Commissioning the HTCondor-CE for the Open Science Grid

2015 J. Phys.: Conf. Ser. 664 062003

(http://iopscience.iop.org/1742-6596/664/6/062003)

Journal of Physics: Conference Series **664** (2015) 062003 doi:10.1088/1742-6596/664/6/062003

Commissioning the HTCondor-CE for the Open Science Grid

Bockelman B[1], Cartwright T[2], Frey J[2], Fajardo E M[3], Lin B[2], Selmeci M[2], Tannenbaum T[2] and Zvada M[1]

[1] Department of Computer Science and Engineering, University of Nebraska-Lincoln, 118 Schorr Center, Lincoln, NE 68588-0150
[2] Department of Computer Science, University of Wisconsin-Madison, Madison WI
[3] Department of Physics, University of California San Diego, La Jolla, CA

E-mail: bbockelm@cse.unl.edu

Abstract. The HTCondor-CE is the next-generation gateway software for the Open Science Grid (OSG). This is responsible for providing a network service which authorizes remote users and provides a resource provisioning service (other well-known gateways include Globus GRAM, CREAM, Arc-CE, and Openstacks Nova). Based on the venerable HTCondor software, this new CE is simply a highly-specialized configuration of HTCondor. It was developed and adopted to provide the OSG with a more flexible, scalable, and easier-to-manage gateway software. Further, the focus of the HTCondor-CE is not job submission (as in GRAM or CREAM) but resource provisioning. This software does not exist in a vacuum: to deploy this gateway across the OSG, we had to integrate it with the CE configuration, deploy a corresponding information service, coordinate with sites, and overhaul our documentation.

1. Introduction

The Open Science Grid (OSG) is a national, distributed partnership for data-intensive science [1]. It is a world leader in distributed high-throughput computing, which focuses on maximizing usage of computing resources throughput a distributed infrastructure. For the OSG, the key component of the distributed infrastructure consists of approximately 100 computing clusters, mostly within the US.

In order to assemble the disparate set of computing resources into a coherent whole, OSG virtual organizations (VOs) utilize a technique called "resource provisioning." Through starting virtual machines or processes within a batch system, remote resources are "acquired" and added to a global pool of resources. In the case of the GlideinWMS [2] service used by multiple OSG VOs, the global pool of resources is a HTCondor [3] pool; the batch system worker node startup script launches a HTCondor worker node. Effectively, this allows the VO to create a global HTCondor batch system that grows and shrinks with the availability of worker nodes at remote clusters.

There are several models for acquiring resources at a site; the classic mechanism is to have a central *factory* which submits resource requests to a site-specific *gateway*. A gateway has the following three responsibilities:

- **Remote submission**: Submission of a resource request from a remote client to the CE service.

- **Authentication and Authorization**: Establishing the identity of the remote client and determining whether it is authorized to use the CE service.
- **Request Routing**: Conversion of the resource request from an abstract description to a concrete request into the local resource management system. This includes the transformation and tracking of the resource instance through its lifetime (submit/status/kill).

The request is materialized as a *pilot job* running within a batch system. Historically, the resource allocation model grew out of the infrastructure designed for remote job submission model; resource allocation requests were made to look identical to a job submitted remotely to the cluster.

The OSG has recently introduced new gateway software, the **HTCondor-CE**. This new gateway is tailored to meet the needs of the resource provisioning problem without carrying the baggage of the previous remote job submission model. It is meant to be a scalable, flexible, and easy-to-operate service. The HTCondor-CE integrates within the larger product; the OSG-CE, in turn, is meant to integrate with various central OSG services.

In Section 2, we provide an overview of commonly-used gateway software. Section 3 provides an overview of the constituent components of the HTCondor-CE. The primary CE integration and commissioning work is discussed in Section 4. Section 5 gives a few results and metrics about the CE. Finally, in 6, we discuss conclusions and future work.

2. Background

2.1. Job gateways

As described in Section 1, a gateway is a mechanism for acquiring resources at a computational site; a *job gateway* is specifically a gateway that provides access to a resources managed by a batch system (such as HTCondor, PBS or others). Before being used for resource provisioning, end-users would submit jobs directly to a single resource - or have the job submitted to the "best" available resource (according to some ranking metric) through the use of a *resource broker*.

The difference between resource acquisition and brokering is often explained through a grocery store analogy. Suppose there are ten lines to checkout in a store; each has a different-length line moving at an unknown rate. In the resource brokering model, you would estimate which line is shortest and stand in that. In the resource acquisition model, you would have ten people stand in different lines; the first person to get to the checkout register will give you her place in line.

Besides the HTCondor-CE described in this paper, commonly-used gateways include Globus GRAM (the previous gateway used by OSG) [4], CREAM [5], Arc-CE [6], and BOSCO [7]. Table 1 does a simple comparison between these systems.

Table 1: A high-level comparison of job gateways.

Name	Remote Submission	Routing	Auth
GRAM	GRAM (custom binary protocol)	Custom Perl scripts	GSI (X509-based)
CREAM	SOAP API	blahp daemon [8]	GSI
Arc-CE	GridFTP-based	Custom shell scripts	GSI
BOSCO	SSH	blahp daemon	SSH public keys

2.2. Pilot Systems

The method of resource acquisition on the OSG is a "pilot system"; it is envisioned that these are the most common users of the HTCondor-CE. The most popular examples on the OSG are PanDA [9] and GlideinWMS [2].

In these systems, end-users submit jobs to a central, global queue. The system will match the jobs to one or more compute resources; the aggregate of these matches forms a measurement of *pressure* for each site.

Based on the per-site pressure, a separate component (the "factory") submits generic jobs to the site gateway; it attempts to keep a fixed number of idle jobs in queue (up to a certain max running limit). These generic jobs - referred to as "pilot jobs" - enter the site batch queue. When started, they will connect to the central queue and run one or more end-user jobs. Thus, the pilots and central queue form a global batch system composed of resources acquired from each site. See Figure 1.

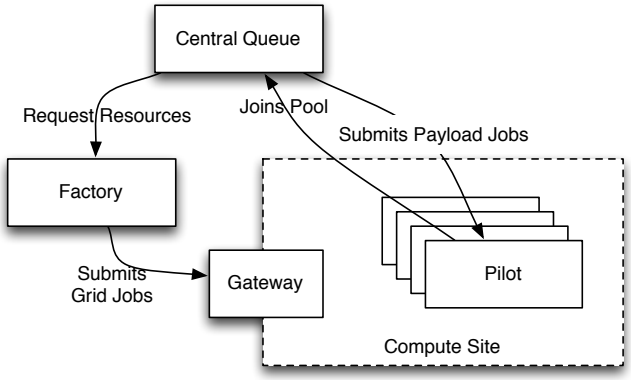


Figure 1: An overview of the pilot model for resource provisioning.

2.3. Other Resource Acquisition Models

While our work on the HTCondor-CE is partly motivated by addressing the needs of pilot systems, it should be mentioned there's a plethora of resource allocation (how a resource is assigned to an entity) and acquisition (once allocated, how a resource is accessed by the remote entity) models. In the Vac model [10], the site starts a specified VM image for a VO which starts a pilot that joins the global pool of resources (the resource appears "as if from the vacuum," hence the name). The site decides the number of appropriate VMs to launch and manages the credentials used to join the pool. With Amazon Web Service's CloudFormations [11] service, the VO specifies a template describing policies for when to launch VMs and provides the security credentials to use.

3. Composition of the HTCondor-CE

At its core, the HTCondor-CE is a special single-host configuration HTCondor. All running daemons are part of the HTCondor batch system - albeit in a non-standard configuration. The high-level gateway functions are provided by a combination of the `condor_schedd`, `condor_job_router` and `blahp` daemons.

- **Remote Submission**: Requests are submitted to a `condor_schedd` daemon as a ClassAd [12], a key-value-like description of the request and a set of input files (such as the pilot

startup script). The protocol between client and host is referred to as *HTCondor-C* and is built on top of the CEDAR [13] communication library.

- **Authentication and Authorization**: Authentication with the `condor_schedd` is done at the CEDAR transport layer. CEDAR allows the client to negotiate an appropriate authentication protocol (such as SSL, GSI, local filesystem, or Kerberos); for the HTCondor-CE, we utilize the GSI support. This allows us to invoke the Globus callout library to perform the mapping from a GSI proxy certificate to a user name; HTCondor's existing authorization policy layer allows fine-grained control over what actions each user may perform.
- **Request Routing** Once a request has been submitted to the HTCondor-CE's `condor_schedd`, it must be customized according to the site's policy and sent to the site's batch system. The `condor_job_router` examines incoming jobs and creates a transformed copy according to a set of site policies called *job routes*, which are examined at length in Section 3.1. The jobs are not run as a vanilla HTCondor batch system job, but forwarded to the site batch system using Condor-C for HTCondor batch systems or the `blahp` for all other batch systems; both Condor-C and the `blahp` transform the job ad into an appropriate job description file and manages basic job lifecycle activities such as submission, status queries, and job deletion. For example, when the site uses the PBS batch system, the `blahp` is the component which invokes `qsub`, `qstat`, and `qdel` for submit, status, and deletion, respectively.

Figure 2 illustrates the processes that typically run on the CE for a site with a HTCondor backend.

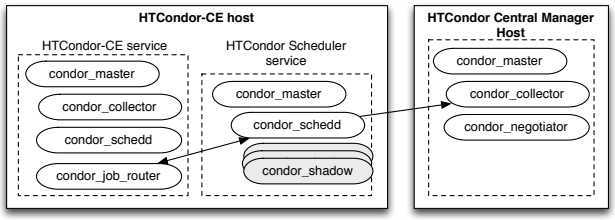


Figure 2: An overview of the processes present on the CE for a HTCondor backend. Note the CE's `condor_job_router` is responsible for communication between the CE and the site batch system; the `condor_schedd` joins the site batch system by advertising its presence to the collector

Submission to the CE is typically done through the use of Condor-G [14]; however, Section 4 discusses command line tools available for testing. From client to batch system, there exists several views of the submitted job:

- **Client job** This is the initial copy of the pilot job submitted to the queue on the glidein factory. From the glidein factory, it is submitted to the HTCondor-CE using the Condor-C submission protocol.
- **Grid job** This is the initial copy of the job on the CE; it is specified completely by the client submitter and has not been customized to fit site policy.
- **Routed job** A site-customized version of the job, after applying the edits specified by the `condor_job_router`.

- **Batch system job** The final version of the job, submitted to the site's batch system. In the special case of the HTCondor backend, the `condor_job_router` can route directly from the grid job to the backend batch system; in this case, the routed job and the batch system job are the same thing. Otherwise, the routed job is managed by the HTCondor-CE's `condor_schedd`.

Figure 3 outlines the job submission and execution process for a PBS-based site.

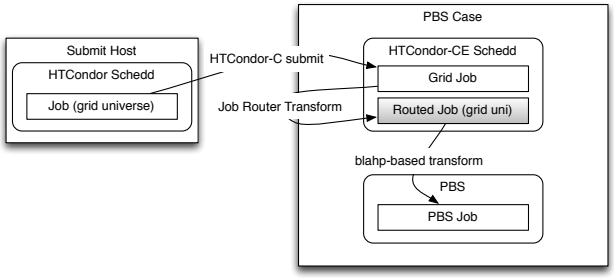


Figure 3: The sequence of job transforms from the submit host (a pilot factory) to the site's batch system.

3.1. Job Routes

The HTCondor-CE is unique among job gateway software in that, the HTCondor JobRouter [15] has an explicit, declarative language for expressing how to transform a resource request into a site job.

For each incoming job, the JobRouter will iterate through a list of routes and perform matchmaking to select a destination for the job. The route is a ClassAd specifying a requirements expression and a list of attributes to set, edit, and delete for matching jobs. See Figure 4 for an example of job routes; the full domain language is covered in [16]. This mechanism allows the site to express policies such as "CMS-owned jobs go to PBS queue *cms* while all other jobs go to PBS queue *other*."

Critically, routing allows us to maintain an abstraction layer. The router allows remote submitters to simply describe the resources needed. There is no need to describe site-internal details such as queue names or other site-custom attributes. As the ClassAd describing the grid job is schema-free, VOs and cooperating sites can define their own extensions without requiring a new release of the CE; the site and VO must agree on the attribute semantics and the CE administrator must implement these semantics using the job routes.

4. Commissioning the CE

4.1. Integration with the OSG-CE

At HTCondor-CE's conception, the OSG-CE software stack was a mature product with the Globus GRAM Gateway as its job gateway solution (as mentioned in Section 2). The migration from GRAM to HTCondor-CE required extensive manual QA testing and automated nightly testing. Initial rollout proceeded one volunteer site until approximately five CEs were in production; subsequently, we worked to convert all WLCG T2 sites (some urgency was provided by the restart of the LHC). For remaining sites, we have a goal of dropping GRAM support by April 2016.

The most difficult piece was the validation of all OSG-supported batch systems with most of the troubleshooting centering around the `condor_job_router` and the blahp. HTCondor-CE

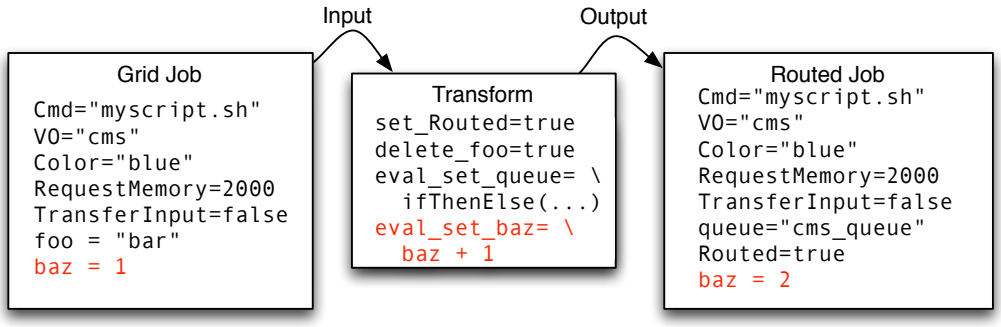


Figure 4: A simple illustration of a job route and how it transforms a grid job to a routed job. For example, note how the attribute `baz=1` in the grid job is set to the value of 2 in the routed job by the evaluation of the statement `eval_set_baz = baz + 1` in the transformation.

made unprecedented and heavy use of these components, which revealed bugs in the `blahp` and a lack of examples in the HTCondor JobRouter documentation. As site testing ramped up, job router configuration became a documentation priority. Comprehensive installation and troubleshooting guides soon followed, allowing more sites to come up more quickly. One lesson learned is the HTCondor-CE documentation needed to be significantly better than the previous GRAM documentation; site admins were silently ignoring deficiencies in the GRAM documentation because they had ten years' experience with the product.

Although HTCondor-CE strives to be only a special configuration of HTCondor, we found it advantageous to develop a few debugging tools:

- `condor_ce_job_router_info`: A command line tool which parses and executes the JobRouter logic. This allows sysadmins to check their routes for configuration errors and manually test the route against a specified job.
- `condor_ce_trace`: A tool which systematically tests each functional layer of the HTCondor-CE from service discovery to batch system integration. This allows admins to test end-to-end functionality and to isolate the problematic layer when the CE is not working.
- `condor_ce_run`: A tool for launching a single script on the remote host; useful for debugging remote systems, although not appropriate for running more than one or two jobs at a time.

Much of the other technical integration work was straightforward and resulted in changes to factory job submission, updates to the automated configuration tool, and the Resource and Service Validation (RSV) monitoring software.

4.2. Providing Resource Provisioning Information Services

For factories to discover available resources to provision, an information discovery service is needed. Historically, the OSG information services were focused on describing the state of a batch system: they provided information about each batch queue, the number of jobs in queue per VO, and the estimated response time. This was necessary for resource-broker submission - users needed to estimate which site was likely to finish their job first.

However, a new approach is needed for resource provisioning. The `condor_schedd` generates a HTCondor ClassAd containing the CE's contact information and a *resource catalog*. The resource catalog is a list of sub-ClassAds describing all resource types at the site and the pilot job attributes necessary to access the resource. For example, the site might require the pilot to specify an attribute `foo = "blue"` to access multicore resources. The factory does not need to

understand the semantics of the attribute "foo", just that it must be set to access the multicore batch slots.

The resource catalog model is analogous to Amazon EC2's webpage listing available instance types and EC2 users requesting a given instance in their request. Unlike the previous model, we purposely do not attempt to advertise the number of running pilots or available resources in the cluster.

5. Performance Results

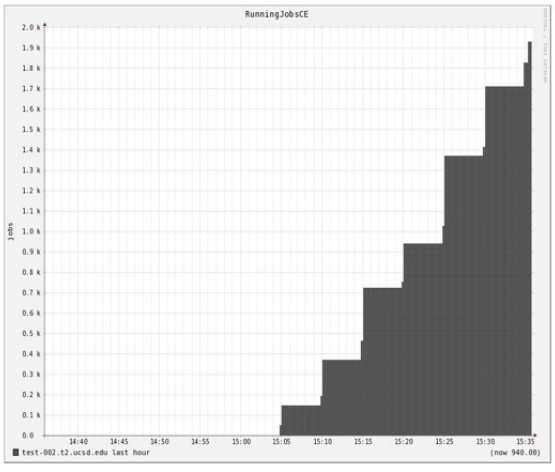
As mentioned in Section 1 HTCondor-CE is meant to be a scalable service; to verify, the OSG Software team performed scale tests analogous to prior GRAM tests [17]. These focus on measuring submission rates and the maximum sustained running jobs.

To perform these large scale tests without impacting real jobs running at a site a sleeper/shadow pool was used. A sleeper pool is a batch system deployed in parallel to a production pool (on the same worker nodes); jobs running in this separate pool by policy may not use CPU or other resources (network, disk or memory).

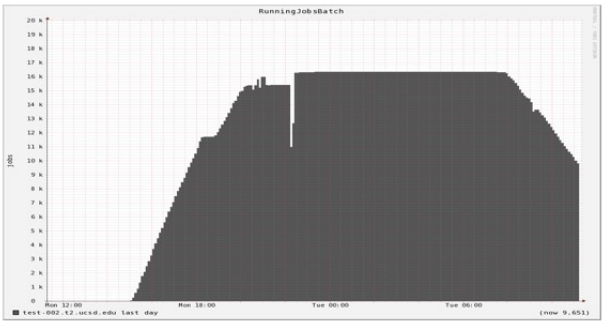
Along with a sleeper pool, we created an artificial load of submissions to the new CE, using the same Condor-G tool as production factories. This load was created using the loadtest_condor [18] updated to submit to the HTCondor-CE . We used tests 8 hour long jobs (typical production pilots are 48 hours), 100 jobs were kept idle in the CE queue at all time and input sandboxes of $50KB$ were used.

The maximum submission rate observed was of 70 jobs per minute and an average of 60 jobs per minute; the ramp-up is illustrated in Figure 5a. The maximum number of running jobs a single HTCondor-CE achieved was approximately 16,000 parallel running jobs, as can be seen in Figure 5b. This is twice as what is currently needed by any single site using the OSG software stack and the current limit of our sleeper pool.

The level of adoption of the new CE software can be observed by the number of entries at the CE Collector mentioned in Section 4.2. At the time of writing, 25 endpoints are reporting to the collector.



(a) The ramp up of 70 jobs per minute from idle state to running in an HTCondor-CE.



(b) A test illustrating 16,000 parallel jobs through a single CE.

Figure 5: Results from performance testing.

6. Conclusions and Future Work

We have successfully commissioned the HTCondor-CE for the OSG. We achieved our goals of constructing a job gateway solely out of HTCondor-based components, minimizing the amount of software maintenance. The HTCondor-CE is continuing to roll out to additional OSG sites and OSG is improving integration with the OSG-CE product.

Until now, we have focused on feature parity with GRAM gateway. In the future, we hope to expand on the "resource provisioning" model; in particular, future versions of HTCondor add support for container-based technologies such as Docker. We believe the future HTCondor-CE will be able to seamlessly provision resources for batch systems, containers, and VM-based resources.

Acknowledgments

This work is supported in part by the National Science Foundation through awards PHY-1148698 and ACI-1321762.

References

[1] A science driven production cyberinfrastructure: the Open Science Grid, author=Altunay, Mine and Avery, Paul and Blackburn, Kent and Bockelman, Brian and Ernst, Michael and Fraser, Dan and Quick, Robert and Gardner, Robert and Goasguen, Sebastien and Levshina, Tanya and others, journal=Journal of Grid Computing, volume=9, number=2, pages=201–218, year=2011, publisher=Springer

[2] Sfiligoi I 2008 GlideinWMS: a generic pilot-based workload management system *Journal of Physics: Conference Series* vol 119 (IOP Publishing) p 062044

[3] Thain D, Tannenbaum T and Livny M 2005 Distributed computing in practice: the condor experience. vol 17 pp 323–356

[4] Foster I and Kesselman C 1999 The globus toolkit pp 259–278

[5] Andreetto P, Bertocco S, Capannini F, Cecchi M, Dorigo A, Frizziero E, Gianelle A, Mezzadri M, Monforte S, Prelz F *et al.* 2012 New developments in the cream computing element *Journal of Physics: Conference Series* vol 396 (IOP Publishing) p 032004

[6] Nordugrid arc computing element URL `http://www.nordugrid.org/arc/ce/`

[7] 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

[8] Mezzadri M, Prelz F and Rebatto D 2011 Job submission and control on a generic batch system: the blah experience *Journal of Physics: Conference Series* vol 331 (IOP Publishing) p 062039

[9] Maeno T 2008 PanDA: distributed production and distributed analysis system for ATLAS *Journal of Physics: Conference Series* vol 119 (IOP Publishing) p 062036

[10] McNab A, Stagni F and Garcia M U 2014 Running jobs in the vacuum *Journal of Physics: Conference Series* vol 513 (IOP Publishing) p 032065

[11] AWS CloudFormations URL `http://aws.amazon.com/cloudformation/`

[12] Raman R, Livny M and Solomon M 1998 Matchmaking: Distributed resource management for high throughput computing *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC7)* (Chicago, IL)

[13] Miller Z, Bradley D, Tannenbaum T and Sfiligoi I 2010 Flexible session management in a distributed environment *Journal of Physics: Conference Series* vol 219 (IOP Publishing) p 042017

[14] Frey J, Tannenbaum T, Foster I, Livny M and Tuecke S 2002 Condor-G: A computation management agent for multi-institutional grids vol 5 pp 237–246

[15] Bradley D, Dasu D, Livny M, Mohapatra A, Tannenbaum T and Thain G 2010 Condor enhancements for a rapid response adaptive computing environment for lhc vol 219 p 062035 URL `http://iopscience.iop.org/1742-6596/219/6/062035/pdf/1742-6596_219_6_062035.pdf`

[16] Htcondor version 8.3.5 manual URL `http://research.cs.wisc.edu/htcondor/manual/latest/`

[17] Sfiligoi I, Pi H, Wüerthwein F, Theissen C and Dost J M 2011 Scalability of network facing services used in the open science grid vol 331 p 062023 URL `http://stacks.iop.org/1742-6596/331/i=6/a=062023`

[18] Sfiligoi I and Wüerthwein F 2010 Loadtest_condor —a workload generating framework for testing scalability and reliability of the condor system URL `http://osg-docdb.opensciencegrid.org/0010/001015/001/chep10_osgscal_rc5.pdf`