Campus Grids: Bringing Additional Computational Resources to HEP Researchers

Derek Weitzel¹, Dan Fraser², Brian Bockelman³, and David Swanson⁴
¹,³,⁴ University of Nebraska Holland Computing Center, 118 Schorr Center, Lincoln NE 68588, USA
² Argonne National Laboratory, 9700 S. Cass Ave., Bldg 240, Argonne, IL 60439, USA
E-mail: ¹dweitzel@cse.unl.edu, ²fraser@anl.gov, ³bbockelm@cse.unl.edu, ⁴dswanson@cse.unl.edu

Abstract. It is common at research institutions to maintain multiple clusters that represent different owners or generations of hardware, or that fulfill different needs and policies. Many of these clusters are consistently under utilized while researchers on campus could greatly benefit from these unused capabilities. By leveraging principles from the Open Science Grid it is now possible to utilize these resources by forming a lightweight campus grid. The campus grids framework enables jobs that are submitted to one cluster to overflow, when necessary, to other clusters within the campus using whatever authentication mechanisms are available on campus. This framework is currently being used on several campuses to run HEP and other science jobs. Further, the framework has in some cases been expanded beyond the campus boundary by bridging campus grids into a regional grid, and can even be used to integrate resources from a national cyberinfrastructure such as the Open Science Grid. This paper will highlight 18 months of operational experiences creating campus grids in the US, and the different campus configurations that have successfully utilized the campus grid infrastructure.

1. Introduction
A campus grid is traditionally made of numerous resources spread across a campus or institution. There are many reasons for multiple resources, such as different owners, policies, or even generations of hardware. A distributed set of on-campus resources can lead to uneven utilization of these resources. A campus grid can be used to unify these disparate resources into a coherent framework that can benefit High Energy Physics (HEP) and other users.

Disparate owners can cause a campus to have a multitude of resources. The owners want to control access and give priority to their users. It is important for owners to implement their own policies. In many cases, the resources can be underutilized by the owners, while others may be oversubscribed. A campus grid could overflow jobs between the clusters to better utilize the resources.

Computing centers traditionally purchase resources in generations, keeping each purchased resource as an independent cluster rather than adding it to an existing resource. This partitioning makes it difficult for users to efficiently submit jobs to idle resources. Further, it can lead to uneven utilization as most users will want to use the new hardware for potential speed increases. Or, users could stay on an older cluster for sense of reliability. Either way,
users who could potentially use both clusters are left with few options to split their workflows between clusters.

Campus grids come in many sizes and configurations. Some campus grids are based around a single technology that makes interconnection of resources easy. This single technology could be a common batch system, such as Condor \cite{1}. Or the single technology could be a shared filesystem such as AFS. Wisconsin implements both of these technologies in order to enable their campus grid.

Most campuses do not have these shared technologies. A more common type of campus grid is a series of resources that do not share a file system and have distinct batch systems. Distinct batch systems are used to implement policies for the resources that reflect the owners preferences. The batch systems might also be wholly incompatible with each other, such as the Portable Batch System and Condor. In this campus grid, no easily available solution to combine these resources was available.

High Energy Physics is only one of many sciences that are run on campus grids. At Virginia Tech, researchers are running complex system simulations \cite{2} on their campus grid. At the University of Wisconsin - Madison, researchers are running protein folding applications. The Nebraska campus grid runs many applications such as protein searching \cite{3}.

Each of these applications have very similar requirements:

- A similar environment between the submit host and the executing host
- Ability to stage data between the execute host and remote storage resources.

Traditionally, these requirements have been met with custom solutions, such as software installed on a shared drive, AFS \cite{4} at Wisconsin, or the more general solution, software can be sent with the job. Though shipping software with the job does allow the job to expand outside of the campus, it is rarely scalable for even moderately sized applications. Software is usually treated the same as data on campus grids, even though they are fundamentally different in both the access patterns and the data sizes. We will treat software and data as separate entities, allowing for greater flexibility in submission and execution of the workflows.

The contributions of this paper are the creation and demonstration of the BLAHP Over SSH Condor Overlay (BOSCO) system, and the demonstration of its use for campus High Energy Physics research. The BOSCO software simplifies the addition of resources to the campus computing grid. Further, it provides a consistent interface for the user whether the job is executing locally or anywhere on the campus. Another contribution is made by enabling transparent software and data availability on remote resources just as on the local system.

The BOSCO package was designed to transparently expand outside of the campus, either to other campuses or to the Open Science Grid \cite{5}. We therefore have an expanding circle of resources available to the campus researcher. At each level, the amount of resources increases, as does the level of trust that a researcher must have. At the local level, the researcher is allowed access to the local resources. At the campus level, the resource manager on campus must trust the researcher to run on their resources. At the grid level, a higher level of trust is required, therefore a user needs a grid certificate for access to resources. Figure 1 shows the increasing resources at each level of trust.

The next section will discuss the OSG Campus Grid Architecture and how it is used by campus researchers. Section 3 will cover the experience of running CMS jobs through the campus grid. Finally, the Section 4 will discuss our conclusions and possible opportunities for future work in this direction.

2. Campus Grid Architecture
The goal of a campus grid is to enable easy access resources. To this end, we designed the campus grid solution with these goals:
Transparent expansion of computing resources
- Environment consistency from submit host to execute host
- Simple installation and operation

The software and architecture of the campus grid reflects these goals. The BOSCO software toolkit includes the software:
- Condor
- BLAHPD
- Campus Factory

Additionally, for HEP experiments we used the Parrot Virtual File System [6] from University of Notre Dame’s Cooperative Computing Tools and XRootD [7].

2.1. Condor and BLAHPD
Condor is used as the batch system and user interface for the job execution in BOSCO. When the user submits jobs the job will be managed by Condor. Job execution on a local Condor pool is unchanged from default behavior. The job will be scheduled and then execute on the attached Condor pool. When the job completes, output is transferred back to the submit host via condor file transfer mechanisms.

Condor has recently added the ability to submit jobs directly to PBS[8], LSF[9], and SGE[10] schedulers. This was done by integrating the Batch Local ASCII Helper Protocol Daemon (BLAHPD) from INFN. It translates a Condor job into a PBS, LSF, or SGE job. It also communicates the jobs status back to Condor.

Condor has very recently added the ability to submit jobs through SSH using a BLAHPD installed on the remote node. Submitting jobs to remote clusters through SSH has many advantages. SSH is a universal method of accessing remote clusters. Most, if not all, campus resources are accessible for interactive login via SSH. Second, SSH has built-in non-interactive features that can be used for remote submission, similar to what is used by many applications such as rsync and scp. Further, remotely executed commands through SSH can provide standard output (stdout) back to the originator and accepts standard input (stdin), the same protocol that is used by Condor to interact with its other remote job managers such as those for globus submissions [11].
In order for Condor to submit through SSH, the user must setup the remote node to accept the connection. This is done with the `bosco_cluster` command. This command will setup the passwordless ssh keys on the local and remote systems. It will copy over the necessary components of the BLAHPD to the remote node as well.

When a user submits a job requesting a SSH job submission, Condor will first start the SSH connection to the remote cluster. The remote connection will start the SSH connection, and begin communicating with the remote BLAHPD. The remote BLAHPD is unmodified and communicates just as before, using stdout and stdin as the interface to the remote submitting Condor. The jobs’ executables are assumed to be on the remote system as the current version of BOSCO does not contain SSH file transfer. The BLAHPD will submit the jobs on behalf of the user, and continue to monitor the status and report to the submitting Condor.

### 2.2. Campus Factory

The Campus Factory is a condor overlay generator designed explicitly for use on campuses. The factory packages and submits Condor worker node executables (glideins) to be run inside other batch systems. This is especially useful when a Condor user interface is desired when interfacing with another batch system.

The Campus Factory queries the users queue for idle jobs. If idle user jobs are detected, it will submit glideins to the local condor queue as type: universe = pbs. This directive indicates that Condor should use the blahp to submit jobs directly to the local PBS queue. The job, once started by PBS on a worker node, will unpack the glidein and start the daemons. Once the daemons start, they will register with the Condor instance running beside the campus factory and be available to run jobs. The campus factory also implements glidein submission policies, such as, only five glideins should be idle in queue at a time.

### 2.3. The BLAHPD Over SSH Condor Overlay

Putting together Condor, BLAHPD, and the Campus Factory, we get the BLAHPD Over SSH Condor Overlay (BOSCO) toolkit. This package is able to provide a consistent Condor interface to the user by submitting glideins to remote resources through SSH. The Campus Factory is integrated into the Condor install as a contrib package that is installed and started with Condor.

The user installs BOSCO with the package available on the OSG website. It contains all of the software listed above. The user then configures it with the addresses of the remote hosts using the `bosco_cluster` command used in Condor. The `bosco_cluster` command is modified to also send the necessary campus factory components to the remote cluster. The user then submits their jobs to Condor just as they would a local Condor pool.

The Campus Factory picks up the idle user jobs and begins submitting glideins to the remote hosts configured with `bosco_cluster`. Condor handles the submission as a remote SSH BLAHPD submission. The glidein executables that were sent with the `bosco_cluster` command are run as the PBS job. When the glidein starts, it registers with BOSCO and begins running the users jobs. The Campus Factory will detect the newly registered glideins, and submit more if more idle jobs are detected.

BOSCO provides a Condor only interface to many different types of batch systems. This provides the advantage of only one interface for the user to learn. Additionally, the user may use unique condor features such as robust file transfer, fine grained matchmaking, and workflow management to optimize their job execution.

The BOSCO package is designed to be run as a regular user. This is so that scientists can run it on existing resources without administrator intervention. Also, the remote BOSCO daemon runs as the user on the remote resource, requiring an administrator to give only SSH access to the cluster to enable BOSCO submissions.
Figure 2. Flow of the Campus Factory from job submission by the user, to remote submission of the glideins, and finally the execution of the user’s jobs on the remote execute nodes.

2.4. Additional Software For HEP

Parrot and XrootD are additional packages not distributed with the BOSCO package. They provide specific features that are very useful for large software distributions (Parrot) and large data sets (XrootD).

The Parrot Virtual File System provides a user space transparent file system. It works by intercepting open, read and write system calls and redirecting them to remote file systems. In our case, we use Parrot to access CernVM-FS [15] file system hosted at the University of Wisconsin – Madison.

CernVM-FS is a HTTP based file system developed for the Worldwide LHC Computing Grid collaboration. As a HTTP filesystem, is has many advantages for use in distributed computing. For example, it can be easily cached on remote systems by using HTTP caches, which are widely deployed on OSG, and are easily deployable on campus grids.

XrootD is an application and protocol for file access. It uses a tree structure in order to optimally transfer data from a distributed set of resources. A user first forms a request for a file and sends it to an XrootD redirector. The redirector then queries all of the registered servers it knows of to ask if they have the file. The registered servers may be other redirectors that in turn ask further servers regarding the existence of the file. If it reaches a server that has the file, the server will send a positive response to the redirector, which in turn returns the address of the server to the user. In our instance, we use the USCMS Federated XrootD implemented as part of the NSF funded Any Data, Anytime, Anywhere [16] project. This project federates all USCMS Tier-2s and the USCMS Tier-1 into a coherent file namespace. This allows users to access any CMS file that is located in the US through XrootD.

The CMS software framework, CMSSW [17], has integrated XrootD for remote file access. The default configuration at most USCMS Tier-2 sites is to use local storage first, then fall back to the Federated XrootD if the local site fails to find the file. This is useful if the local site is missing a file, if the file is corrupted, or if the user purposively submits their jobs to sites that do not have the files they requires. A user could do this in order to access more computing
resources rather than waiting on a busy site that may have their sites. Our implementation forces the application to use the XrootD for data access.

3. Experience with Campus Grids

In this section, we will discuss our experiences in creating and maintaining campus grids at various universities in the U.S.

Several campus grids have been established using the OSG campus grid initiative. Not all of the campuses use the architecture laid out in the previous section. Some mix and match components in order to better fit their policies. Here, we will highlight some of the campus grid configurations that we have helped establish.

The Sunshine grid is an emerging state grid in Florida. The collaboration of universities extends across the state. As part of the prototype, a campus grid was established between University of Florida and Florida International University to run custom CMS analysis jobs. This simple campus grid consisted of a small Condor pool at FIU communicating with the Tier-2 and another cluster at the University of Florida, both based on Condor. Since both sides are Condor, we configured its built in job distribution technique of flocking. Flocking will send jobs to remote Condor pools for execution when the local pool is unable to meet demand.

The Virginia Tech campus grid is based out of the Virginia Bioinformatics Institute [18]. The mission of the institute is: Modeling and simulation of interaction-based, co-evolving technical, biological, and social networks in support of national policy. The institute uses a web based portal for job submission of pre-defined applications on their current HPC resources.

The institute's primary application is a parallel application that scaled to small numbers of cores. They wanted to use High Throughput Parallel Computing (HTPC) to run their applications, a method of allocating a whole node, usually 8 cores, to a single parallel application. This required a special configuration of the blahpd in order to request whole node. The campus grid consists of two clusters, each running an instance of the Campus Factory which flocks with a central submit host. When the campus grid is unable to provide enough resources, the jobs would go out to the OSG to run on HTPC compatible resources.

3.1. Evaluation of CMS on the Nebraska Campus Grid

The Nebraska CMS Tier-3 is connected and executes jobs on the local Tier-2. Even though the Tier-2 is large, the Tier-3 is nearly always using its full allocation. Therefore, users of the Tier-3 would like to expand access to other on campus resources. Nebraska has numerous local general purpose clusters that the Tier-3 users have access, but no reliable method of sending jobs.

At the University of Nebraska, we created a campus grid that included users of our CMS Tier-3. Figure 3 shows the campus grid with the Tier-3. Submission to the Tier-3 is done using CRAB, which in turn submits the job to Condor. We then configured the Tier-3 to flock to our campus submit host running BOSCO. The submit host is running the Campus Factory which detects idle jobs in the remote Tier-3, and submits glideins to a PBS cluster to meet demand. The glideins start on the PBS cluster and report to the BOSCO submit host, which then allocates their usage to the Tier-3.

CMSSW requires a large software install. On the Nebraska CMS Tier-2, the cms software install is 312GB and 11 million files. The software is much too large to install on each campus resource, and too large to send with each job. Therefore, we must stream in the software using the Parrot Virtual File System. We configured parrot to connect to the UW-Madisons CernVM-FS installation which includes the CMS software. Parrot will cache the files retrieved from CernVM-FS using HTTP proxies located at Nebraska. It will also cache files locally in the machines temporary directory.

When the glidein starts a job, it will begin the job using parrot so that the executables system calls can be captured by it. The CMS jobs will look for CMS software using the environment
variable $\text{OSG\_APP}$. Using this convention, we modified the jobs starting environment to point $\text{OSG\_APP}$ to the location that parrot would redirect to CernVM-FS.

For input data, we used XrootD built into the cms software. The CMS jobs will look for data in the order specified in a special site configuration file. For our campus grid, the file listed the Federated XrootD redirector as the source for input data. XrootD would then stream data into the job transparent to the user.

We successfully ran jobs from our Tier-3 to other non-CMS cluster on our campus using the BOSCO infrastructure. The user was unaware that they ran on the campus grid, an indication that our goal of transparent execution was successful.

4. Conclusions and Future Work
We plan to turn the Nebraska Tier-3s connection to the campus grid into a production service. The plans include an automatic overflow when jobs have been waiting in queue over a threshold time.

BOSCO is being actively developed for a release in the official Condor packages. The Campus Factory is being prepared to be integrated into Condor as a contrib module. Integration of these components with upstream Condor will ensure future updates will be less likely to break functionality.

As part of the OSG mission, we are committed to new user and campus engagement. To this end, we are always looking for new users for our campus grid software and techniques. Each campus is different, with varying needs and policies. Therefore, our software needs to flexible in order to be useful on campuses. Possibly even more important than the software, we share our expertise in distributed computing with new campus users in order to maximize their science. Our eventual goal is to generate new campus and OSG users through this engagement.

In this paper, we described the implementation of a campus grid emphasizing how HEP users can utilize campus grids. We described a prototype grid constructed at the University of Nebraska for local Tier-3 users.

Campus grids are the natural extension of local computing. A user can expand incrementally,
from the local cluster, to the local campus, and finally to the grid. Each level requires more trust of the user, but the user is given more resources to use. The BOSCO toolkit makes creation of a campus grid easy and transparent to end users. Campus grids are a natural gateway to the national cyberinfrastructure.

5. Acknowledgements
This research was done using resources provided by the Open Science Grid, which is supported by the National Science Foundation and the U.S. Department of Energy’s Office of Science.

This work was completed utilizing the Holland Computing Center of the University of Nebraska.

References
[1] Litzkow M, Livny M and Mutka M 1988 Condor—a hunter of idle workstations Distributed Computing Systems, 1988., 8th International Conference on (IEEE) pp 104–111
[2] Bisset K R, Chen J, Feng X, Ma Y and Marathe M V 2010 INDEMICS: An interactive data intensive framework for large scale epidemic simulation ICS ’10: Proceedings of the 24th ACM International Conference on Supercomputing (New York, NY, USA: ACM) pp 233–242 ISBN 978-1-4503-0018-6 {http://doi.acm.org/10.1145/1810085.1810118}
[3] Powers R, Copeland J, Stark J, Caprez A, Guru A and Swanson D 2011 BMC research notes 4 17
[4] Howard J and Center C M U I T 1988 An overview of the andrew file system (Carnegie Mellon University, Information Technology Center)
[5] Pordes R, Petrivick D, Kramer B, Olson D, Livny M, Roy A, Avery P, Blackburn K, Wenaus T, Würthwein F et al. 2007 The open science grid Journal of Physics: Conference Series vol 78 (IOP Publishing) p 012057
[6] Thain D and Livny M 2005 Scalable Computing: Practice and Experience 6 9–18
[7] Dorigo A, Elmer P, Furano F and Hanushevsky A 2005 WSEAS Transactions on Computers 1
[8] Adaptive Computing Torque resource manager http://www.adaptivecomputing.com/products/ open-source/torque/
[9] Platform Computing High performance computing http://www.platform.com/workload-management/ high-performance-computing
[10] Oracle Oracle grid engine http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html
[11] Frey J, Tannenbaum T, Livny M, Foster I and Tuecke S 2002 Cluster Computing 5 237–246
[12] Sfiligoi I 2007 Making science in the grid world: using glideins to maximize scientific output Nuclear Science Symposium Conference Record, 2007. NSS’07. IEEE vol 2 (IEEE) pp 1107–1109
[13] Raman R, Livny M and Solomon M 1998 Matchmaking: Distributed resource management for high throughput computing High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on (IEEE) pp 140–146
[14] Frey J 2002 University of Wisconsin, Dept. of Computer Science, Tech. Rep
[15] Buncic P, Aguado-Sanchez C, Blomer J and Harutyunyan A 2011 Cernvm: Minimal maintenance approach to virtualization Journal of Physics: Conference Series vol 331 (IOP Publishing) p 052004
[16] Ian Bird, Frank Würthwein Any data, anywhere, anytime white paper http://osg-docdb.opensciencegrid. org/cgi-bin/ShowDocument?docid=1033
[17] Innocente V, Silvestris L and Stickland D 2001 Computer Physics Communications 140 31–44
[18] Virginia bioinformatics institute http://www.vbi.vt.edu/