximately 2000 hosts. Our observations of the scalability of the single, older pool at Purdue matched up well with Thain's data, so we elected not to size our Condor pools on the leading edge of scale. To break this monolithic pool into smaller sets, we elected to operate the newly redesigned infrastructure as a *flock* [7] of Condor pools, each containing approximately 1000 nodes.

As time progressed, the need to operate several pools has been rendered moot by improvements to the Condor software, and our installation has transformed from one with five pools of the 1000-machine scale, into one with three pools with approximately 2500 machines

in each. Had we done this deployment in late 2007 instead of 2005, the installation would have been implemented considerably differently. However, we have found unexpected benefits in operating a flock of multiple pools. For example, we can logically separate pools that cannot be reached from off campus from pools that can, and can designate a single central manager and its pool as one dedicated to serving machines from all over campus.

3.4. Submission Host

Our initial Condor configuration required users to submit Condor jobs from a single submission host, which quickly became a bottleneck, due to the aforementioned limitations in the pre-version 6.9.3 *schedd*, as the Purdue Condor installation grew. When we created *flocks* to address scaling problems, the number of hosts configured as submission hosts was also increased to allow every cluster head node to submit jobs to Condor. We also installed extra user-accessible login nodes from which jobs could be submitted.

3.5. Usage tracking

Usage tracking within a large, multi-organization Condor flock is surprisingly difficult. By design, queue and history exist on the *schedd*, which makes collecting history information about completed jobs for accounting purposes difficult, especially when a submit host is a user's desktop. Options such as Gratia [4] or new capabilities in Quill available in the 6.9.4 version of Condor may prove to be solutions, but they all require additional effort on the part of the *schedd* owner, and as of this writing we have not yet evaluated them in the context of the campus grid. In the end, we used a collection of Perl scripts to process the Condor history files and load the information into a PostgreSQL database.

In many cases, this worked admirably, but in some cases, invalid wall clock times were reported. Condor documentation describes the difference between the `ClassAds RemoteWallClockTime` and `CumulativeSuspensionTime` as the total run time of a job. Sometimes strange values can appear in the `CumulativeSuspensionTime` `ClassAd`, such as a large negative number or a number that looks suspiciously like a timestamp, when `JobRunCount` is 1. After reporting these issues to the Condor developers for fixing, we added code to our usage tracking tools to monitor and react to this problem when it emerges.

3.6. Storage

We learned quickly that the most common problems encountered in a Purdue Condor flock are all related to storage. One hydrologic simulation run in Condor used 50-200 MB/second of disk I/O. Multiplying 50 MB/sec across 100 computational jobs quickly saturated all storage subsystems in operation.

Linux NFS servers proved to scale poorly for the large number of NFS clients running on Condor computational nodes. We found that a single user using the same fileserver for running 1000 jobs, each generating accompanying data files, output, error, DAGman, and user logs, could reliably crash the Linux NFS server.

Allowing users to write user and DAGman logs to NFS is a recipe for trouble: an unresponsive NFS server results in a *schedd* unable to lock and write the user log file, which cascades into a locked and unresponsive *schedd* process that is rendered unusable. The only recourse to this problem is a labor-intensive process of manually editing the `job_queue.log` file, which results in the loss of the contents of the Condor queue and unhappy users.

We pursued a two-pronged solution to address these problems. One approach is social: training users to forego the use of distributed filesystems in favor of local filesystems and Condor's file transfer capabilities. The second approach was technical: RCAC has phased out Linux NFS servers and replaced them with enterprise-grade NAS servers such as the BlueArc Titan [2].

3.7. Expansion to Campus

Based on our success operating the cycle recovery service on the community clusters operated by RCAC, we investigated expanding the collection of systems that could be managed using this two-tier model. We found a very large pool of underutilized resources that could be put to good use for the research community. 2200 Windows PCs in student labs around campus were already set up to run Condor for a distributed rendering application [12]; a 48-node GPU cluster in the Envision Center for Data Perceptualization was added to support the TeraDRE [9]; the College of Technology installed Condor on student lab machines; and Condor was installed on several hundred catalog terminals in the Purdue Libraries.

Through the Northwest Indiana Computational Grid (NWICG) [3], Purdue's Calumet campus installed Condor on its own instructional labs, and RCAC set up a flocking relationship with the Calumet campus and the University of Notre Dame. Today, Notre Dame appli-

cations regularly run transparently at Purdue and vice-versa.

As researchers began to widely use Condor for their high-throughput work, scientists in the Physics and Biology departments contributed their own machines into the pool. We have found that the flexibility principles described by Thain, Tannenbaum, and Livny [20] are crucial when reaching out to new communities of machine owners and users. Users contributing resources of their own to the campus grid need to feel confident that they retain control of what they allow to execute on their machines.

The resulting Condor flock, named "BoilerGrid", is now one of the largest academic Condor installations in the nation, offering 7700 job slots to users. BoilerGrid is available for allocation to NSF researchers through the NSF TeraGrid [5], as well as to the Open Science Grid (OSG) [15] through Purdue's role as a Compact Muon Solenoid (CMS) Project Tier-2 facility.

One challenge facing the operators of any grid of distributed ownership is agreeing on permission and access control. For example, until some agreement was made between machine owners, users in Physics and Biology were unable to execute jobs on each others' machines when submitting from their own desktop. And even today, users submitting from their desktops in Biology are not able to connect to machines in the campus' student labs. We have addressed this issue by the ensuring that all groups of machine owners allow submissions from RCAC systems, and encouraging users to do submissions from these central systems in RCAC. This has the added benefit of allowing us to funnel usage onto *schedds* on which we can access history information.

3.8. Education, Outreach, and Training

At the TeraGrid'07 conference held in Madison, Wisconsin, the Purdue team delivered a day-long training session on how to effectively use Condor to an audience of over a dozen users from around the nation.

Additionally, on our own campus, we have engaged the owners of large numbers of potential Condor nodes, teaching them now to run Condor, and preparing them to recognize the sorts of their users' problems that Condor is well-suited for solving. In October, 2007, we held the first Condor "boot camp" at the West Lafayette campus, hosting IT staff from 11 different organizations on campus, and from 8 other institutions across the state of Indiana, including one minority-serving institution.

3.9. Networking

By design, Condor requires bi-directional connectivity between submit hosts and execution nodes. As the scope of the Condor flock leaves the boundaries of Purdue and other other campuses, security mechanisms, networking policies, and other network-related issues proliferate. For example, network firewalls have proven to be a constant problem. Working with remote IT staff, we have been able to work through these issues by documenting recommendations for firewalls: open port 9816 to all *condor_collectors*, define HIGHPORT and LOWPORT for inter-daemon communication and open corresponding holes in firewalls for those port ranges.

Private (non-routable) IP spaces and NATs are more problematic. Clusters at Purdue are increasingly being assigned to private IP spaces, which complicates connectivity from Purdue clusters to remote partners. Potential solutions such as virtual private networks or GCB [17] exist, but have not yet been evaluated.

Finally, high demand can adversely affect the network link of a remote partner. Computational jobs matched by Condor to resources at Purdue's Calumet campus resulted in a significant amount of file transfer activity. This saturated the Calumet campus' external internet link, which impacted connectivity for the entire campus. As a result, Calumet instituted network quality-of-service limitations, and instituted policies delimiting when Condor jobs may be run on campus.

3.10. Staffing

At Purdue, a single system administrator can support the upkeep and management of the enterprise-scale Condor infrastructure with approximately 25% of his effort. Assuming that reasonable large-scale system administration infrastructure is already in place, we expect that this level of effort should hold true for other sites as well.

We have two science support staff (with roles much like "consultants" at national centers) with substantial Condor expertise, plus the systems administrator who provides some tier-2 end-user support. We find that more science support staff are necessary as the number of users from different disciplines using Condor at Purdue increases. In an environment with a wide variety of applications and potential users, investing in more staff effort to directly support the end user is recommended.

4. Application Case Studies

BoilerGrid has proven to be a tremendously valuable resource for the research community at Purdue and on the TeraGrid, with users from many disciplines including high-energy physics, forestry, hydrology, materials science, astrophysics, business, electrical engineering, and structural biology. Some individual research groups have been able to perform a staggering 5 million hours of computation in one year - all from recovered cycles!

This section highlights one application that benefited from the use of cycle recovery through Condor.

4.1. Building a Database of Hypothetical Zeolite Structures

Since early 2006, Dr. Michael Deem of Rice University and Dr. David Earl of Pitt have run the application *Zefsa II* [8] in the Condor flock at Purdue through an NSF TeraGrid allocation. *Zefsa II* is used for identifying hypothetical zeolite structures, which are porous crystalline materials that have a wide range of applications in catalysis, ion exchange, and molecular sieving applications. The procedure for identifying these structures require 200,000,000 executions of a parameter sweep, which was quickly identified as a perfect candidate for Condor and therefore for allocation at Purdue.

During the effort to adapt this application to effectively use Condor, Cheeseman [6] et. al. made a number of observations on using Condor to run *Zefsa II*:

- The Intel compiler provided a 30% speedup in execution. This performance requirement precluded the use of the standard universe, and left the application subject to preemption.
- Execution times were not well defined or regular. Some parameter sets may have taken less than 1 hour, but others took as long as 15 hours. The environment at Purdue nearly guarantees preemption, and adaptation of the workflow was performed to constrain execution times to approximately 1 hour.
- Data handling is an issue with any parameter sweep of this size: 7000 parameter sets will yield 700,000 output files. Sending 2000 jobs consistently to a networked filesystem will quickly identify problems with the scratch storage.

5. Conclusions

We are excited about the new possibilities opened by this Condor-based campus cyberinfrastructure: new

resources become available on a regular basis on the West Lafayette campus, and Purdue regional campuses and other institutions across Indiana are showing interest in utilizing Condor as a true low barrier-to-entry CI that can quickly benefit their users. Condor's flocking mechanism enables us to build a computing infrastructure as close to a transparent computing grid as anything in existence can today.

Combining the enormous productivity seen by users who effectively use Condor with the fact that Condor efficiently harvests underused resources, Condor is an excellent value for an enterprise. Condor leverages expenditures already made for equipment acquisition, cooling, power, and administration. With only a small initial investment of staff time and upkeep, a powerful resource can be created for high-throughput science.

When deploying an enterprise-scale Condor infrastructure, we recommend several best practices. Closely tracking the development of Condor's development series for production infrastructure, installing reasonably-sized pools, providing many *schedds* for users to submit jobs, but most of all, if users are going to use a shared filesystem instead of file transfer, the volume of work possible in Condor may require an enterprise-scale storage system.

6. Future Directions

Our experiences developing the industrial strength cyberinfrastructure described in this paper opens several new areas for further work. Two areas in particular, science gateway job submission wizards, and personalized monitoring tools for resources and job status, have the potential to improve the usability of large Condor installations.

In the first area, job wizards, many users learning to use Condor first inquire about which Condor *ClassAds*² can be used to select desired computers. In place of time spent by staff teaching everyone the intricacies of various ClassAd attributes and values specific to the system, a ClassAds wizard will help a user select a system through questions and answers.

The other area of future work, personalized monitoring, will provide users with new tools to support the ability to easily monitor the status of their jobs in place of logging into the system. Through recent funding from the NSF HPCOPS program, we plan to simplify the experience of using BoilerGrid by creating a portal to address both of these areas of future work.

²*ClassAds* are Condor's representation of any attribute describing characteristics of a computer or job.

Acknowledgments

We would like to acknowledge Sebastien Goasguen, now at Clemson University, for his guidance with this campus CI during its infancy. Miron Livny, Todd Tannenbaum, Alain Roy, Nick LeRoy, Alan DeSmet, Dan Bradley and the rest of the Condor Team have been invaluable in everything that we do. And special thanks to our users (Dr. Deem, Dr. Earl, Dr. Wen Jiang and students, Drs. Phil Cheeseman, Jeff Linderoth and many more) who have continually challenged us to raise the bar for what our Condor pools can enable them to do. And lastly, Gerry McCartney, Purdue's Vice President for Information Technology, without whose support we could never have succeeded to the degree that we have.

This work was supported in part by the National Science Foundation under TeraGrid Resource Partners grant OCI-0503992 and the National Science Foundation Compact Muon Solenoid Project PHY-0516857.

References

- [1] Altair corporate - innovation intelligence. [Online]. Available: <http://www.altair.com>
- [2] Bluearc - titan 2100, 2200, and 2500 - unified storage without compromise. [Online]. Available: <http://www.bluearc.com/>
- [3] Northwest indiana computational grid. [Online]. Available: <http://www.nwicgrid.org>
- [4] P. Canal, S. Borra, and M. Malani, "Gratia, an resource accounting system for osg," in *CHEP - Computing in High Energy Physics*, 2006.
- [5] C. E. Catlett, "TeraGrid: A foundation for US Cyberinfrastructure," in *NPC*, ser. Lecture Notes in Computer Science, H. Jin, D. A. Reed, and W. Jiang, Eds., vol. 3779. Springer, 2005, p. 1.
- [6] P. A. Cheeseman, M. W. Deem, D. J. Earl, and W. I. Whitson, "Adapting an application for use in a condor based parameter sweep on teragrid," in *TeraGrid '07*, June 2007.
- [7] D. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne, "A worldwide flock of Condors: Load sharing among workstation clusters," *Future Generation Computer Systems*, vol. 12, pp. 53–65, 1996.
- [8] M. Falcioni and M. W. Deem, "Zefsa ii: Zeolite framework solution." [Online]. Available: <http://www.mwdeem.rice.edu/zefsaII/>
- [9] S. L. Gooding, L. Arns, P. Smith, and J. Tillotson, "Implementation of a distributed rendering environment for the TeraGrid," in *Challenges of Large Applications in Distributed Environments, 2006 IEEE*, June 2006, pp. 13–21.
- [10] T. J. Hacker and B. C. Wheeler, "Making research cyberinfrastructure a strategic choice," *Educate Quarterly*, no. 1, 2007. [Online]. Available: <http://www.educause.edu/ir/library/pdf/eqm0713.pdf>
- [11] M. Livny, J. Basney, R. Raman, and T. Tannenbaum, "Mechanisms for high throughput computing," *SPEEDUP Journal*, vol. 11, no. 1, June 1997.
- [12] K. Madhavan, G. Bertoline, and L. Arns, "The design of a distributed rendering environment to support classroom instruction in animation and scientific visualization," *IEEE Computer Graphics and Applications*, vol. 25, no. 5, pp. 22–38, Sept/Oct 2005.
- [13] M. W. Mutka, "Sharing in a privately owned workstation environment," Ph.D. dissertation, University of Wisconsin-Madison, 1988.
- [14] A. Pacelli, "Moca: A 0/1/2-d monte carlo device simulator," *Moca User's Manual*, 1997.
- [15] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Würthwein, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, and R. Quick, "The open science grid," *Journal of Physics: Conference Series*, vol. 78, p. 012057 (15pp), 2007. [Online]. Available: <http://stacks.iop.org/1742-6596/78/012057>
- [16] M. Shuey, G. Veldman, C. Baumbauer, P. Finnegan, D. McKay, and S. Goasguen, "Local community clusters experiences with resource sharing in international and national grid infrastructure," in *2005 NSF CISE/CNS Infrastructure Experience Workshops*, 2005. [Online]. Available: http://www.cs.uiuc.edu/events/expwork-2005/Sebastian_Goasguen_Abstract.pdf
- [17] S. Son and M. Livny, "Recovering internet symmetry in distributed computin," in *Cluster Computing and the Grid*, ser. IEEE/ACM International Symposium on, 12 May 2003, pp. 542–549.
- [18] T. Tannenbaum. Condor roadmap - condor week 2007. [Online]. Available: http://www.cs.wisc.edu/condor/PCW2007/presentations/tannenba_roadmap_2007-v4.ppt
- [19] ——. What's new in condor? - condor week 2006. [Online]. Available: http://www.cs.wisc.edu/condor/CondorWeek2006/presentations/tannenba_roadmap_2006.ppt
- [20] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience." *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.
- [21] ——. "How to measure a large open-source distributed system: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 18, no. 15, pp. 1989–2019, 2006.