What's new in the HTCondor Software Suite (HTCSS)?
What's coming up?

HTC 23 – Madison, WI

Todd Tannenbaum
Greg Thain
Center for High Throughput Computing
University of Wisconsin-Madison
Release Channels

› **Long-Term Support (LTS) Releases** *(formerly 'stable series')*
  - Only bug fixes
  - vMajor.0.Update (e.g. 10.0.0, 10.0.1, 10.0.2, …)
  - Today: HTCSS v10.0.8

› **Feature Releases** *(formerly 'developer series')*
  - Bug fixes plus new features
  - vMajor.Minor.Update (e.g. 10.1.0, 10.2.0, 10.3.0 …)
  - Currently: v10.8.0 …. *This is the v11.0.0 release candidate!*
Release Channels

› Long-Term Support (LTS) Releases (formerly 'stable series')
  • Only bug fixes
  • vMajor.0.Update (e.g. 10.0.0, 10.0.1, 10.0.2, …)
  • Today: HTCSS v10.0.6

› Feature Releases (formerly 'developer series')
  • Bug fixes plus new features
  • vMajor.Minor.Update (e.g. 10.1.0, 10.2.0, 10.3.0 …)
  • Next month: v10.8.0 …. *This is the v10 release candidate!*

v23.0.0
Why?

• To synchronize support and major release cycles with OSG Software
• Allows HTCSS binaries to be the same between htcondor.org and OSG software (only build/package once)

New Major version each summer

Major Version number = year of release (23, 24, 25, …)

The two most recent LTS series will be supported e.g.
• Summer 2025: Version 25.0.0 will be released. Support for v24.0.x and v25.0.x; support for 23.0.x will end.

HTCondor-CE version number will reflect HTCSS version.
Release/Versioning Change

Why?

• To synchronize support and major release cycles with OSG Software
• Allows HTCSS binaries to be the same between htcondor.org and OSG software (only build/package once)

New Major version each summer

Major Version number = year of release (23, 24, 25, …)

The two most recent LTS series will be supported e.g.
• Summer 2025: Version 25.0.0 will be released. Support for v24.0.x and v25.0.x; support for 23.0.x will end.

HTCondor-CE version number will reflect HTCSS version.
Some New Terminology

HTCondor Software Suite (HTCSS)

- Access Point (AP)
- Execution Point (EP)
- HTCondor Pool = Central Manager + Execution Point(s) (EP)
- HTCondor System = AP + EP + CM
- HTCondor Compute Entrypoint (HTCondor-CE)

Now it is just OSG

You mean placing jobs at an HTCSS Access Point

Arrrrgh!

I ♥ Open Science Grid

I ♥ submitting jobs with HTCondor!
Some New Terminology

HTCondor Software Suite

- **Access Point (AP)**
- **Execution Point (EP)**

**HTCondor Pool** = Central Manager + Execution Point(s) (EP)

**HTCondor Compute Entry Point (CE)**

Now it is just OSG

I ♥ submitting jobs with HTCondor!

You mean placing jobs at an HTCSS Access Point?

Arrrrgh!
Glossary

AP (Access Point)
An Access Point (AP) is the machine where users place jobs to be queued to be run. It usually runs the `condor_schedd` and other daemons.

Classad
A classad is a set of key value pairs. Every object in an HTCSS is described by a classad. Classad values can also be an expression, which can be evaluated in the context of another classad, in order to provide matching or ranking policy.

CM (Central Manager)
The Central Manager (CM) is the machine with the central in-memory database (`condor_collector`) of all the services, an accountant and `condor_negotiator`.

Daemon
A long-running process often operating in the background. An older term for "service". The `condor_master`, `condor_collector`, `condor_schedd`, `condor_startd` and `condor_shadow` are some of the daemon in HTCSS.

EP (Execution Point)
The Execution Point (EP), sometimes called the worker node is where jobs run. It is managed by the `condor_startd` daemon, which is responsible for dividing all of the resources the machine into slot.

Job
Job has a very specific meaning in the HTCSS. It is the atomic unit of work in HTCSS. A job is defined by a job classad, which is usually created by `condor_submit` and a submit file. A job can have defined input files, which HTCSS will transfer to the EP. One or more operating system processes can run inside a job. Every job is a member of a cluster of jobs, which have cluster id. Each job also has a "proc id". The job id uniquely identifies every job on an AP, the id is the cluster id followed by a dot followed by the proc id.

Sandbox
A directory containing shared attributes, includin input files. The directories are called "sandbox".
In the past year, there were 12 LTS releases and 10 Feature releases incorporating 140 enhancements and 162 bug fixes.
Since HTCondor Week 2022, there were 12 LTS releases and 10 Feature releases incorporating 140 enhancements and 162 bug fixes.

See "Detailed Notes" at https://htcondor.org/htcondor/release-highlights/
So, What's Cooking?

Let's start with some User Experience work at the Access Point…
Evolving new command line user interface

› *htcondor* <noun> <verb>
  
  • "htcondor job submit", "htcondor job status", …
  • "htcondor dag submit", "htcondor dag status", …
  • "htcondor jobset submit", "htcondor jobset status", …
  • "htcondor annex create", "htcondor annex status", …
  • "htcondor eventlog read"

› Legacy tools (condor_q, condor_submit, condor_history, …) not going anywhere…
**condor_q - looking at one running job**

$ condor_q 123.45

```
-- Schedd: login04.osgconnect.net : <192.170.231.217:9618?><... @ 07/12/23
21:43:13
OWNER  BATCH_NAME    SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
toddt  ID: 123       7/12 21:37    _      1      _   10000 123.45

Total for query: 1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0
held, 0 suspended
```
condor_q - looking at the details

$ condor_q -l 123.45

AccountingGroup = "group_opportunistic.EvolSims.toddt"
AcctGroup = "EvolSims"
AcctGroupUser = "anushd"
AllowedExecuteDuration = 72000
Args = "44"
AutoClusterId = 3232
BytesRecvd = 0.0
BytesSent = 0.0
ClusterId = 36371245
Cmd = "/home/anushd/cluster_code/cwrapper.sh"
CommittedSlotTime = 0
CommittedSuspensionTime = 0
CommittedTime = 0
CondorPlatform = "${CondorPlatform: X86_64-CentOS_7.9 $"
CondorVersion = "${CondorVersion: 10.6.0 2023-06-26 PackageID: 10.6.0-0.656423 RC $"
CoreSize = 0
CpusProvisioned = 1
...

<table>
<thead>
<tr>
<th>CONDOR PLATFORM: X86_64-CentOS_7.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDOR VERSION: 10.6.0 2023-06-26 PackageID: 10.6.0-0.656423 RC</td>
</tr>
<tr>
<td>CORE SIZE: 0</td>
</tr>
<tr>
<td>Cpus Provisioned: 1</td>
</tr>
</tbody>
</table>

...
Job 123.45 is currently running on host exec221.chtc.wisc.edu.
It started running again 2.1 hours ago.
It was submitted 3.6 hours ago.
Its current memory usage is 2.5 GB out of 4.0 GB requested.
Its current disk usage is 3.8 GB out of 5.5 GB requested.
It has restarted 2 times.
Goodput is 80% (0.5 hours badput, 2.1 hours goodput).
$ htcondor job status 123.45

Job 123.45 is currently running on host exec221.chtc.wisc.edu.
It started running again 2.1 hours ago.
It was submitted 3.6 hours ago.
Its current memory usage is 2.5 GB out of 4.0 GB requested.
Its current disk usage is 3.8 GB out of 5.5 GB requested.
It has restarted 2 times.
Goodput is 80% (0.5 hours badput, 2.1 hours goodput).
What about a DAGMan workflow?

$ htcondor dag status 223
DAGMan Job 223.0 [simple.dag] has been running for 52 days 04:12:46.
DAG has submitted 382 individual job(s), of which:
  45 are running.
  10 are idle.
  0 are held.
  162 have completed successfully
DAG has failed nodes but will continue until all possible work is finished:
  5 nodes failed.
  10 nodes waiting to begin.
  24 nodes running.
Goodput is 92%
[#--------------------------] 34% complete.
What is this "eventlog" noun?

› *htcondor* <noun> <verb>

• "htcondor job submit", "htcondor job status", …
• "htcondor dag submit", "htcondor dag status", …
• "htcondor jobset submit", "htcondor jobset status", …
• "htcondor annex create", "htcondor annex status", …
• "htcondor eventlog read"

› Legacy tools (condor_q, condor_submit, condor_history…) not going anywhere…
What is this "eventlog" noun?

› `htcondor <noun> <verb>`
  - "htcondor job submit", "htcondor job submit"
  - "htcondor dag submit", "htcondor dag submit"
  - "htcondor jobset submit", "htcondor jobset submit"
  - "htcondor annex create", "htcondor annex create"
  - "htcondor eventlog read"

› Legacy tools (condor_q, condor_submit, condor_history…) not going anywhere…

See demo by Todd Miller
https://agenda.hep.wisc.edu/event/1733/contributions/25505/attachments/8274/9664/go
What is this "eventlog" noun?

› *htcondor* <noun> <verb>*
  • "htcondor job submit", "htcondor job status", …
  • "htcondor dag submit", "htcondor dag status", …
  • "htcondor jobset submit", "htcondor jobset status", …
  • "htcondor annex create", "htcondor annex status", …
  • "htcondor eventlog read"

› Legacy tools (condor_q, condor_submit, condor_history…) not going anywhere…
Lots of info users/admins can get from AP and EP job history files

```
$ condor_history

$ condor_history -file startd-hist

$ condor_adstash
```
Different info in event logs

000 (076.000.000) 2023-07-06 16:29:23 Job submitted from host: <...>.

001 (076.000.000) 2023-07-06 16:29:23 Job executing on host: <...>...

005 (076.000.000) 2023-07-06 16:30:19 Job terminated.

(1) Normal termination (return value 0)

Usr 0 00:00:00, Sys 0 00:00:00

Usr 0 00:00:00, Sys 0 00:00:00

0 - Total Bytes Sent By Job

0 - Total Bytes Received By Job
NEW info in event logs w/ v10.6

000 (076.000.000) 2023-07-06 16:29:23 Job submitted from host: <...>.

001 (076.000.000) 2023-07-06 16:29:23 Job executing on host: <...>...

SlotName: slot1@example.com
CondorScratchDir = "/condor/execute/dir_123"
Cpus = 1
Disk = 27093557
Memory = 128
THERE'S A KNOB FOR THAT™

ulog_execute_attrs = GLIDEIN_Site
e.g. GLIDEIN info in event logs

000 (076.000.000) 2023-07-06 16:29:23 Job submitted from host: <...>.

001 (076.000.000) 2023-07-06 16:29:23 Job executing on host: <...>...

SlotName: slot1@example.com
CondorScratchDir = "/condor/execute/dir_123"
Cpus = 1
Disk = 27093557
Memory = 128
GLIDEIN_Site = "BestOSPoolSite"
1. Your per-job (workflow?) set by the `log = some_log_file` in the submit file
2. For a DAGMan job, in the DAGman nodes.log
3. If configured, the global AP events file which has events for all jobs from all users on that AP

Using the most constrained file you need gives you efficiencies—faster to only query your own workflow log rather than global
Enter htcondor eventlog

Work in progress to read these with htcondor
e.g.

$ htcondor eventlog read my_log_file

Note new noun "eventlog" in cli
Works on all three kinds of eventlog files
$ htcondor eventlog read my_log_file

<table>
<thead>
<tr>
<th>Job</th>
<th>Host</th>
<th>Start Time</th>
<th>Evict Time</th>
<th>Evictions</th>
<th>Wall Time</th>
<th>CPU Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>79.0</td>
<td>foo</td>
<td>7/6 17:24</td>
<td>7/6 17:25</td>
<td>1</td>
<td>0+00:01:00</td>
<td>0+00:00:52</td>
</tr>
<tr>
<td>80.0</td>
<td>foo</td>
<td>7/6 17:35</td>
<td>7/6 17:36</td>
<td>1</td>
<td>0+00:01:00</td>
<td>0+00:00:52</td>
</tr>
<tr>
<td>81.0</td>
<td>foo</td>
<td>7/6 17:43</td>
<td>7/6 17:44</td>
<td>1</td>
<td>0+00:01:00</td>
<td>0+00:00:53</td>
</tr>
</tbody>
</table>
And with group-by

```bash
$ htcondor eventlog read -group-by GLIDEN_Site my_log_file
```

<table>
<thead>
<tr>
<th>Site</th>
<th>CPU Usage</th>
<th>Job Starts</th>
<th>Job Successes</th>
<th>Job Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>MWT2</td>
<td>0+01:13:00</td>
<td>59</td>
<td>57</td>
<td>2</td>
</tr>
<tr>
<td>NWICG_NDCMS</td>
<td>0+00:50:20</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>UConn-HPC</td>
<td>0+00:05:00</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>UColorado_HEP</td>
<td>0+00:51:11</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>NMSU-Discovery-CE</td>
<td>0+04:24:01</td>
<td>7</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
What other nouns should we have?

› *htcondor* `<noun>` `<verb>`
  • "htcondor ap status", "htcondor cm status", ???
  • "htcondor wallet add", "htcondor wallet remove", ???

› Others?

› What about web interface?
  • Open OnDemand for job placement?
  • Command-line noun verb exercise still useful
Submit File Enhancements

› Can make "+CustomAttributes" first class with knobs
  EXTENDED_SUBMIT_[COMMANDS|HELPFILE]:

  Instead of:
  
  +SiteJobType = "analysis"
  queue

  Just:
  
  SiteJobType = analysis
  queue

› Define job submit templates:

  use template: Matlab( mymatrix.m )
  queue
File Transfer Error Propagation

Hold Reason and Reason Codes now useful

Example of "condor_q –hold" (output directory missing on the Access Point)

Old Message:
Error from slot1@TODDS480S: STARTER at 127.0.0.1 failed to send file(s) to <127.0.0.1:50288>; SHADOW at 127.0.0.1 failed to write to file C:\condor\test\not_there\blah: (errno 2) No such file or directory

New Message:
Transfer output files failure at access point SUBMIT1 while receiving files from execution point slot3@NODE5. Details: writing to file C:\condor\test\not_there\blah: (errno 2) No such file or directory
File Transfer Enhancements

- Submit macro `preserve_relative_paths = True`
  
  `transfer_input_files = result_data/x`

  ends up creating 'result_data/x' on EP job sandbox, not 'x'.

- OSDF File Transfer Client comes with HTCSS, so can have out of the box
  
  `transfer_input_files = osdf:///foo/xxx/yyy`
Saved DAG Progress

- Added new saved progress file for a DAG in V10.5.0 that is kind of like a video game save
  - File is similar to a rescue file
  - Written at the first start of a specified node

---

```
... SAVE_POINT_FILE S1
SAVE_POINT_FILE S2 post_simulation1.save
SAVE_POINT_FILE S3 ./post_simulation2.save
SAVE_POINT_FILE S4 ../../foo/mid_analysis.save
...```

---

Example Workflow Visualized

- Sample DAG

  - `Sample.dag`
  - `Setup`
  - `Simulation 1`
  - `Simulation 2`
  - `Analysis Part 1`
  - `Analysis Part 2`
 Improve life for administrators as well…

› Administrative Quick-Start Guide

› Fast "one-line" install and configure
  • Installs all needed packages
  • Configures a secure HTCondor pool using ID tokens

› Install an up-to-date secure system including CM, AP, and EP in < 5 min
Execution Point Developments
P(partitionable) Slots Today

- pslot is never claimed, which constrains AP planning.
- AP claims a series of independent dslots made from the pslot
- AP is dependent on CM for managing pslot resources
  - CM can take resources away and offer them to other APs
Pslot Claiming

AP can now claim entire pslots
  • For a set period of time
  • Minus resources still in use by previous claimant

This will allow improved resource management
  • AP can manage pslot resources without aid or hinderance of a pool CM
    • Intelligent draining for different sized jobs
  • Less work for CM negotiator
  • CM can be unaware of users at the AP
EP advertises container runtimes available, and uses whichever one can get the job done.

Now EP does a lot of testing of container runtime and the container image to determine if errors are the systems fault or the jobs fault.

New world order:

```
container_image = /cvmfs/my/image/dir/
# Or container_image = docker://Debian
# Or container_image = myImage.sif
# Or container_image = http://xxx/image.sif
```
Containers, cont.

- First class EP support for a default container image to use
- Deal with Docker Hub banning
- Package Apptainer into HTCSS EP Product
HTCondor has long been able to
• detect GPU devices and schedule GPU jobs
• monitor/report job GPU processor utilization
• monitor/report job GPU memory utilization

Now also support for heterogenous GPUs in one server
• E.g. a server with two different models of GPU cards
• NVIDIA Multi-Instance GPU (MIG) partitioning

Currently on the radar:
• Working on concurrent jobs on one device
• Dealing with locked-up GPUs (dead slots?)
• Controlling exposure of GPU devices w/ device namespaces

Submit File Example:
Executable = foo.exe
RequestGPUs = 1
RequireGPUs =
   Capability > 7.0
Queue
First-class Backfill pslots

- Motivating Scenario: I want my GPU rich server to give priority to GPU jobs, but backfill with CPU-only jobs
- A p-slot provisioned from a shadow set of resources that tracks contention with the primary set of resources
  
  \[
  \text{SLOT\_TYPE\_<N>_BACKFILL} = \text{TRUE}
  \]
- BACKFILL slots have special attributes
  
  - BackfillSlot = true
  - ResourceConflict = "Memory, GPUs, GPU-aabbccddd"
- ResourceConflict given the names of the resources that contend with an active primary slot
Ex: Backfill the CPUs on a GPU node

```plaintext
# make a TYPE_1 primary P-slot and give it all of the resources
#
use FEATURE : GPUs
use FEATURE : PartitionableSlot(1, 100%)
SLOT_TYPE_1_START = TARGET.RequestGpus > 0

# make a TYPE_2 backfill P-slot with 90% of the shadow resources and no GPUs
#
SLOT_TYPE_2_BACKFILL = true
use FEATURE : PartitionableSlot(2, 90%, GPUs=0)
SLOT_TYPE_2_PREEMPT = size(ResourceConflict?:""') > 0

# The backfill slot should only run jobs that opt in as BackfillJob
SLOT_TYPE_2_START = TARGET.BackfillJob
```

Will likely expose 🔄 as "use policy: PreferGPUJobs"

Better Job Disk Space Management

› Config knob `STARTD_ENFORCE_DISK_LIMITS = True`
  • Create an ephemeral filesystem for the job sandbox on the EP.
  • Uses LVM or makes a loop-back file system
  • Reserves space on disk for the job
  • Enables EP to perform fast usage queries and cleanup

Couple other bits of goodness

› No longer "rename" job executable to `condor_exec.exe` w/ file transfer
› Better OOM Killer handling
HTCondor-CE Activities

› HTCondor-CE Dashboard. Admin can point their browser at the CE and see how their site is being used, e.g.:
  • Current number of active and idle glideins
  • Usage by project

› Added ability to distinguish when the target batch system is unreachable or not functioning.
  • When this is the case, HTCondor marks the resource as unavailable instead of putting impacted jobs on hold.
  • Grid ads in the collector give details about why the remote scheduling system is considered unavailable.

Plus even more…

› Data Enhancements
› Working on managing SPOOL usage with LOTMAN
   • Protection for delegated transfers
   • Checkpoint to OSDF
   • Pelican Integration in the months to come
› Security Enhancements
   • Plugin capability for token mapping
   • SSL proxy cert support without GSI
   • Need to improve token UX beyond condor_submit (e.g. dagman)
Recent OS and Architecture Additions

› Enterprise Linux 8
  • Processors:
    • Intel/AMD ("x86_64")
    • ARM ("aarch64")
    • Power PC ("ppc64le")

› Enterprise Linux 9 😊😊😊
  • Required work to support cgroups v2 and OpenSSL 3
  • Processors:
    • Intel/AMD ("x86_64")
    • ARM ("aarch64")

Future of Enterprise Linux rebuilds (aka Alma, Rocky) uncertain... 😰
Thank you!

Follow us on Twitter!
https://twitter.com/HTCondor

This work is supported by NSF under Cooperative Agreement OAC-2030508 as part of the PATH Project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.