# Migrating INFN-T1 from CREAM-CE/LSF to HTCondor-CE/HTCondor

**S Dal Pra, F Fornari, D Michelotto, A Chierici**
INFN-CNAF, v.le B. Pichat 6/2, Bologna 40100, IT
E-mail: `dalpra@infn.it`

**Abstract.** The INFN Tier-1 datacentre provides computing resources to several HEP and Astrophysics experiments. These are organized in Virtual Organizations submitting jobs to our computing facilities through Computing Elements, acting as Grid interfaces to the Local Resource Manager. We are phasing-out our current LRMS (IBM/Platform LSF 9.1.3) and CEs (CREAM) set to adopt HTCondor as a replacement for LSF and HTCondor-CE in place of CREAM. A small instance has been set up to practice with the cluster management and evaluate the feasibility of our migration plans to a new LRMS and CE set. A second cluster instance has been setup to work on production. A number of management tools have been adapted or rewritten in order to integrate the new system with the existing infrastructure. Two different accounting solution for the HTCondor-CE have been implemented, and the more reliable one have been adopted. A python tool has been written to disentangle the management of HTCondor machines from our puppet instance, and to enable a quicker configuration of the cluster nodes. The monitoring tools tied to the old system are being adapted to also work on the new one. Finally, the most relevant setup steps have been documented in a public wiki page and a support mailing has been created to help other INFN sites willing to migrate their LRMS and CE to HTCondor. This document reports about our experience with HTCondor-CE on top of HTCondor and the integration of this system into our infrastructure.

## 1. Introduction

The INFN-T1 currently provides a computing power of about 400KHS06 over more than 35000 computing slots on approximately one thousand physical Worker Nodes. These resources are accessed from remote by 24 Grid VOs and locally by 25 or more user groups. The IBM/Platform LSF 9.1.3 batch system [1] arbitrate access to all these competing user groups, both Grid and local, according to a fairshare policy designed to prevent underutilization of the available resources or starvation of lower priority groups, while ensuring a medium–term share proportional to configured quotas.

Grid users cannnot directly interact with the site resources. Instead, they contact Computing Elements (CE) at the site. These services are designed to act as a frontend for Grid users to the underlying Local Resource Manager (LRMS) submitting and managing their jobs on behalf of them through their whole life cycle.

We adopted the CREAM [2] CE implementation at our site for several years and it proved to be an effective solution at providing Grid access to our local resources, managed by LSF. However, we decided to phase out LSF because of licensing cost reasons and move to adopt an open source alternative, such as HTCondor [3]. Starting with January 2019 we ceased the renewal for LSF official support and software updates, while remaining owner of the existing licenses for the product, so that we could keep using it normally; during the same time, the CREAM CE was no more being actively developed and no further releases were expected. In fact, the end of Support for the CREAM-CE by the end of 2019 was announced in February of the same year.

Because of these reasons, we decided to migrate our LRMS from LSF to HTCondor, and we selected HTCondor-CE as a replacement for CREAM-CE. This is a quite natural choice, because it is maintained by a development team tightly coupled with that of HTCondor and both are being actively maintained and developed. In the following sections we provide a report about our experience so far with HTCondor and HTCondor-CE, our plans and migration status.

## 2. Differences between LSF and HTCondor

An in–depth comparison of the differences between LSF and HTCondor is beyond the scope of this document, however it is useful to understand the main differences of their conceptual model:

### *2.1. LSF conceptual model*

LSF follows a centralized model: a single LSF Master node knows and control the status and behaviour of every node in the cluster. The composition and the configuration of the whole cluster is defined by a small set of text files, which are usually kept on a shared filesystem where the Master is the only node having write access. Jobs are submitted to the master from nodes which must belong to the cluster; the master queues them and manages dispatching of selected pending jobs to selected Worker Nodes, while monitoring the status of jobs and Worker Nodes. This model has the advantage that a complete cluster management is possible from a single machine, with the drawback of being more prone to scalability issues.

### *2.2. HTCondor conceptual model*

HTCondor follows a distributed model: Jobs are submitted from one or more Submit Nodes (also known as Schedd, by the name of the Scheduler Daemon they run) and are defined by a set of resource requirements (JobAds). In order for the jobs to run, these requirements must match with corresponding MachineAds defined by the Compute Nodes in the cluster (or pool, in HTCondor terminology). A Central Manager is in charge of collecting information and negotiating between resources and resource requests. As with LSF, this machine must be unique in the pool, however it has by design a limited and specific role, so that it can be much more lightweight. Every machine in the pool has his own set of configuration files. An implication of this fact is that a method to distribute configuration files across the cluster nodes must be devised.

This model has greater flexibility and a pool can easily grow in size and scale well by just adding more Submit Nodes, with the drawback of adding some complexity in the pool management.

### *2.3. Differences between HTCondor-CE and CREAM-CE*

The main task of the HTCondor-CE is basically the same as that of the CREAM-CE: when an authorized remote Grid user submits a job to a CE, that is translated to a corresponding batch job submission. A set of facilities and functionalities are implemented to enable the remote user to check or control the job status, provide input files at submission, and retrieve ouptut at job end. A major difference is that the HTCondor-CE was designed to specifically support pilot jobs: these fundamentally are rather simple scripts that when starting on a Compute Node connect to a remote service owned and managed by the submitter, who provides them with a suitable payload to run (*pull model*). CREAM was designed to support a more complex and general case, where each submitted Grid job is supposed to provide the complete description of the actual software to run (*push model*). The pull model is nowadays a de-facto standard, thus the HTCondor-CE implementation can be more lightweight because handling pilots at CE level is a simpler task.

## 3. The HTCondor Cluster

### *3.1. The testbed*

To learn and practice with the new batch system and CEs, to verify how these would work together, how other components, such as monitoring, provisioning and accounting systems can be integrated with HTCondor and HTCondor-CE, a small HTCondor 8.6.13 cluster was set up during spring 2018. Soon after, a HTCondor-CE instance was added, in late April, and finally a common Submit node, dedicated to all of our local users. One important goal for us was that of reducing at most the impact of the change for our user communities and a test instance has been fundamental to devise a reasonable migration plan.

HTCondor is a very mature opensource product, deployed at several major Tier-1 for years, thus we already know that it will certainly fit our use cases. On the other hand the HTCondor-CE is a recent product and issues could arise. Hence, we took the decision to concentrate our integration resources on this one.

The test cluster consists of:

- a HTCondor-CE on top of a HTCondor Submit Node.
- a HTCondor Central Manager, hosting Collector and Negotiator daemons
- 3 Worker Nodes (Compute Nodes, in HTCondor terms), 16 slot each.
- a Submit Node for local users.

In order to make the migration more transparent for our local user communities, a choiche was made to set up a single Submit Node instead of adding one on every User Interface. Doing so, users could continue working from their owned User Interface as they used to do with LSF, and simply adapt their submission command. Moreover, the activity of the users cannot interfere with the one of the Schedd, which remains on a separate host.

### *3.2. HTCondor-CE Installation and setup*

Our first HTCondor-CE installation was not straightforward since the original maintainers of the package included several tools and configurations that were only relevant in the OSG [4] ecosystem, meaning that most of the default settings and dependencies were unmet for EGI standards. Short after however, HTCondor-CE RPMs were made available on the same official repository of HTCondor.

*3.2.1. Setup* There exist readily available Puppet configurations for HTCondor and HTCondor-CE but these were not “out of the box” ready for our application given the extensive use of Hiera.

The modules have been adapted to make them compatible with our system, by translating into regular Puppet files the parameters configured by hiera. In the meanwhile, the setup of HTCondor was finalized looking at the official documentation.

*3.2.2. Configuration* The first configuration was completed manually. The only original documentation for HTCondor-CE was in the OSG documentation and specific to the OSG ecosystem, thus the setup had to be completed by trial and error. Once a working configuration was obtained, a set of integration notes were added to a public wiki, to share our experience with other non OSG users. Currently, the oficial documentation [5] has greatly improved, and maintaining custom documentation has become less important.

## 4. Accounting

As of 2018, at the time when this activity started, the official EGI accounting tool, APEL [6], had no official support for HTCondor-CE. On the other hand, INFN-T1 has had a custom accounting system [7] in place for several years after decommissioning DGAS [8], which was the official accounting tool adopted by italian Grid sites until 2014. Since we wanted to keep our own account infrastructure we took the decision to spend development and integration work to adapt it to the new batch system, looking for a suitable way to retrieve from HTCondor the same information that we retrieve from CREAM-CE and LSF. This would enable us to perform a gradual migration and have both LSF and HTCondor managing production activity, without having to keep two distinct accounting methods for the time of the transition.

### *4.1. Python Bindings*

A promising approach to do so has been that of using python and the *python bindings*, a set of api interfaces to the HTCondor daemons. These can be used to query the SCHEDD at the CE and retrieve a specified set of data about recently finished jobs, which are subsequently inserted into our local accounting database. An important fact to note, is that the grid information (User DN, VO, etc.) are directly available together in the job classAd together with all the needed accounting data. This greatly simplifies the accounting problem, as it is no more necessary to collect grid data separately from the BLAH [9] component and then look for matches with the corresponding grid counterpart, as it was the case when using the CREAM-CE.

This solution has been used during the 2018 to provide accounting for the HTCondor-CE testbed cluster. After a few months however, it became clear that this method couldn't be considered robust and reliable enough, as occasional timeouts or corner cases were experienced. In order to deal with these exceptions one should enforce further consistency checks (such as preventing double counting or detecting unread records) to ensure that each accounting round executed correctly, and this could have greatly complicated an otherwise simpler task. Because of this, a different method to retrieve accounting data was adopted.

### *4.2. HTCondor job history log*

A HTCondor configuration allows to define a directory where an accounting text file is written by the Submit Node for each finished job. This is much similar to what happens with LSF, where a textual one-line record is appended to a daily accounting file. It is worth noticing that both the Schedd of the HTCondor-CE and the Schedd of the Submit Node can write their accounting file records. In our case we only collect those at the Submit Node side, as they provide all the information that we want to collect.

- Enable per job accounting in HTCondor by defining `PER_JOB_HISTORY_DIR` to point to an existing folder; text files named `history.<ClusterId>.<ProcId>` will be created there at the end of each job.
- These files are made of `<key> = <value>` pairs, one per line. A very simple python function reading one such file can return the key,value pairs into a dictionary.
- Each file is *complete*, because it also contains the Grid information that the X509 proxy certificate used by the user at submission time brings.
- The key/value pairs of interest are collected and used to compose an `INSERT INTO` SQL query to add the data into our accounting database.
- After parsing the file and adding the extracted usage record to the database, the file is moved to a backup directory, to prevent double counting.

This accounting model is even simpler than when using CREAM-CE with LSF, since there is no more need to lookup for matches between sets of grid records (produced by the blah component of the CE) and batch records (produced by LSF). Moreover, this is the same way OSG does the accounting via gratia [10] into GRACC [11].

We collect the same set of keys that the apel HTCondor parser collects, and a few more, for internal use, as we also do for several years with CREAM/LSF. The ability to do so adds some versatility to our accounting model, as we can track a new kind of resource or metric for our own interest just by collecting a new key/pair value from the job history files.

The accounting data stored into our PostgreSQL database are to be transmitted to the EGI Accounting Portal in Apel Format, which can be directly obtained using a SQL view.

### *4.3. Accounting for GPU usage*

A special Worker Node in the HTCondor pool is equipped with two NVIDIA K40 GPUs which can be requested by Grid jobs submitted to our HTCondor-CE. The usage record for GPU resources is identified by the keys `AssignedGPUs` and `GPUsProvisioned` in the job history file. Tracking these allows us to account for GPU usage using a direct SQL query:

```
acct=> SELECT COUNT(*) AS "N",
acct->        sum(runtime) AS "WCT",
acct->        username, exechosts
acct->  FROM htjob WHERE gpu=1 GROUP BY
acct->   username,exechosts limit 5;
   N   |   WCT    |  username   |   exechosts
-------+----------+-------------+---------------
     5 |        4 | atlas220    | hpc-200-06-07
     1 |        1 | dteam003    | hpc-200-06-07
    11 |        9 | dteam039    | hpc-200-06-07
 54474 | 12931308 | pilatlas030 | hpc-200-06-07
     5 |     1275 | virgo034    | hpc-200-06-07
(5 rows)

Time: 805.730 ms
```

Succesful tests of GPU usage have been performed so far by users of the ATLAS and VIRGO community.

## 5. HTCondor-CE and HTCondor

After some time to become confident with the main configuration tasks, the testbed began working with pilot jobs submitted by the four LHC experiments starting from September 2018. The system proved to be stable and smooth, being able to work unattended. This confirms that this system can be a reliable substitute for CREAM-CE and LSF.

The HTCondor batch system is a mature product with a large user base. We have put less effort at investigating it deeply. We already know that most or all of needed features will work well. Rather, some effort have been put on dealing with configuration management.

### *5.1. Configuration management*

Eventhought a standard base of Puppet classes have been adapted to our management system, an additional python tool have been written to improve flexibility and readiness. The tool works by reading and enforcing on each node of the cluster a set of configuration directives written on text files accessible from a shared filesystem. The actual set and the read order depends on the host role. In addition, our Worker Nodes can belong to several named hostgroups (an enclosure, a rack, a cpu model and so on) files with name of the form `<groupname>.conf` are read for WNs belonging to those groups, if such files exists. Doing so, a large cluster can be quite easily managed as a collection of host sets. For example, if a specific classAd requirement must be enforced on a set of nodes, it is sufficient to define the hostgroup in a text file, write the classAd into the file named after the hostgroup and reconfigure the startd on the interested machines.

The tool is not yet publicly available but it is now becoming more reliable and we plan to make it public soon.

### *5.2. Management tools*
A set of command line utilities have been written using the condor python bindings api. When using LSF, a set of three commands is sufficient to get a quite complete report about te state of the Cluster, the running or pending jobs and the Compute Nodes. In HTCondor, the command line tool offers a much richer set of capabilities, but manually crafting a complete set of options to obtain the desired output can quickly become cumbersome. For this reason, three python scripts have been implemented, to simplify usage by restricing to the most useful and used commands:

- `hjobs to query for running, pending or finished jobs`
- `hhosts to query about Compute nodes status`
- `hqueues to report number of pending and running jobs per user.`

These scripts are being gradually improved by supporting more command line options and have been also used to provide data about running and pending jobs to our Monitoring infrastructure [12].

An other tool have also been implemented to manage and configure group shares, and a few more have been implemented to provide sanity checks for the Compute Nodes (i.e. health status of gpfs and cvmfs file systems and other).

## 6. The migration
After using the testbed cluster a plan for a smooth migration have been devised and actuated as follow:

- Install and setup a new HTCondor cluster, with a few more HTCondor-CE and an initial small set of Worker Nodes
- Enable the LHC VOs on the new cluster (by May, 2019)
- Enable other Grid VOs
- Enable local submissions. These are made from a heterogenous set of users, with a potentially rich set of individual needs and can require a considerable administrative effort to meet all of them.
- Add more WN to the new cluster gradually (in progress)
- Move to HTCondor the last remaining WNs, when no more users are needing it.

At the time of writing (June 2020) the transition to HTCondor is almost completed. Figure 1 shows Grid and Local running jobs, for a total of nearly 33 thousand cores and 400 KHS06. Nodes have been moved gradually from LSF to HTCondor according with the needs of our user communities: some could work with both systems at the same time and other preferred to stop using LSF and directly start with HTCondor immediately after. A positive fact is that converting the individual WNs did not require reinstall nor reboot of the machines. Closing the machines to LSF and starting HTCondor after the drain was enough. In some cases we also

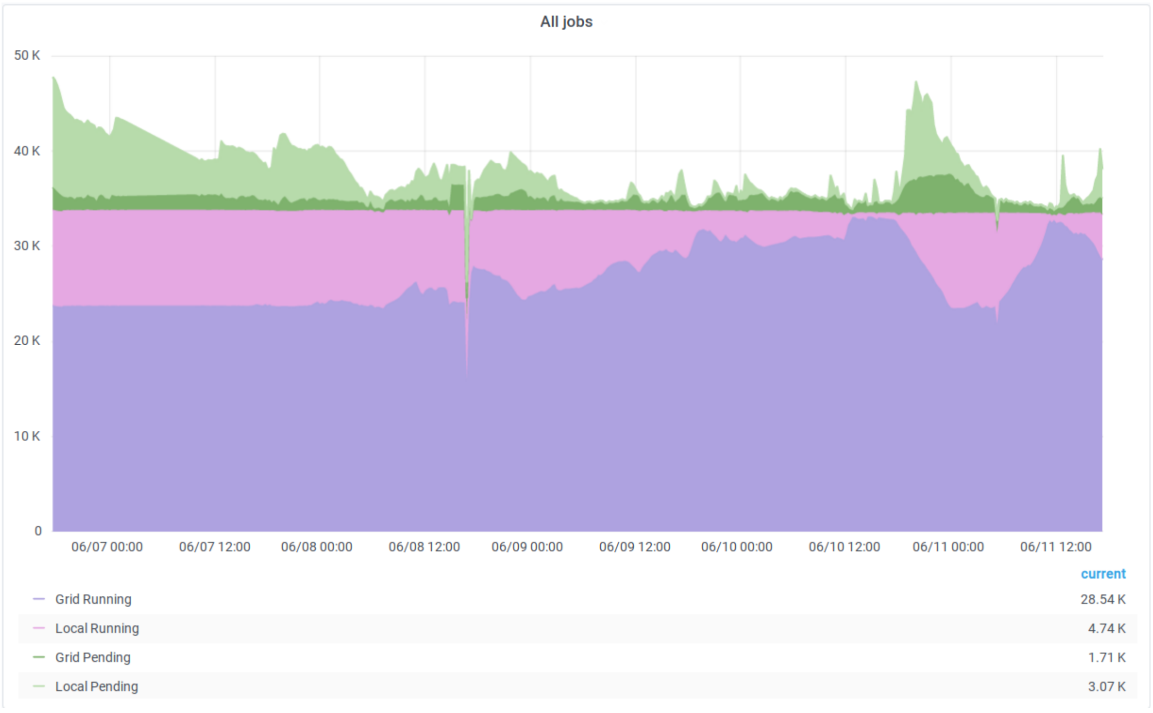

**Figure 1.** Migrating new Worker Nodes to HTCondor

have had both LSF and HTCondor jobs running at the same time on several nodes. This was done to allow long term running jobs on LSF to complete while not losing CPUtime in the wait for a complete drain. To do so, we used the management tools described in the previous section: we defined an hostgroup in HTCondor and configured it to use less CPUs than those actually available.

While the cluster was growing in size we added more HTCondor-CEs up to a total of six. We differentiated by installing two of them as bare metal with a fast Solid State Disk (SSD) for the spool directories, as adviced by HTCondor documentation, two as Virtual Machines managed by Ovirt and two managed by VMware. We have not observed overload issues in the core services so far.

### *6.1. Experience with HTCondor*

Several configurations and features have been gradually added to the production cluster during the time of the transition. Part of these were to reimplement in HTCondor features already defined for LSF and other were set to meet user needs, such as wrapping jobs inside singularity containers or supporting DAG jobs [13]. During this process we have learnt how HTCondor really differs and how it can significantly enhance the capabilities and the quality of our computing cluster. One revealing example for us has been that of the sanity checks. With LSF we were used to experience sets of WNs being occasionally closed by the master because some GPFS [14] filesystem was repeatedly causing job failures at the node. With HTCondor now the Compute Node can update the status of each gpfs filesystem in the set of his machine classAds and submitted jobs are modified to require machines having the specific filesystem they need in good status. This method prevents job failures and machine being closed for high job failure rate.

HTCondor and HTCondor-CE are working well and can easily sustain our volume of work and the expected size growth at our Tier-1.

A number of minor issues are still present and we are working to solve them with the effective support provided by the HTCondor developers and the user community. At this stage of our migration process we can say that this work was definitely worth the effort. We need to gain more experience but we can clearly see now that more capabilities are at our hand.

## References

[1] Etsion Y and Tsafrir D 2005 *School of Computer Science and Engineering, The Hebrew University of Jerusalem* **44221** 2005–13

[2] 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

[3] Thain D, Tannenbaum T and Livny M 2005 *Concurrency and computation: practice and experience* **17** 323–356

[4] Pordes R, Petravick D, Kramer B, Olson D, Livny M, Roy A, Avery P, Blackburn K, Wenaus T, Würthwein F *et al.* 2007 The open science grid *Journal of Physics: Conference Series* vol 78 (IOP Publishing) p 012057

[5] 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

[6] Jiang M, Novales C D C, Mathieu G, Casson J, Rogers W and Gordon J 2011 An apel tool based cpu usage accounting infrastructure for large scale computing grids *Data Driven e-Science* (Springer) pp 175–186

[7] Dal Pra S 2014 *Accounting data recovery. A case report from INFN-T1*

[8] Piro R M, Guarise A and Werbrouck A 2003 An economy-based accounting infrastructure for the datagrid *Proceedings. First Latin American Web Congress* (IEEE) pp 202–204

[9] 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

[10] Canal P, Borra S and Malani M 2006 *CHEP-Computing in High Energy Physics*

[11] Retzke K, Weitzel D, Bhat S, Levshina T, Bockelman B, Jayatilaka B, Sehgal C, Quick R and Wuerthwein F 2017 Gracc: New generation of the osg accounting *Journal of Physics. Conference Series* vol 898 (Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States))

[12] Bovina S and Michelotto D 2017 The evolution of monitoring system: the infn-cnaf case study *J. Phys. Conf. Ser.* vol 898 p 092029

[13] Couvares P, Kosar T, Roy A, Weber J and Wenger K 2007 Workflow management in condor *Workflows for e-Science* (Springer) pp 357–375

[14] Schmuck F B and Haskin R L 2002 Gpfs: A shared-disk file system for large computing clusters. *FAST* vol 2