

TWiki > LCG Web > AccountingTaskForce > HTCondorAccounting > HtCondorCeAccounting (2019-09-12, SteveJones)

#### [HTCondor-CE Accounting via APEL client software](#)

##### [Introduction](#)

##### [Technical setup](#)

##### [Build a HTCondor-CE/HTCondor cluster system](#)

##### [Implement scaling factor](#)

##### [Set up data extraction scripts](#)

##### [Configure and run the parser](#)

##### [Configure and run the apelclient to make the jobrecords](#)

##### [Tests on a HTCondor-CE](#)

## HTCondor-CE Accounting via APEL client software

### Introduction

The Liverpool T2 is using a HTCondor-CE in front of some of its HTCondor worker-nodes. There was no standard accounting tool for HTCondor-CE, so we adapted the standard APEL client software to fit this new use case. This article describes the principles behind this approach, and how to deploy it at another site. For those who are interested, we wrote some [Design notes](#) to describe the choices we made (the notes contain a great deal of information about scaling factors.)

At present, support only exists for HTCondor-CE in front of a HTCondor batch system. There are plans to extend the support to other batch systems.

### Technical setup

#### Build a HTCondor-CE/HTCondor cluster system

We built a HTCondor-CE/HTCondor batch cluster with (at present) ~ 915 slots. This was a non-trivial task, so I out reference the setup of this to [HTCondor-CE Cluster](#). For the purposes of this article, the important detail is that the batch system is configured to insert into the batch job log data a named data-value (known as a [ClassAd](#), in HTCondor-speak) which tells us the scaling factor used on the node (this is used for regularising the work from our heterogeneous cluster; different nodes have different powers and hence have different factors.) This scaling factor is pulled from the system by our batch system data extraction script and used to build the batch log files.

So, with a HTCondor-CE/HTCondor batch cluster, jobs are run and job histories are stored. The task then is to extract two types data as input into the APEL client software; CE logs (also known as BLAH logs) and batch system logs. These are written, on a daily basis, to a well known location, where they are picked up by the APEL client software when it runs. The APEL client software has a database table that remembers when a file has been parsed, preventing it being parsed again.

Note: The host running HTCondor-CE/HTCondor must be set up to be a client of some "site APEL system" holding the APEL mysql database (documented elsewhere). This is standard MYSQL functionality. And versions of the APEL parser and client software containing changes for scaling factors (using a field called `cputmult`) must be installed; this feature first appeared in `apel-client/apel-parser 1.8.0-1`, but a superior version, `1.8.1-1`, has been released since; use that if possible. The matter is discussed in some detail in the [Design notes](#).

```
# yum install apel-client-1.8.1-1.el7.noarch apel-parsers-1.8.1-1.el7.noarch
```

Note that full configuration details on APEL client is not covered here; I only deal with the details needed to use it with HTCondor-CE.

#### Implement scaling factor

Please note: in this discussion, slots, threads, cores, CPUs, and so on are words that I use somewhat interchangeably, although there are subtle differences in their meanings; e.g. by slot, I'm referring to a single core slot (although some slots can have more cores.)

Jobs that run on the batch system are measured to find the amount of resource they utilised. Generally, we multiply the number of slots used by the job by the time it ran for, and then multiply the result of that by some value that represents the power of a slot on the node. The concept of "power of a slot on the node" needs some discussion. Nodes have multiple cores to run threads on. We want to use the node to its maximum. So we want to load it with a sufficient number of threads to get the maximum throughput. If we use too few threads, we will not be running the node optimally. And if we use too many threads, we will also not be running optimally. For any particular node, there is some ideal number of threads where the maximum amount of work gets through the system in a certain period of time. That is how we want to load our nodes. And we load our nodes in units of one thread (for a single core job) or eight threads (for a multicore job). We need the load to be somewhere near that ideal spot, and we need to do trials and tests to find the spot. That's called benchmarking, and from the benchmark process, for any particular node, we get a number of threads which represents the sweet spot and a value of the power output of each of those threads when it's going. And it's that power value that I am calling the "power of a slot on the node".

For reasons I won't go into (because I don't know what they are) it is customary to adopt a standard (i.e. "benchmark") power of the slots (on any node) in a heterogeneous cluster, e.g. 10 HEPSPEC06. Once done, we assume that all jobs ran on such a slot. But some (even all) nodes don't actually deliver 10

HEPSPEC06 per slot/thread. Hence the idea of "scaling in the batch system" was born. We make all the nodes look as if they deliver 10 (say) HEPSPEC06 per slot/thread by modifying the run time to make the maths correct. If this seems like a kludge, you're right - it is. But that is how things are done. And once the values are scaled, one can send them, along with the benchmark value to which all the scaling was done (i.e. using a comparison standard of 10 HEPSPEC06 in my example) and all the accounting will work out right. The portal will assume that all jobs ran on a 10 HEPSPEC06 node, so, to get the work (hours of HEPSPEC06) it just needs to multiply the submitted run time values by the cores and by (in this case) 10, since 10 HEPSPEC06 is the basis for all the measurements.

So, the only problem left is how to do "scaling in the batch system". Some batch systems, famously PBS/Torque, have inbuilt features to do this. Many, including HTCondor, do not. So we must implement it somehow. The rest of this snippet will discuss how we go about this at the HTCondor-CE/HTCondor site at Liverpool.

On the head node, looking at the `/etc/condor/config.d/17system_job_machine_attrs` file, we see this config (Ral stands for Rutherford Appleton; the origin of this scheme.)

```
SYSTEM_JOB_MACHINE_ATTRS = RalScaling
```

This defines a [ClassAd](#) variable, [RalScaling](#), which specifies a machine attribute (something special to the node) that will be recorded in the job [ClassAd](#), and hence will follow the job, and eventually wind up in the history file. [RalScaling](#) will be inserted into the job [ClassAd](#). Hence the scaling factor, which is set on the node, will be called [MachineAttrRalScaling0](#) in the history logs. Next, let me show you the changes on the worker node.

```
# cd /etc/condor/
# find . -type f -exec grep -li ralscaling {} \;
./config.d/00-node_parameters
./config.d/condor_config.local
```

So the config exists in two files. Looking at `00-node_parameters`

```
RalScaling = 1.062
```

This scaling factor, which is specific to this sort of node and is set by the build system, shows that this node delivers 1.062 times the benchmark value. Hence it will travel, as [MachineAttrRalScaling0](#), to the history files. The other config element, in `condor_config.local`, says this:

```
STARTD_ATTRS = $(STARTD_ATTRS) RalScaling
```

With this set up, the [ClassAd](#) variable will show up in the job histories where it can be extracted later. The scaling multiplication is not done on the node. It is actually done in the parsing phase, at the point where when the APEL parser reads the batch system log records into the APEL client database.

## Set up data extraction scripts

No RPM was available to install the data extraction scripts prior to htcondor-ce 3.3.0-1. For those versions, it is necessary to do a [Manual Install](#).

To make installation easier, starting with htcondor-ce 3.3.0-1, an rpm containing the data extraction scripts is now shipped, e.g. htcondor-ce-apel-3.3.0-1.el7.noarch.rpm. I briefly describe the installation process here.

It's very simple. You need to yum install the htcondor-ce-apel rpm matching the version of the htcondor-ce main application version.

The `/etc/condor-ce/config.d/50-ce-apel.conf` file contains default settings that are mostly OK to leave, but you will have to edit it to correct the `APEL_SCALING_ATTR` scaling factor to have the correct name (see above.) Also deal with any `.rpm`save and `.rpm`new artefacts, should there be any. The file at Liverpool is shown below.

```
APEL_CE_HOST = $(CONDOR_HOST)
APEL_BATCH_HOST = $(CONDOR_HOST)

# The CE_ID is the CE, the port, and the batch system,
# ending with -condor to show the type of batch system.
APEL_CE_ID = $(APEL_CE_HOST):$(PORT)/$(APEL_BATCH_HOST)-condor

APEL_OUTPUT_DIR = /var/lib/condor-ce/apel/

# The attribute in the batch system that stores the scaling factor
APEL_SCALING_ATTR = MachineAttrRalScaling0
```

You can produce accounting data by running the scripts; it should show up in `/var/lib/condor-ce/apel/`

```
/usr/share/condor-ce/condor_batch.sh
/usr/share/condor-ce/condor_blah.sh
```

## Configure and run the parser

---

To parse these data files, we'll need the APEL client software parser configuration (in `/etc/apel/parser.cfg`) to be set up properly. For this job, a typical setup might be as follows:

```
[db]
hostname = hepgrid95.ph.liv.ac.uk
port = 3306
name = apelclient
username = apelclient
password = SomeSecretPasswordNooneCanGuess

[site_info]
site_name = UKI-NORTHGRID-LIV-HEP
lrms_server = hepgrid6.ph.liv.ac.uk:9618/hepgrid6.ph.liv.ac.uk-condor

[blah]
enabled = true
dir = /var/lib/condor/accounting/
filename_prefix = blah
subdirs = false

[batch]
enabled = true
reparse = false

type = HTCondor
parallel = false
dir = /var/lib/condor/accounting/
filename_prefix = batch
subdirs = false

[logging]
logfile = /var/log/apelparser.log
level = INFO
console = true
```

This tells the APEL client software about the DB it will connect to. It tells it to run the blah parser, then the batch parser, to pick up the files created in the previous steps. Run the `/usr/bin/apelparser` tool to populate the database with "blah" and "event" records, and you might see output like this if all is working well (obviously, all this would be done in the same cron script, normally.)

```
2018-12-18 16:25:56,277 - parser - INFO - =====
2018-12-18 16:25:56,277 - parser - INFO - Starting apel parser version 1.7.1
2018-12-18 16:25:56,298 - apel.db.backends.mysql - INFO - Connected to hepgrid95.ph.liv.ac.uk:3306
2018-12-18 16:25:56,298 - apel.db.backends.mysql - INFO - Database: apelclient; username: apelclient
2018-12-18 16:25:56,299 - parser - INFO - Connection to DB established
2018-12-18 16:25:56,299 - parser - INFO - =====
2018-12-18 16:25:56,299 - parser - INFO - Setting up parser for blah files
2018-12-18 16:25:56,301 - apel.parsers.parser - INFO - Site: UKI-NORTHGRID-LIV-HEP; batch system: hepgrid6.ph.liv.ac.uk:9618/hepgrid6.ph.liv.ac.uk-condor
2018-12-18 16:25:56,301 - parser - INFO - Scanning directory: /var/lib/condor/accounting/
2018-12-18 16:25:56,302 - parser - INFO - Filename does not match pattern: batch-20181217-hepgrid6
2018-12-18 16:25:56,306 - parser - INFO - Parsing file: /var/lib/condor/accounting/blah-20181217-hepgrid6
2018-12-18 16:25:58,236 - parser - INFO - Parsed 2061 lines
2018-12-18 16:25:58,236 - parser - INFO - Ignored 0 lines (incomplete jobs)
2018-12-18 16:25:58,236 - parser - INFO - Failed to parse 0 lines
2018-12-18 16:25:58,239 - parser - INFO - Finished parsing blah log files.
2018-12-18 16:25:58,239 - parser - INFO - =====
2018-12-18 16:25:58,239 - parser - INFO - Setting up parser for HTCondor files
2018-12-18 16:25:58,241 - apel.parsers.parser - INFO - Site: UKI-NORTHGRID-LIV-HEP; batch system: hepgrid6.ph.liv.ac.uk:9618/hepgrid6.ph.liv.ac.uk-condor
2018-12-18 16:25:58,241 - apel.parsers.htcondor - INFO - Site: UKI-NORTHGRID-LIV-HEP; batch system: hepgrid6.ph.liv.ac.uk:9618/hepgrid6.ph.liv.ac.uk-condor
2018-12-18 16:25:58,241 - parser - INFO - Scanning directory: /var/lib/condor/accounting/
2018-12-18 16:25:58,242 - parser - INFO - Parsing file: /var/lib/condor/accounting/batch-20181217-hepgrid6
2018-12-18 16:25:59,451 - parser - INFO - Parsed 2061 lines
2018-12-18 16:25:59,452 - parser - INFO - Ignored 0 lines (incomplete jobs)
2018-12-18 16:25:59,452 - parser - INFO - Failed to parse 0 lines
2018-12-18 16:25:59,452 - parser - INFO - Filename does not match pattern: blah-20181217-hepgrid6
2018-12-18 16:25:59,454 - parser - INFO - Finished parsing HTCondor log files.
2018-12-18 16:25:59,454 - parser - INFO - Parser has completed.
2018-12-18 16:25:59,454 - parser - INFO - =====
```

---

## Configure and run the apelclient to make the jobrecords

---

The apelclient takes the CE information (given in the blah logs) and the batch system information (mainly run times, taken from the batch logs) and joins them to make the jobs logs - records with all the information from both sources of data.

If possible, use apelclient 1.8.1-0 or better. Prior to this version, the software makes a BDII GLUE1 query to obtain the benchmark. Since HTCondor-CE has no Glue1 BDII (at the time of writing), this cannot work. Version 1.8.1-0 fixes this by allowing the benchmark value to be input manually, supplied from local config. To use this feature, the /etc/apel/apelclient.cfg file resembles this example.

```

[db]
hostname = hepgrid95.ph.liv.ac.uk
port = 3306
name = apelclient
username = apelclient
password = SomeSecretPasswordNooneCanGuess

[spec_updater]
enabled = false
site_name = UKI-NORTHGRID-LIV-HEP
lrms_server = hepgrid6.ph.liv.ac.uk:9618/hepgrid6.ph.liv.ac.uk-condor
spec_type = HEPSPEC
spec_value = 10.0

manual_spec1 = hepgrid6.ph.liv.ac.uk:9618/hepgrid6.ph.liv.ac.uk-condor,HEPSPEC,10.0

[joiner]
enabled = true
local_jobs = false

[unloader]
enabled = true
dir_location = /var/spool/apel/
send_summaries = false
withhold_dns = false
interval = latest
send_ur = false

[ssm]
enabled = false

[logging]
logfile = /var/log/apel/client.log
level = INFO
console = true

```

It is worth noting that [spec\_updater] enabled = false disables the BDII query. and the manual\_specN line(s) are inserted in the database, and then used instead.

When it is set up, just run /usr/bin/apelclient to get ssm (stomp message manager) records, i.e. data in the format we use for sending into APEL. In this example, sending via ssm is disabled. You can change that when you think it's right, and set up the ssm config file as appropriate.

If, for any reason, you can't use apelclient > 1.8.1-0 or better, a [Manual Workaround](#) exists.

And that's it; I've run through how to use APEL client software to send SSM records from a HTCondor-CE set up with a HTCondor batch system.

As a convenience, you may wish to combine all; the steps into one script (example below) and run it with one cron for all. It also protects from pile-ups.

```

#!/bin/bash
# accountingRun.sh
# sjones@hep.ph.liv.ac.uk, 2019
# Run the processes of a HTCondor accounting run

/usr/share/condor-ce/condor_ce_blah.sh # Make the blah file (CE/Security data)
/usr/share/condor-ce/condor_batch.sh # Make the batch file (batch system job run times)

/usr/bin/apelparser # Read the blah and batch files in
/usr/bin/apelclient # Join blah and batch records to make job records
/usr/bin/ssmsend # Send job records into APEL system

```

## Tests on a HTCondor-CE

A slightly formal full-loop test was done on 22 January 2019. In this, batch logs from actual jobs that ran on the HTCondor-CE system were independently measured on the client with different software to find the work represented in the jobs. To do this, the following script was written.

```
#!/usr/bin/perl

# workDone.pl - independently measure the H506 hours done in HTCondor-CE batch log file

# Ste Jones, 22 Jan 2019

# Example line:
# batch-20190121-hepgrid6:7017_hepgrid6.ph.liv.ac.uk|prdatl28|197|29|3|1548114739|1548114936|19036|58116|8|0.883|

my $siteNormalisationBenchmark = 10.0;

my $totScaledWallClockSecsWithCores = 0.0;

while (<STDIN>) {
    my $line = $_;
    chomp($line);
    my @fields = split(/\|/, $line);
    my $s = $fields[2] * $fields[9] * $fields[10];
    $totScaledWallClockSecsWithCores = $totScaledWallClockSecsWithCores + $s ;
}

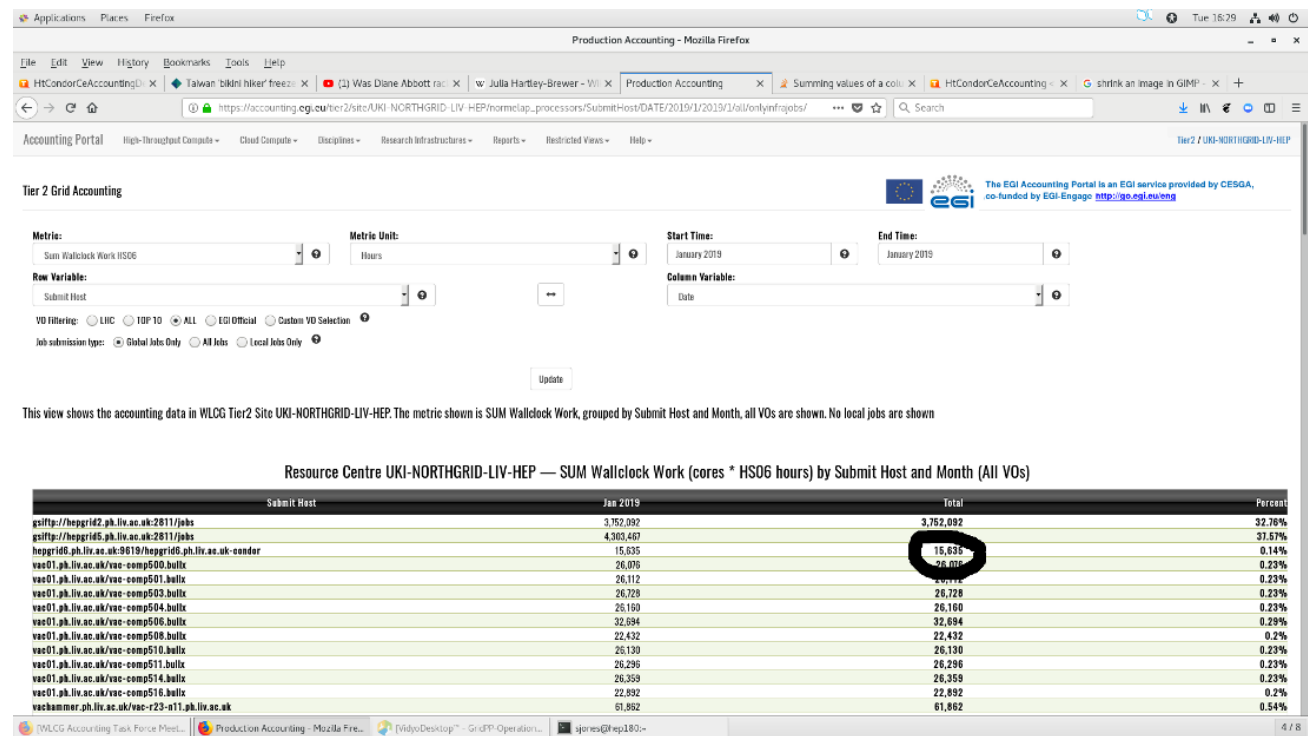
my $hs06Hours = $totScaledWallClockSecsWithCores / 3600.0 * $siteNormalisationBenchmark;

print $hs06Hours, "\n";
```

All the batch logs produced so far were fed into this tool, and the results were summed (less one batch file that had been made during tests and was invalid, and hence never sent to the portal.)

```
# for f in batch-*; do ./workDone.pl < $f; done | perl -ape '$s+=$F[0]{$_}$_=$s'; echo
15641.3933222221
```

This result could then be compared with the current status of hepgrid6.ph.liv.ac.uk in the Accounting Portal:



Tier 2 Grid Accounting

Metric: Sum Wallclock Work H506 Metric Unit: Hours Start Time: January 2019 End Time: January 2019

Row Variable: Submit Host Column Variable: Date

VO Filtering: ☐ LHC ☐ TOP 10 ☒ ALL ☐ EGI Official ☐ Custom VO Selection

Job submission type: ☒ Global Jobs Only ☐ All Jobs ☐ Local Jobs Only

Update

This view shows the accounting data in WLCG Tier2 Site UKI-NORTHGRID-LIV-HEP. The metric shown is SUM Wallclock Work, grouped by Submit Host and Month, all VOs are shown. No local jobs are shown

Submit Host	Jan 2019	Total	Percent
gsiftp://hepgrid2.ph.liv.ac.uk:2811/jobs	3,752,092	3,752,092	32.78%
gsiftp://hepgrid5.ph.liv.ac.uk:2811/jobs	4,303,487		37.67%
hepgrid6.ph.liv.ac.uk:9619/hepgrid6.ph.liv.ac.uk-conda	15,635	15,635	0.14%
vac01.ph.liv.ac.uk/vac-comp500.bellx	26,076	26,076	0.23%
vac01.ph.liv.ac.uk/vac-comp501.bellx	26,112	26,112	0.23%
vac01.ph.liv.ac.uk/vac-comp503.bellx	26,728	26,728	0.23%
vac01.ph.liv.ac.uk/vac-comp504.bellx	26,160	26,160	0.23%
vac01.ph.liv.ac.uk/vac-comp506.bellx	32,094	32,094	0.29%
vac01.ph.liv.ac.uk/vac-comp508.bellx	22,432	22,432	0.2%
vac01.ph.liv.ac.uk/vac-comp510.bellx	26,130	26,130	0.23%
vac01.ph.liv.ac.uk/vac-comp511.bellx	26,296	26,296	0.23%
vac01.ph.liv.ac.uk/vac-comp514.bellx	26,359	26,359	0.23%
vac01.ph.liv.ac.uk/vac-comp516.bellx	22,892	22,892	0.2%
vachammer.ph.liv.ac.uk/vac-r23-n11.ph.liv.ac.uk	61,862	61,862	0.54%

The two values, 15,635 vs. 15,641, match to within 0.04 % . That is an acceptable level of rounding error given the other potential errors in the larger system.

An earlier test report from January 2019 is here.

\* [HTCondorCeAccJan2019.odp](#): [HTCondorCeAccJan2019.odp](#)

-- [SteveJones](#) - 2019-07-04

I	Attachment	History	Action	Size	Date	Who	Comment
	<a href="#">HTCondorCeAccJan2019.odp</a>	r1	<a href="#">manage</a>	7458.2 K	2019-01-17 - 13:59	<a href="#">SteveJones</a>	
	<a href="#">Portal2.png</a>	r1	<a href="#">manage</a>	2664.8 K	2019-01-22 - 17:35	<a href="#">SteveJones</a>	

Topic revision: r19 - 2019-09-12 - [SteveJones](#)

Copyright &© 2008-2019 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.  
Ideas, requests, problems regarding TWiki? [Send feedback](#)

