Contributing opportunistic resources to the grid with HTCondor-CE-Bosco

Derek Weitzel, Brian Bockelman
University of Nebraska – Lincoln Holland Computing Center, US
E-mail: dweitzel@cse.unl.edu

Abstract. The HTCondor-CE [1] is the primary Compute Element (CE) software for the Open Science Grid. While it offers many advantages for large sites, for smaller, WLCG Tier-3 sites or opportunistic clusters, it can be a difficult task to install, configure, and maintain the HTCondor-CE. Installing a CE typically involves understanding several pieces of software, installing hundreds of packages on a dedicated node, updating several configuration files, and implementing grid authentication mechanisms. On the other hand, accessing remote clusters from personal computers has been dramatically improved with Bosco: site admins only need to setup SSH public key authentication and appropriate accounts on a login host. In this paper, we take a new approach with the HTCondor-CE-Bosco, a CE which combines the flexibility and reliability of the HTCondor-CE with the easy-to-install Bosco. The administrators of the opportunistic resource are not required to install any software: only SSH access and a user account are required from the host site. The OSG can then run the grid-specific portions from a central location. This provides a new, more centralized, model for running grid services, which complements the traditional distributed model. We will show the architecture of a HTCondor-CE-Bosco enabled site, as well as feedback from multiple sites that have deployed it.

1. Introduction

In order for a resource to join the Open Science Grid (OSG) [2], the first step is traditionally to install a Compute Element (CE). The CE’s role is to accept pilots from an outside source, authenticate and authorize the pilot, and to transform the outside pilot into one that the local resource manager can understand. The HTCondor-CE [1] accomplishes this by:

- **Accept Pilots**: Use the HTCondor protocol CEDAR [3] to accept pilots submitted from remote resources.

- **Authenticate and Authorize**: Authentication is performed by using X.509 GSI [4] certificates. Authorization is performed by a callout to LCMAPS [5] which in turn can use either a flat file of authorized users and mappings, or communicate with the GUMS [6] authorization and mapping service.

- **Transformation**: Pilot transformation is performed with the BLAHP [7].

The HTCondor-CE has proven itself and is used in production across the OSG. The HTCondor-CE-Bosco builds on top of the technologies and techniques of the HTCondor-CE. It uses the same HTCondor protocol to accept pilots. The authentication and authorization is the same as the HTCondor-CE. The pilot transformation uses the BLAHP as well, but instead of
running the BLAHP on the same node as the CE, it installs and starts the BLAHP on a remote
node and submits to the remote resource manager.

Installing an HTCondor-CE is time consuming and complicated. First, the HTCondor-CE
requires an X.509 host certificate for the CE. The process for receiving a X.509 requires multiple
steps, with several human interventions. In contrast, for a HTCondor-CE-Bosco, all that is
required of the remote cluster is a user account for pilot submissions and support for Secure
Shell (SSH) public keys. No host certificates are required for the remote cluster, instead they
are installed on the HTCondor-CE-Bosco by the grid administrators. Host certificates can be
allocated by existing grid professionals much easier.

Second, Since the HTCondor-CE-Bosco submits pilots to a remote cluster over SSH, the
remote cluster does not require a CE to be installed. Instead, a HTCondor-CE-Bosco can be
hosted centrally by grid professionals which can submit pilots to the remote cluster.

The advantages of using the HTCondor-CE-Bosco over a regular CE are:

- Can be centrally managed by grid professionals
- No certificates required at the institutions
- Only requirements for a cluster to run grid pilots is access to a user account that can submit
to the local resource manager, and SSH pubkey access to that account.

2. Background

Bosco [8] is a set of technologies packaged together to ease submission of pilots to remote
resources. Bosco adds remote submission to remote resource managers through SSH. Submission
of pilots over SSH is the innovation that makes Bosco easy to use and install. The Bosco
architecture is shown in Figure 1.

![Bosco Architecture](image)

**Figure 1.** Bosco Architecture

The user first submits jobs to the HTCondor installation on the Bosco Submit Host. The
job has special attributes in the submit file that will specify where the job should be submitted.
Figure 2 shows the special lines for the submit file required for Bosco submission. The second
argument is the type of remote resource manager that Bosco will submit. The third argument
is the ssh endpoint to use when submitting the job.

HTCondor will use the `grid_resource` line to ssh into the remote cluster’s login node where
it will start the BLAHP. The BLAHP communicates with the scheduler in order to submit jobs.
The BLAHP will translate the job attributes into the resource manager specific language to
universe = grid
grid_resource = batch slurm user@example.university.edu
executable = test.sh
output = output
derror = error
queue

Figure 2. Example HTCondor job submission for Bosco

submit the job. HTCondor will also transfer files from the submit host to the login host over the same SSH connection.

The BLAHP is a set of executables that translate HTCondor submission into a submission to a specific resource manager. The BLAHP receives commands from HTCondor, which directs the BLAHP to submit a job to a particular resource manager with specific job submission attributes.

HTCondor on the Bosco submit host will query the BLAHP for job status periodically. Once the job completes, the output files will be transferred back over the SSH connection to the submit host.

In this paper, we will discuss the HTCondor-CE-Bosco and how it creates an easy on-ramp for new sites. In Section 3, we will discuss the implementation of the HTCondor-CE-Bosco. In Section 4, we will present feedback from institutions using the HTCondor-CE-Bosco. Also, we will highlight the flexibility of the HTCondor-CE-Bosco by illustrating the architectures these institutions have chosen to use.

Finally, we will conclude in section 6 will discuss our ideas for future work with the HTCondor-CE-Bosco. In section 7 we will list the contributions of the HTCondor-CE-Bosco to making an easy on-ramp for new resource contributors.

3. Implementation

3.1. Installation & Configuration

As noted before, the HTCondor-CE-Bosco does not need to be installed on every cluster. Instead, it can be installed centrally and submit pilots to the remote clusters. Since the HTCondor-CE-Bosco is installed centrally by grid professionals, the barrier to getting X.509 certificates and a host is much lower than if each small institution had to install an HTCondor-CE.

Configuration is completed with the osg-configure [9] tool. The system administrator fills out the configuration file, filling in:

- **users**: What users should be able to submit to the remote cluster. During the authorization step, a remote user is mapped to a local user, which must be listed here in order to submit pilot to the remote cluster.

- **endpoint**: The SSH compatible endpoint which Bosco should use to submit pilots. This is in the form of <username>@<hostname.example.com>.

- **batch**: Which operating system to use on the remote cluster. Values could be pbs or slurm.

- **ssh_key**: The location of the SSH key that can be used to access the remote cluster.

Once the configuration file is completed, the administrator runs the osg-configure command to configure HTCondor-CE-Bosco. OSG-Configure will ssh into the remote cluster and install the cluster side client. It will also configure the HTCondor-CE to route pilots to the remote cluster when they are submitted.
3.2. Running the HTCondor-CE-Bosco

The HTCondor-CE-Bosco uses Bosco [8] technology to submit pilots to remote clusters over SSH. The Bosco architecture is shown in Figure 1. The HTCondor-CE-Bosco host acts as the Bosco Submit Host.

![Figure 1. HTCondor-CE-Bosco architecture](image)

When a pilot is received at the HTCondor-CE, it is entered into the queue. A process called the JobRouter transforms the pilot from a generic OSG pilot to a customized pilot for the cluster. These changes could be setting the maximum memory allowed for a pilot. Or the JobRouter could change the queue for which the BLAHP should use to submit pilots on the remote cluster. A diagram of the flow of the pilot is show in Figure 3. The translation parameters are configured as expressions in the HTCondor-CE configuration. An example configuration is shown in Figure 4.

```plaintext
[ 
    GridResource = "batch slurm osg@example.university.edu"; 
    TargetUniverse = 9; 
    name = "Local_BOSCO_example"; 
]
```

![Figure 4. Example configuration for a Slurm HTCondor-CE-Bosco](image)

The pilot transformation configuration shown in Figure 4 is stored on the HTCondor-CE-Bosco host. It is created by the `osg-configure` tool depending on the configuration given to it. It is maintained by the grid administrators since it lives on the HTCondor-CE-Bosco.

Once the pilot has been transformed, the JobRouter will resubmit the pilot to the HTCondor-CE with the transformed attributes. The HTCondor-CE will then start the Bosco process of SSH'ing into the remote cluster and submitting the pilots to the remote resource manager.

Once the pilot is submitted, the HTCondor-CE-Bosco will query for the pilot’s status every minute. The status changes will be reflected in the queue as the pilot goes from idle to running, and finally to completed. Once the pilot is completed, the output files will be transferred back to the HTCondor-CE-Bosco host.

Figure 4 shows one simple configuration. Pilots submitted to the HTCondor-CE-Bosco from different users will all be routed to the same user account on the remote machine. This can be useful if the remote cluster has only a single account set aside for grid pilot. In Section 4, we discuss institutions that wanted multiple cluster users, and how this can be enabled in the HTCondor-CE-Bosco.
In order to gather accounting data from a traditional HTCondor-CE, a scheduler specific probe must be installed that gathers pilot completion details. The OSG maintains Gratia [10] probes for different batch systems, HTCondor, SLURM, PBS, and SGE. Since the scheduler is remote, the HTCondor-CE-Bosco cannot use any of these Gratia probes for accounting. It uses a single Gratia probe originally designed for the HTCondor system that can gather completion details from HTCondor-CE logs. This new probe does not require access to the remote resource manager.

The HTCondor Gratia probe had to be altered in order to work with an HTCondor-CE-Bosco. Since the each pilot that is submitted to the HTCondor-CE-Bosco is transformed into another pilot, as seen in Figure 3, there are 2 pilot in queue for every 1 pilot submitted to the remote cluster. In order to avoid duplication of reporting, only the transformed pilot is used for accounting, while the other is ignored.

4. Feedback from users

4.1. University of Utah

The University of Utah is one first users of the HTCondor-CE-Bosco for their clusters. They were attracted to it because they had multiple clusters they wanted to connect to the OSG, but didn’t want multiple CE’s. With the HTCondor-CE-Bosco, they are able to load balance 4 different clusters. Figure 5 shows the architecture of Utah’s HTCondor-CE-Bosco. Since Utah wanted ultimate control of the CE’s, they chose to run the HTCondor-CE-Bosco at Utah.

In addition to running 4 different clusters load balanced from a single HTCondor-CE-Bosco, Utah was in the process of upgrading from RHEL 6 to RHEL 7. Therefore, the pilots also need to be routed to different clusters depending on the pilot’s OS choice (binaries that work on RHEL 7 are not guaranteed to run on RHEL 6). To route the pilots, Figure 6 shows an example routing configuration for the HTCondor-CE-Bosco.

Note, updated Requirements. When a pilot is submitted to the CE, it has a special attribute named distro. That attribute can either be “RHEL6” or “RHEL7”. The routes only match pilots which have the same distro value as the expression in the Requirements. The first entry
Figure 6. Utah’s Routing Configuration

```plaintext
GridResource = "batch slurm osg@example.university.edu";
TargetUniverse = 9;
name = "Local_BOSCO_example";
Requirements = distro == "RHEL7";
```

```plaintext
GridResource = "batch slurm osg@el6.university.edu";
TargetUniverse = 9;
name = "Local_BOSCO_el6";
Requirements = distro == "RHEL6";
```

Figure 7. Walltime Hours managed by HTCondor-CE-Bosco CEs

shows a route which will only match pilot with distro set to RHEL7; the second will only match pilot with distro set to RHEL6.

These requirements can be used to route pilots depending on any attribute in the pilot’s ClassAd [11]. This includes the submit source, the submitting user’s certificate, and many other attributes.

The example show in Figure 6 illustrates the flexibility of the routing configuration in the HTCondor-CE-Bosco. Not only can it load balance between multiple clusters, but it can also route OS specific pilots.

4.2. Stanford University

Stanford University uses the HTCondor-CE-Bosco in order to contribute resources to the OSG. Stanford uses a central installation managed by the University of Chicago, unlike Utah. The HTCondor-CE-Bosco is managed by OSG staff who are able to easily debug issues when they arise.
5. Results
The HTCondor-CE-Bosco is a major contribution to helping opportunistic and small resources contributed resources to the OSG. There continues to be active development on it. As seen in Figure 7, the HTCondor-CE-Bosco has managed over 1.9 million CPU hours in the last 6 months.

5.1. Accounting
Accounting with the HTCondor-CE-Bosco relies on how much information is gathered by the BLAHP from the batch system when a job completes. Some schedulers, such as Slurm, the scheduler keeps a database of completed job with job statistics. BLAHP is able to query this database to get detailed statistics of completed jobs.

Other schedulers do not have this job completion information, or provide this information in a more obscure way. For example, in a Torque (PBS) installation, completion information is contained in a log file that is constantly appended. This log file is on the head node running the scheduler, and not available to other nodes without special setup. Since the HTCondor-CE-Bosco can only view the remote login node, this log file is unavailable.

6. Future Work
6.1. Scaling
HTCondor-CE-Bosco has challenges in scaling to the same level as the “traditional” HTCondor-CE.

Each pilot submitted requires at least 1 input file, and 2 output files to be transferred. The minimum input file to transfer to the cluster is the executable that will run on the remote cluster. The output files are the stdout and stderr of the pilot. Additional input and output files can be specified by the submitter.

Every transfer from the HTCondor-CE-Bosco to the remote cluster is over the same shared connection. Therefore, we are limited by how much bandwidth is available between the HTCondor-CE-Bosco and the login host. Additionally, the HTCondor-CE-Bosco could be hosted off-site (like in the case for Stanford) increasing the likelihood of a network bottleneck between the two hosts. Anecdotal, the login host of a cluster is rarely optimized for network bandwidth.

Further, each pilot’s status is checked every minute. For a large number of tasks, this can impose a heavy load on the remote cluster’s scheduler. In order to alleviate some of this load, we have implemented caching inside the query scripts. But, when the number of pilots gets large enough, cache lock contention occurs. Possible solutions to this are based on rate limiting. A mitigation of this issue is to configure the HTCondor-CE-Bosco to only send a few status query requests at a time to the BLAHP.

6.2. Accounting
The accounting can be improved by using data from the most recent pilot status update. The status of a pilot is checked approximately every 1 minute. Along with checking if the pilot is idle, running or completed, we gather statistics from the scheduler such as how much resources are being consumed by the pilot. We can use the last status previous to the pilot completing to fill in detailed information about the pilot with minimal data loss. Unfortunately, this method is not able to gather the exit code of the process.

7. Conclusions
HTCondor-CE-Bosco can help opportunistic and smaller sites to easily contribute resources. It allows grid administrators within OSG Operations manage the HTCondor-CE-Bosco on behalf of the small site. Stanford has contributed almost 1 million hours to the OSG in the 6 months ending in January 2017 using the hosted HTCondor-CE-Bosco operational model.
We have also shown that the HTCondor-CE-Bosco is flexible, allowing Utah to distribute pilots to many clusters on their campus. Though, this level of control requires the site to run the HTCondor-CE-Bosco. It can allow sites that have different operating systems, or different architectures, to share the same HTCondor-CE.

Acknowledgments
This work was supported by NSF award PHY-1148698, via subaward from University of Wisconsin-Madison. This research was done using resources provided by the Open Science Grid [2], 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. The support and resources from the Center for High Performance Computing at the University of Utah are gratefully acknowledged.

8. References
[1] Bockelman B, Cartwright T, Frey J, Fajardo E, Lin B, Selmeci M, Tannenbaum T and Zvada M 2015 Commissioning the htcondor-ce for the open science grid Journal of Physics: Conference Series vol 664 (IOP Publishing) p 062003
[2] Pordes R, Petravick D, Kramer B, Olson D, Livny M, Roy A, Avery P, Blackburn K, Wenaus T, Frank W et al. 2007 The open science grid Journal of Physics: Conference Series vol 78 (IOP Publishing) p 012057
[3] Thain D, Tannenbaum T and Livny M 2005 Concurrency and computation: practice and experience 17 323–356
[4] Welch V, Siebenlist F, Foster I, Bresnahan J, Czajkowski K, Gawor J, Kesselman C, Meder S, Pearlman L and Tuecke S 2003 Security for grid services High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on (IEEE) pp 48–57
[5] Allieri R, Cecchini R, Ciaschini V, Gianoli A, Spataro F, Bonnassieux F, Broadfoot P, Lowe G, Cornwall L, Jensen J et al. 2003 arXiv preprint cs/0306004
[6] Lorch M, Kafura D, Fisk I, Keahey K, Carcassi G, Freeman T, Peremutov T and Rana A S 2005 Authorization and account management in the open science grid Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing (IEEE Computer Society) pp 17–24
[7] Weitzel D J 2015
[8] Weitzel D, Sfiligoi I, Bockelman B, Frey J, Wuerthwein F, Fraser D and Swanson D 2014 Accessing opportunistic resources with bosco Journal of Physics: Conference Series vol 513 (IOP Publishing) p 032105
[9] Grid O S 2017 Configuration scripts for osg software installs URL https://github.com/opensciencegrid/osg-configure
[10] Canal P 2011 Gratia: new challenges in grid accounting Journal of Physics: Conference Series vol 331 (IOP Publishing) p 062028
[11] Thain D, Tannenbaum T and Livny M 2003 Grid computing: Making the global infrastructure a reality 299–355