Accessing opportunistic resources with Bosco

D Weitzel1, I Sfiligoi2, B Bockelman1, J Frey3, F Wuerthwein2, D Fraser4, D Swanson1

1 University of Nebraska - Lincoln Holland Computing Center, 1101 T St, Lincoln NE 68588, USA
2 University of California, San Diego, 9500 Gilman Dr, San Diego, CA 92093, USA
3 University of Wisconsin, Madison, WI, USA
4 Argonne National Laboratory, 9700 S. Cass Ave., Bldg 240, Argonne, IL 60439, USA
E-mail: dweitzel@cse.unl.edu

Abstract. Bosco is a software project developed by the Open Science Grid to help scientists better utilize their on-campus computing resources. Instead of submitting jobs through a dedicated gatekeeper, as most remote submission mechanisms use, it uses the built-in SSH protocol to gain access to the cluster. By using a common access method, SSH, we are able to simplify the interaction with the cluster, making the submission process more user friendly. Additionally, it does not add any extra software to be installed on the cluster making Bosco an attractive option for the cluster administrator. In this paper, we will describe Bosco, the personal supercomputing assistant, and how Bosco is used by researchers across the U.S. to manage their computing workflows. In addition, we will also talk about how researchers are using it, including an unique use of Bosco to submit CMS reconstruction jobs to an opportunistic XSEDE resource.

1. Introduction

Opportunistic resources are required by experiments that have exhausted their owned resources but continue to need more. Examples of this are the high energy physics experiment the Compact Muon Solenoid (CMS) [1]. Opportunistic resources are also required when experimenters have little or no owned computing. In each of these cases, the experiments must reach out to organizations beyond their administrative domains to ask for the donation of resources. In many circumstances, organizations are willing to donate idle resources to other experiments, as long as it does not incur a significant investment in time or people.

Bosco [2] is used to effortlessly create a remote submission endpoint on a cluster, without the administrator installing any software. A High Performance Computing (HPC) cluster is optimized for large parallel scientific applications which spread processing across multiple nodes and require efficient communication between running applications. High Throughput Computing (HTC) cluster is optimized for multiple independent and process running on each node, with little or no communication required between each running application. Further, an HTC cluster is optimized for fault tolerance in the case of application or hardware failures. Bosco can turn a HPC cluster into a HTC cluster by overlaying HTC semantics onto the local HPC scheduler. It is built upon the resilient distributed computing software HTCondor [3]. Research groups, such as CMS, have used Bosco’s ability to connect remote resources in order to utilize opportunistic resources.
Improving the user experience was a primary goal of Bosco. We addressed the user experience by improving the interaction with the user during the installation / configuration. Another problem area we found is when a user must debug issues with distributed software. In order to address this, we created a traceroute [4] like utility. The traceroute utility tests every step of the job submission process, from network access to a properly configured remote scheduler.

The rest of this paper is organized as followed: Section 2 describes the architecture of Bosco, including a detailed look at the unique traceroute utility used for debugging. Section 3 discusses how CMS used Bosco to enable the use of a large opportunistic resource to augment their owned resources in order to process event data.

2. Bosco Architecture

The Bosco user experience can be described in 2 sections: installation/configuration and running jobs. Each of these areas were approached with the goal of improving the typical user experience for installing and running distributed computing software.

The Bosco architecture is divided into the submit host and the cluster login node. The submit host is where the user submits their jobs and where the user interacts with the Bosco system. The login node is the submit host for a cluster. The login node is assumed to not be maintained by the user submitting to Bosco. The login node has access to the local scheduler with commands such as qsub and qstat (for PBS). Bosco requires a login account in order submit and manage jobs on the cluster. For larger collaborations, the number of necessary accounts can be reduced drastically by using multi-user pilots that are in production use by both ATLAS and CMS.

2.1. Installation & Configuration

Though typically separate, Bosco combined the installation and configuration in order to improve the user experience. Both are done by a single script, the bosco_quickstart script that follows several steps:

(i) Determine the platform and download the appropriate version of Bosco (supports Mac, EL5/6, Debian 6)

(ii) Install Bosco into the user’s home directory.

(iii) Prompt the user for details of connecting their first cluster to Bosco.

The script downloads the Bosco binaries from a central server to the submit host. Bosco is installed into the user’s home directory by default in order to enable non-root installations. Connecting a cluster to the Bosco submit host requires configuring the secure connection to the cluster, and installing a small amount of software on the submit node that will be used for job submission and job status checks.

We have also created a native installer, a PKG, for the Mac version of Bosco. It is distributed in an Apple Disk Image (DMG) for consistency with other Mac software. Unlike the Linux installer, It does not automatically configure a cluster at first installation.

2.2. Running Jobs

Job submissions are done from the Bosco submit host, which in turn submit to the connected cluster login node. The image shown in Figure 1 shows the architecture of job submissions of a Bosco submit node.

First, the Bosco submit node connects to the login node over an SSH connection and creates a forwarded SSH connection back to the submit node which is used for file transfer. It creates the forwarded SSH connection in order to minimize the number of necessary connections between the Bosco submit host and the login node. By reusing the same SSH connection, we reduce...
Bosco Job Submission

Figure 1. Bosco Architecture

the number of necessary connections and SSH logins to 1. Further, the number of connections is not dependent on the number of jobs, as Bosco will reuse the same connection for all jobs submitted to a cluster. Minimizing the number of connections to a login node is important since many login nodes include firewall rules to slow down brute force SSH logins that block frequent successive SSH connections. The Bosco submit host does not require any open ports in a firewall, only outgoing connections to remote login node’s SSH port.

Next, Bosco checks for the necessary installed software on the login node, and starts the BLAHP daemon that will communicate with the scheduler on the login node. Next, the BLAHP daemon on the login node starts the file transfer daemon to connect back to the submit host through a forwarded SSH tunnel that Bosco created. The transfer daemon creates and transfers the job sandbox to the login node. The transfer to the login node is performed by HTCondor’s fault tolerant file transfer mechanisms and are entirely encrypted over the SSH connection. Authentication between the login node and the Bosco submit host is performed by a shared secret that was previously sent over the SSH connection to the BLAHP daemon.

After the files have been transferred, Bosco sends the job’s submission Classad \([5]\) to the BLAHP daemon to translate to the local site’s scheduler language. The BLAHP supports PBS \([6]\), LSF \([7]\), SGE \([8]\), SLURM \([9]\), and HTCondor schedulers. The BLAHP creates the submission file, and submits to the cluster’s scheduler. Bosco periodically polls the BLAHP over the SSH connection for the status of the job. Once the job is detected to have been completed, the BLAHP starts HTCondor’s transfer daemon to transfer the output sandbox back to the submitter machine.

2.3. Improving User Experience
In order to improve the user experience, at each step of the job process extra effort has been given to provide useful error messages in case of failures. For example, HTCondor was modified to relay the standard error for any commands sent to the BLAHP daemon, as useful debugging information is available there.

Also, an additional *traceroute* was created to test each step of the job submission process, including:

(i) SSH connection to the remote login host. Tests network connection, login host availability, passwordless SSH setup.
(ii) Job submission to the Bosco submit host. Tests Bosco daemon availability, Bosco submit host file system availability.
(iii) Job submission to the remote login host. Tests the remote scheduler availability, remote cluster software setup, input file transfer, cluster file system availability.
(iv) Job completion and status update from login host. Tests Bosco status check process, output file transfer.

The *traceroute* is very useful for debugging issues with a Bosco installation. It is designed to test each step in the job execution life cycle and give meaningful error messages and possible solutions.

3. Bosco and CMS
Using Bosco, the CMS experiment was able to process 125TB of collision data [10]. Bosco enabled the seamless integration of the Gordon [11] XSEDE resource into the CMS workflow framework. Bosco made it easy to add Gordon as an opportunistic resource.

Bosco provides a consistent submit syntax for the glideinWMS [12] Factory used by CMS for workflow management. The glideinWMS Factory ordinarily submits jobs using Condor-G [13] to gatekeepers that implement the Globus [14] or CREAM [15] gatekeepers. The submit syntax used for the Condor-G submissions and the Bosco submissions are very similar, easing the modifications necessary to the glideinWMS Factory.

The glideinWMS Factory relies on Globus or CREAM to provide status information of the submitted pilots and to transfer and secure credentials for the pilots. Bosco provides both of these abilities for glideinWMS. It is capable of querying the status of jobs submitted to the remote cluster. Further, it facilities the transfer of proxies through a secure connection to the remote cluster. It also allows the update of proxies on expiration without resubmitting the job.

![CMS Job Submission on Gordon Cluster](image)

**Figure 2.** Bosco Architecture

The Figure 2 shows the architecture of the Gordon solution. The glideinWMS Factory submits pilots to Bosco which in turn submits them to Gordon. The Gordon supercomputer runs the Torque [6] scheduler to manage the cluster. Bosco submits to the Torque scheduler using PBS syntax, which then runs the pilots on worker nodes in the cluster. The pilots then communicate with the glideinWMS VO Frontend in order to run jobs. The glideinWMS Factory monitors the status of the pilots on the Gordon cluster and submits more pilots as they complete.
4. Conclusions and Future Work

Development continues on the Bosco submission mechanisms. Future work will include improving the native installers and resilience of Bosco.

The HTCondor RPM already includes Bosco. But we do not have native installers for Debian or Mac. A prototype native installer has already been created for the Mac, but it has the downside of not having a GUI in order to configure or run Bosco. HTCondor already creates Debian packages, but they are not well tested when being used with Bosco. Further, the official Bosco Debian packages only support one version of Debian, and not the many versions and spinoffs (such as Ubuntu [16]).

The resilience of Bosco will be improved by allowing it to tolerate transitive errors. In the experience we have gathered running Bosco, local schedulers running PBS or SLURM occasionally are unresponsive. Bosco will allow up to 5 minutes of unresponsive scheduler before giving up and stating the jobs are in an unknown state. A better solution would be an exponential back off mechanism with a maximum time between queries. This would help back off when Bosco is the cause of the unresponsiveness (which has not been observed yet), or when the scheduler is unresponsive due to local issues.

Bosco is an easy to install solution for transforming a HPC computer to HTC. It has been used by experiments such as CMS to add opportunistic resources to existing HTC grids with minimal changes to their infrastructure. The Bosco user experience and resilience is continuing to be improved. Small and large experiments are trying Bosco and finding it to be efficient and useful for utilizing opportunistic resources.

Acknowledgments

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.

References

[1] Collaboration C, Adolphi R et al. 2008 Jinst 3 187
[2] Weitzel D 2013 Bosco — job submission manager URL http://bosco.opensciencegrid.org/
[3] Litzkow M J, Livny M and Mutka M W 1988 Distributed Computing Systems, 1988., 8th International Conference on (IEEE) pp 104–111
[4] Mao Z M, Rexford J, Wang J and Katz R H 2003 Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (ACM) pp 365–378
[5] Raman R, Livny M and Solomon M 1998 High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on (IEEE) pp 140–146
[6] Computing A and Computing G 2013 Torque resource manager
[7] Computing P Lsf scheduler
[8] Gentzsch W 2001 Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on (IEEE) pp 35–36
[9] Yoo A B, Jette M A and Grondona M 2003 Job Scheduling Strategies for Parallel Processing (Springer) pp 44–60
[10] Wagner R, Tattneni M, Hocks E, Yoshimoto K, Sakai S, Norman M L, Bockelman B, Stiligo I, Tadel M, Letts J et al. 2013 Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery (ACM) p 59
[11] SDSC 2013 Gordon user guide: Technical summary URL http://www.sdsc.edu/us/resources/gordon/
[12] Stiligo I 2008 Journal of Physics: Conference Series vol 119 (IOP Publishing) p 062044
[13] Frey J, Tannenbaum T, Livny M, Foster I and Tuecke S 2002 Cluster Computing 5 237–246
[14] Foster I and Kesselman C 1997 International Journal of High Performance Computing Applications 11 115–128
[15] Aiftimiei C, Andreetto P, Bertocco S, Dalla Fina S, Dorigo A, Frizziero E, Gianelle A, Marzolla M, Mazzucato M, Sgaravatto M et al. 2010 Future Generation Computer Systems 26 654–667
[16] Canonical 2013 Ubuntu: The world’s most popular free OS URL http://www.ubuntu.com/