Building the HTCondor-CE cluster

Ste Jones

Liverpool

28 Mar 2019

Introduction

An HTCondor-CE can be used instead of a CREAM CE or an ARC CE. It is developed by the vendors of the HTCondor batch system, and is hence suitable for putting in front of a HTCondor cluster. But it is not tied to that batch system, and can support others.

CREAM CE will be phased out during 2019/20. ARC CE is a viable option to replace it, and that is well known and documented elsewhere. HTCondor is presently less familiar, so the objective of this document is to show how to make a basic HTCondor-CE installation. For simplicity, I have chosen to use HTCondor as the batch system behind the CE.

I'll describe first how to make a manual install, without any configuration system. I'll cover how to make a head-node that hosts both a HTCondor-CE (to handle grid jobs that come in) and a HTCondor central manager to dispatch those jobs to a worker node. To enable me to test the system right through, I'll also cover how to manually make a HTCondor worker-node. This approach will illustrate all the necessary packages and files used in a minimal but useful installation.

Having described a manual install, I'll move on to show how the work can be done at a site with a Puppet configuration system. I will not cover the details on how to puppetise the HTCondor central manager or the worker-nodes, since the idea is not to give a Puppet course. I'll just make use of existing work for those aspects.

So, those are the objectives. Now let's set up a HTCondor-CE system manually.

Manual Installation

You'll want root access to some admin system on your network as a central place to hold some files and copy them about as you make your cluster. I'll call it the admin system. You'll use the admin system as a place from which you will make the head-node, and the worker-node.

On the Admin system, in /root, get the kit of parts that I've prepared to help with the manual installation.

```
# git clone https://github.com/gridpp/htcce.git
```

Then follow the ./htcce/README.md file to create and test the new HTCondor-CE system.

Puppet installation

The manual install is easy, and it gives us something to compare to when we are trying to configure the system automatically with a configuration system such as Puppet. If we can get it going by hand, it should be possible to automate it. That's the theory I'm using, anyway. So I vaguely recommend, if you have time, to try the manual installation if you want to understand what's going on. Having said that, you can just have a go with the puppet module, which mostly works with some caveats that I'll explain as I go through it.

The htcondor_ce puppet module

Puppet can parametrise and automate the process described above, but it's a bit of work to get it going. Hence I'll describe how we used pre-existing htcondor_ce puppet module to make the build.

Since I'm covering only HTCondor-CE, not the batch system, you'll have to adopt some other scheme to get the batch system installed with a HTCondor central manager and a worker-node. There are various resources for doing this in the HEP-Puppet github repo.

```
git clone https://github.com/HEP-Puppet
```

Alessandra Forti gave a talk on these resources at GridPP40, see contributions 46 and 36.

```
https://indico.cern.ch/event/684659/contributions/
```

Alternatively, you can use any existing measures at your site to make the HTCondor batch system or you can use the manual procedures described above to manually make the batch system central manager and worker node respectively. It's up to you, but from now on, I'll assume you have a working HTCondor batch system. Note that I won't cover APEL integration, since the topic is documented fully here.

```
https://twiki.cern.ch/twiki/bin/viewauth/LCG/HtCondorCeAccounting
```

Yum Repositories

To start with, I suggest you use the repositories given in the manual build, i.e.

```
# git clone https://github.com/gridpp/htcce.git
# cd htcce/cm
# ls -lrt repos.tar
-rw-r--r-- 1 root root 10240 Mar 22 10:53 repos.tar
```

Copy that file into /etc/yum.repos.d/, untar it and yum clean all. And you then have the live repositories set up. Of course, you should make some puppet modules for this, depending on your own site standards.

Installing the module

Once you have those repos in place, you can have a go at installing HTCondor-CE. I'll explain a bit about how it works here at Liverpool. Basically, to install HTCondor-CE on our batch system head-node, we used a puppet module that comes from here:

```
# git clone https://github.com/HEP-Puppet/puppet-htcondor_ce.git htcondor_ce
```

(Note that another copy, the same at this date, is at Cern: https://github.com/cernops/puppet-htcondor_ce.)

Rsync the resulting htcondor_ce directory en masse into your puppet modules directory. But I found that I needed to apply some edits to the puppet files at our site. It would be right to roll these back to the authors at CERN, but for now, I'm just patching the files by hand. The edits are all described in Appendix 2. Make those edits.

Once all that is in place, it is necessary to perform a small number of extra tasks that don't get done by the puppet module. If you want, you can do these by hand or write another puppet module; it's your call. I describe these extra tasks further down below.

Setting up with Hiera

The Hiera set-up at Liverpool is noted in the bibliography. It should be sufficient here to show the settings that are required to drive these modules. The set-up might be quite different at your site, so I won't go into the details of how to set all that Hiera stuff up. Here are the parameters settings that drive the build of HTCondor-CE. Since the worker-nodes are just dumb, condor worker-nodes with no cognisance of the CE, these is nothing relevant there to be done. Obviously you'll have to substitute your own site particular, e.g. node names and so forth. In our setup, the headnode is hepgrid9, and the argus server is hepgrid3.

```
htcondor_ce::pool_collectors:
  - hepgrid9.ph.liv.ac.uk
htcondor_ce::condor_view_hosts:
  -
htcondor_ce::job_routes_template: 'htcondor_ce/job_routes.conf.erb'
htcondor_ce::uid_domain: $::domain
htcondor_ce::use_static_shadow: false
htcondor_ce::ce_version: '3.2.0-1.el7.centos'
htcondor_ce::lrms: 'condor'
htcondor_ce::lrms_version: '8.3.12-1.el7'
htcondor_ce::gsi_regex: '^\/DC\=ch\/DC\=cern\/OU\=computers\/CN\=(host\/)?([A-
Za-z0-9.\-]*)$'
htcondor_ce::manage_service: true
htcondor_ce::gsi_backend: 'argus'
htcondor_ce::argus_server: 'hepgrid3.ph.liv.ac.uk'
htcondor_ce::argus_port: 8154
htcondor_ce::argus_resourceid: 'http://authz-
interop.org/xacml/resource/resource-type/arc.hg6'
htcondor_ce::install_bdii: true
```

```
htcondor_ce::supported_vos: ['alice', 'atlas', 'biomed', 'calice',
'cernatschool.org', 'cms', 'dteam', 'epic.vo.gridpp.ac.uk', 'esr', 'fusion',
'geant4', 'gridpp', 'hyperk.org', 'ilc', 'lhcb', 'lz', 'lsst', 'magic', 'mice',
'na62.vo.gridpp.ac.uk', 'ops', 'pheno', 'planck', 'snoplus.snolab.ca',
'skatelescope.eu', 't2k.org', 'vo.northgrid.ac.uk', 'zeus', 'dune',
'vo.moedal.org']
htcondor_ce::goc_site_name: 'UKI-NORTHGRID-LIV-HEP'
htcondor_ce::benchmark_result: '09.99-HEP-SPEC06'
htcondor_ce::execution_env_cores: 8
htcondor_ce::election_type: 'leader'
htcondor_ce::election_hosts: $::fqdn
```

So, to recap, get the module, install it in your puppet set-up, and make the changes as described in Appendix 2 . Set up heira according to your site standard, and finally define a head-node that uses the puppet modules and inherits the Hiera parameters similar to above. I can't give much specific guidance on that because each site has different versions of puppet, hiera and various site specific standards.

Extra things to do after the puppet module

The puppet module doesn't do everything. Hence you'll need to do this work as well, over and above the work accomplished by htcondor_ce.

For the security, install the host certificates and keys, as per following. Note that the condor versions are just copies.

```
-r--r--r-- 1 root root 1606 Mar 12 15:41 /etc/grid-security/hostcert.pem
-r-------- 1 root root 1675 Mar 12 15:41 /etc/grid-security/hostkey.pem
-r--r--r-- 1 condor condor 1606 Mar 12 15:41 /etc/grid-security/condorcert.pem
-r-------- 1 condor condor 1679 Mar 12 15:41 /etc/grid-security/condorkey.pem
```

To make the BDII work, do this. Edit /etc/condor-ce/config.d/01-common-auth.conf, adding FS added to the SEC_DEFAULT_AUTHENTICATION_METHODS line.

And put this BDII file, 16setCentralManager.config, in (both?) /etc/condor/config.d/ and /etc/condor-ce/config.d/ with this content:

```
CENTRAL_MANAGER = hepgrid9.ph.liv.ac.uk
GLUE2DomainID = UKI-NORTHGRID-LIV-HEP
```

ARGUS Integration

It took me an inordinate amount of time to get this to work, so I'll try to flatten the issues I had.

In general principles, this is how it works. A job flies in to the HTCondor-CE (hepgrid9.ph.liv.ac.uk at Liverpool). It passes over the authentication and mapping to some ARGUS server (hepgrid3.ph.liv.ac.uk.) The ARGUS server looks at the credentials, the proxy, and compares them with its policies for the system in question. If all is well, it authenticates the user and returns a mapping, i.e. an account to use. In due course, HTCondor-CE submits the job into the batch system under the guise of the mapped user. Right, that's the idea. So we'll look at specifics of how this is set up at my site.

I believe the relevant Hiera parameters for configuring this set-up (i.e. for the htcondor_ce puppet module) are as follows:

```
htcondor_ce::gsi_regex: '^\/DC\=ch\/DC\=cern\/OU\=computers\/CN\=(host\/)?([A-Za-z0-9.\-]*)$'
htcondor_ce::gsi_backend: 'argus'
htcondor_ce::argus_server: 'hepgrid3.ph.liv.ac.uk'
htcondor_ce::argus_port: 8154
htcondor_ce::argus_resourceid: 'http://authz-interop.org/xacml/resource/resource-type/arc.hg6'
```

I'm not sure what the first line does, but the other lines hook up to ARGUS.

BDII Installation

If you have set up the system as laid out above, there should be no further work needed to get the BDII going, apart for the tweaks in Appendix 1. These tweaks correct the slot counts to correspond with the actual slots allocated in a node; there is a slight error in the CERN version of this code.

I'll also add this note that may help some poor sap, since it took me forever to work it out for myself. Here are some helpful facts. First, HTCondor-CE is just a special deployment of HTCondor. Second, the BDII uses HTCondor Python Bindings to interact with the HTCondor system. Third, the HTCondor Python Bindings expect to work off of the /etc/config dir (not the /etc/condor-ce dir). Hence things get tangled up. The simplest fix I could find was to put the config for the HTCondor-CE BDII in the /etc/condor dir, instead of /etc/condor-ce. The upshot is this: if you are having trouble with the BDII, think about using this layout instead of the standard one.

Tests

Use the tests described in the manual section, above.

Accounting Integration

The bibliography contains a link that gives full details on how APEL accounting works in a HTCondor-CE context, so I won't go over it here.

GOCDB Registration

You register the node in GOCDB with this service type:

```
org.opensciencegrid.htcondorce
```

Once APEL accounting is working, add it also with this service type:

```
gLite-APEL
```

Bibliography

HTCondor-CE Accounting.

https://twiki.cern.ch/twiki/bin/view/LCG/HtCondorCeAccounting

I'll give some sources to the most important pieces of information I could find for installing HTCondor-CE.

Talks by Iain Steers at HTCondor workshop.

https://indico.cern.ch/event/467075/contributions/1143797/

https://indico.cern.ch/event/467075/contributions/1143807/

The Hiera set-up at Liverpool is shown here:

https://www.gridpp.ac.uk/wiki/Centos7_Adoption

These two articles on the GridPP wikisite go over the ideas behind benchmarking and publishing:
https://www.gridpp.ac.uk/wiki/Benchmarking_procedure,
https://www.gridpp.ac.uk/wiki/Publishing_tutorial

An example build involving HTCondor, but based on SL6 so you'll need to adjust it a bit and omit the ARC bits.

https://www.gridpp.ac.uk/wiki/Example_Build_of_an_ARC/Condor_Cluster

Various other ways to install a HTCondor batch system were discussed at this GridPP40 gathering at Pitlochry, in particular in a talk by Alessandra Forti on migrating to CentOS7.

https://indico.cern.ch/event/684659/contributions/2897404/

Appendix 1 - Fix the BDII provider to use real slots provided, not just detected CPUS

The BDII provider, /var/lib/bdii/gip/provider/htcondor-ce-provider, that comes with the system is a bit naive. It just says that if a node that has, say, 16 possible hyperthreads (cores * 2, what have you) then it must have 16 job slots on it. This is not so. Some of our 16 hyperthread nodes can only have max 12 jobs running, due to memory.

To fix this in a set-up like ours, with partitionable slots, this patch is used (see below). What does this mean? It means I'm counting up only the slots with ID = 1. This tells me the allocation, which the right logic for this job.

To put in the patch, around line 129 of /var/lib/bdii/gip/provider/htcondor-ce-provider, change this constraint 'State=!="Owner"' to 'SlotTypeID == 1'. And, from line 129 to line 145, change all 'DetectedCpus' to 'TotalSlotCpus'. You will then get the right counts in the BDII output.

Appendix 2 – Edits to the CERN htcondor_ce module

I “patched” the htcondor-ce puppet module to make it work for me. The changes were to prevent it from installing some packages I didn't want, update some GSI security settings , and to get rid of some validation code that was broken, perhaps due to the puppet version I am using (3, not 4.) I should really pass all this over to the maintainers to get it done properly.

File: /htcondor_ce/manifests/config/bdii.pp

Change: Alter the location and name of the BDII conf file, since the BDII seems to get its config from the condor system, not the condor-ce system, when they co-located on the same head-node. This is an issue with condor python bindings. Note that CERN uses, perhaps, a separate computer for the BDII, and hence the installation does not have a dedicated condor. But we are co-locating both CE and batch system, hence we needed to change this location.

```
7c7
<   $bdii_ce_config      = '/etc/condor-ce/config.d/06-ce-bdii.conf'
---
>   $bdii_ce_config      = '/etc/condor/config.d/06-ce-bdii.conf'
```

File: htcondor_ce/manifests/install.pp

Change: Don't install some packages. I found I didn’t need them, since we are collocating the servers. And we don’t “require” condor either, because we installed that by hand.

14c14

< package { ['globus-rslsl', 'blahp', 'empty-ca-certs']: ensure => present, }

> #package { ['globus-rsl', 'blahp', 'empty-ca-certs']: ensure => present, }

18c18

< require => Package['condor', 'blahp', 'globus-rsl', 'empty-ca-certs'],

---

> #require => Package['condor'], # , 'blahp', 'globus-rsl', 'empty-ca-certs'],

File: /htcondor_ce/templates/condor_mapfile.erb

Change: Put in a line to map root.

> GSI "^\/C\=UK\/O\=eScience\/OU\=Liverpool\/L\=CSD\/CN\=hepgrid9\.ph\.liv\.ac\.uk$" root

File: htcondor_ce/manifests/auth/argus.pp

Change: Get rid of the validate_string, validate_integer checks, since they broke my build; we use Puppet 3.

```
19,21c19,21
<   validate_string($argus_server)
<   validate_integer($argus_port)
<   validate_string($argus_resourceid)
---
>   #validate_string($argus_server)
>   #validate_integer($argus_port)
>   #validate_string($argus_resourceid)
```

End of document