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

European HTCondor Workshop 2025
Prague

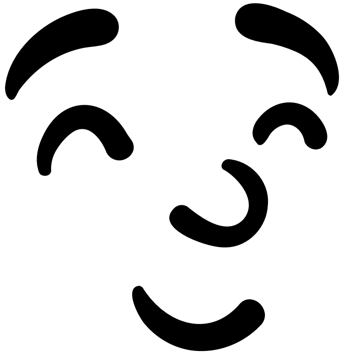
Todd Tannenbaum
Center for High Throughput Computing
University of Wisconsin-Madison

CENTER FOR HIGH THROUGHPUT COMPUTING

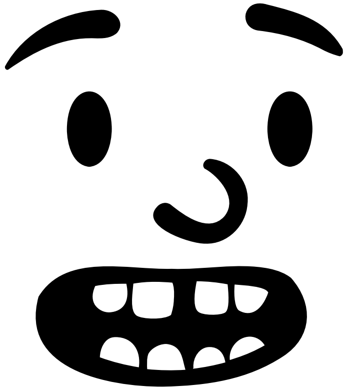
HTCondor Software Suite

PAThh PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Legend



= "Voila"



= "What's the Bad News?"

Release Numbering Scheme

- Major.minor.patch, e.g. HTCSS 24.12.2
- New Major version in Q3 each year
- Major Version number = year of release (23, 24, 25, …)
- Two release series for each major version:
  - LTS ("Long Term Support") Channel: *only bug fixes, please!*
  - Feature Channel: *gimme the latest greatest features!*
- Most recent TWO major versions are supported
  - E.g. security bug patches for two years
  - In Sept 2025 Version 25.x will be released. *Support for v24.x and v25.x; support for 23.x will end.*

Since last year…

New releases monthly containing a total of

+ 130 documented features
+ 193 documented bugfixes

Highlights on the web, details in the Manual

Highlights:

https://htcondor.org/htcondor/release-highlights/

Details:

https://htcondor.readthedocs.io/en/latest/version-history/index.html

- Documented all the new features / mechanisms that have been added at each version
- Notes about "gotchas" when upgrading from version X to Y
- condor_upgrade_check tool assists administrators upgrading an HTCondor instillation between major version (i.e. V24 -> V25)
  - Intended for use by those running LTS release of HTCondor.
  - Checks current installation and setup for well known incompatibilities and informs administrator of actions to take to resolve said issues.

Using Container Images

- Users can have their job run inside a Docker container image via

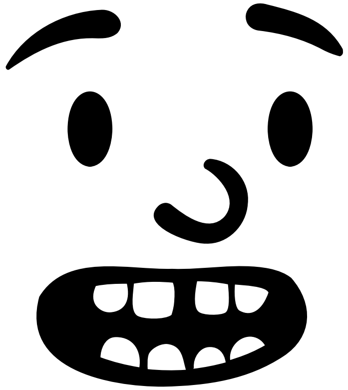
```
container_image = docker://repo:tag
```

Using Container Images

- Users can have their job run inside a Docker container image via

```
container_image = docker://repo:tag
```

- Bad News: Rate Limits!
- Unauthenticated user limits at Docker Hub
  - Max of 100 image pulls per 6-hour window
    - per IPv4 address or
    - per IPv6 /64 subnet
- Authenticated users have higher limits
- Users with a Docker subscription have no limits



New Authenticated Docker Support

- On the AP, first do:

```
docker login
```

- When placing your job, use submit command:

```
docker_send_credentials = true
```

- Get matched to a root EP
- Voila!
- *Q: How does this work?*

New Authenticated Docker Support

- On the AP, first do:

  `docker login`

- When placing your job, use submit command:

  `docker_send_credentials = true`

- Get matched to a root EP
- Voila!
- *Q: How does this work?*

  A: Tokens!

New Authenticated Docker Support, cont.

- *Q: Does this mean I can use a private Docker repository?*

New Authenticated Docker Support

- *Q: Does this mean I can use a private Docker repository?*

  A: It depends!

Can pull private images,

but Docker's cache has

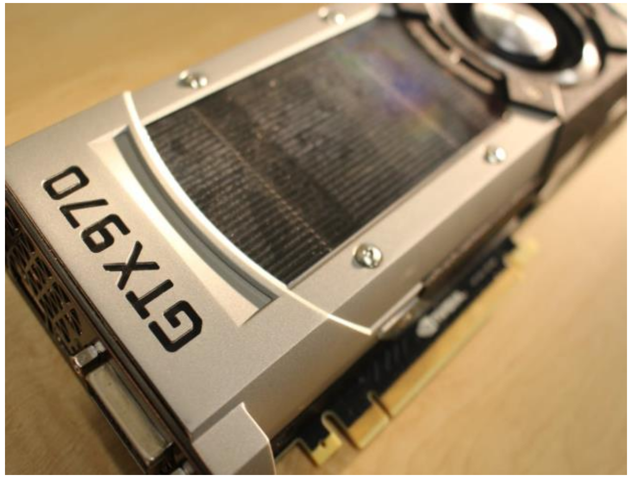
no access control to

images it caches at the EP!

Who can run the Docker CLI on the EP? Do you trust them?

GPU Scheduling Activities

- HTCondor has long been able to
  - detect GPU devices and schedule GPU jobs
  - monitor/report job GPU utilization
- New: Support for heterogenous GPUs in one server
  - E.g. a server with two different models of GPU cards
  - NVIDIA Multi-Instance GPU (MIG) partitioning
- New: Jobs can only see the GPU devices assigned by the EP by using cgroup device namespaces
- New: Support for AMD ROCm
- More first-class keywords in submit file



Submit File Example:

```
Executable = foo.exe
RequestGPUs = 1
gpus_minimum_capability = 7.0
gpus_minimum_memory = 16GB
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

```
SLOT_TYPE_<N>_BACKFILL = TRUE
```

- BACKFILL slots have special attributes

```
BackfillSlot = true
ResourceConflict = "Memory, GPUs, GPU-aabbccdd"
```

- `ResourceConflict` given the names of the resources that contend with an active primary slot

CENTER FOR HIGH THROUGHPUT COMPUTING

HTCondor

Ex: Backfill the CPUs on a GPU node

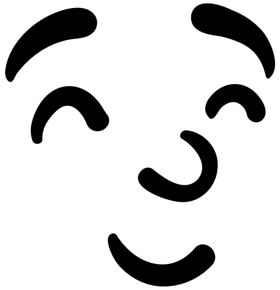
```
# make a TYPE_1 primary P-slot and give it all of the resources
#
use FEATURE : GPUs
use FEATURE : PartitionableSlot(1, 100%)
SLOT_TYPE_1_START = TARGET.RequestGpus > 0

# 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
```

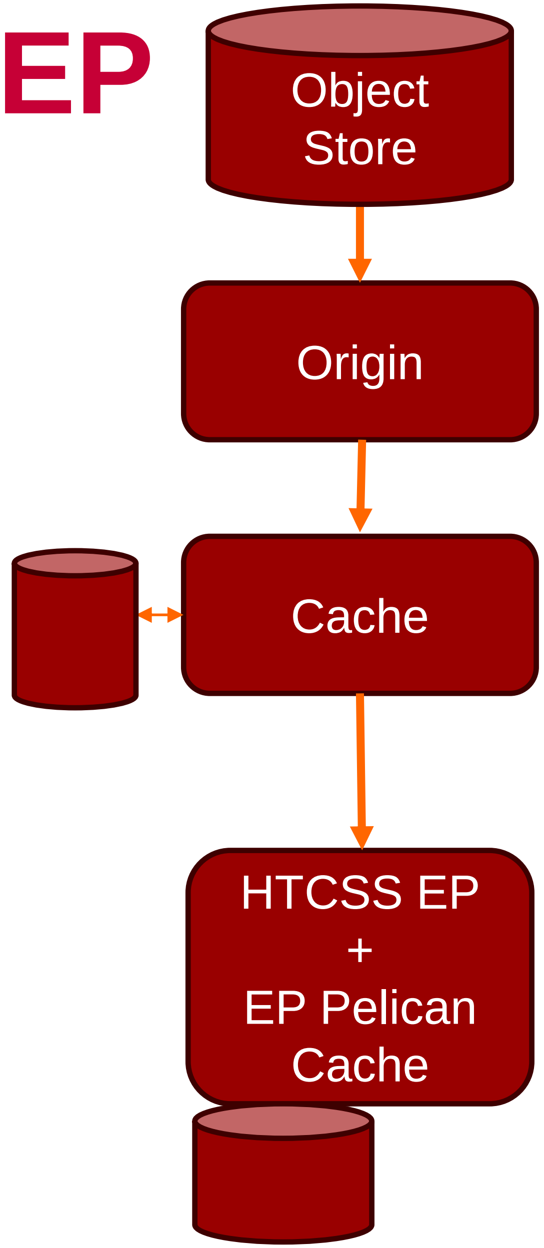
Pelican Local Cache on the EP

- Cache files coming from OSDF (Pelican) on the EP

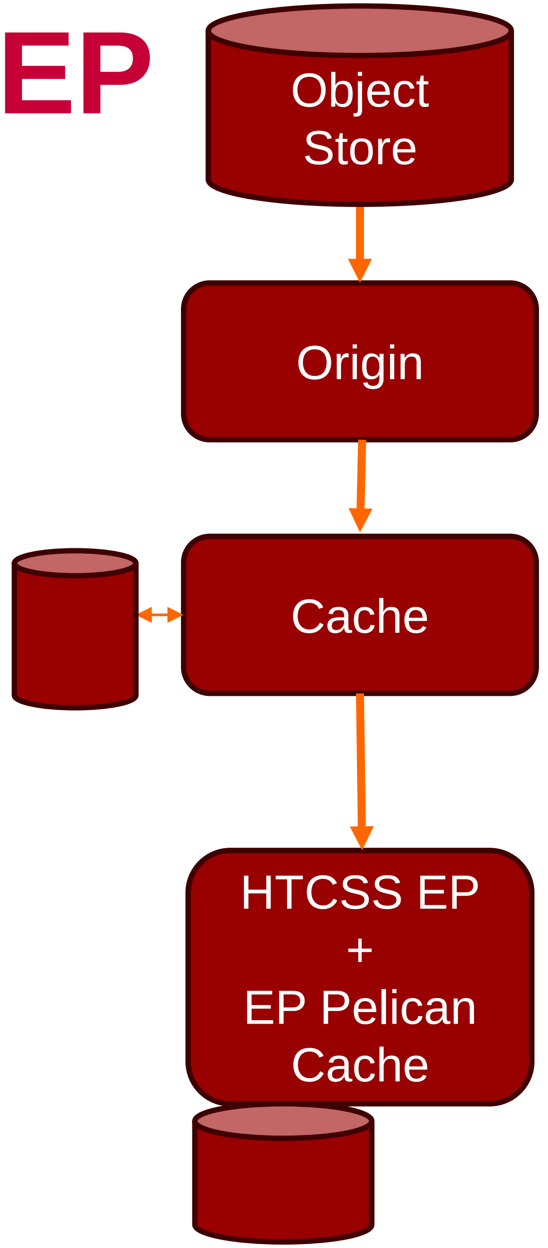
```
use feature:PelicanCache(size of cache)
```

Works with a glidein EP as well. *Voila!!*





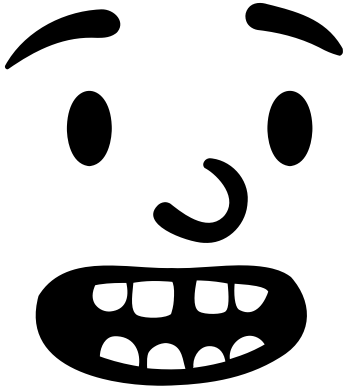
Pelican Local Cache on the EP



› Cache files coming from OSDF (Pelican) on the EP

```
use feature:PelicanCache(size of cache)
```

Works with a glidein EP as well. *Voila!!*

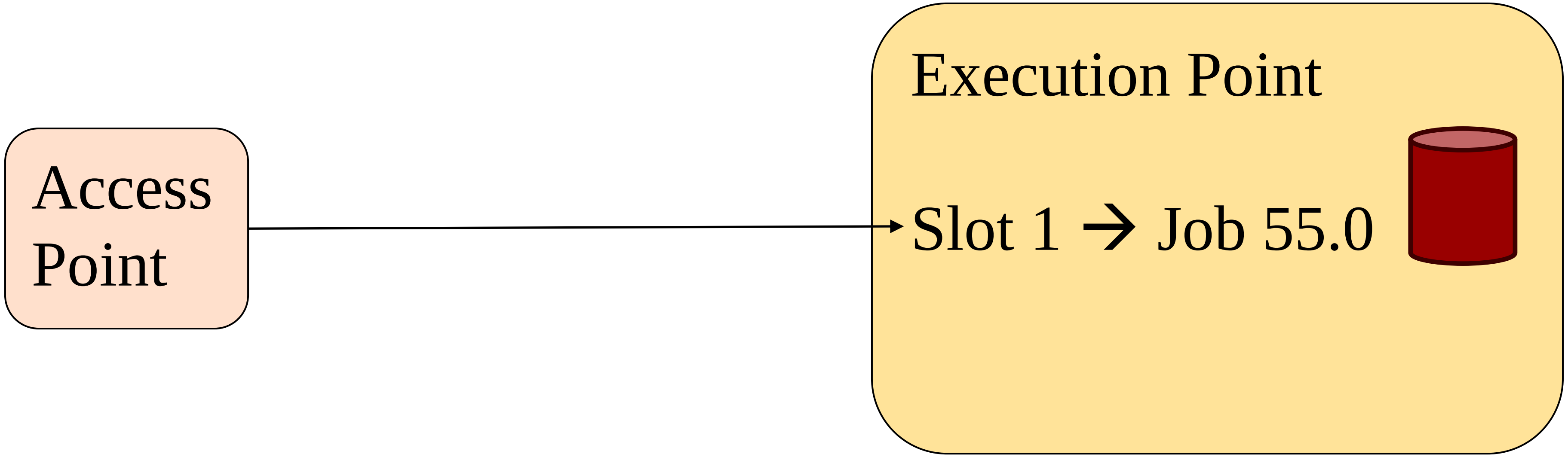


› What's the Bad News?
- Admin must fragment EP Disk Space
- What about data transferred without Pelican?
  - Files from the AP (HTCondor File Transfer / CEDAR)
  - Files from Web Server URLs / HTTP
- Data Stored twice! Need double the EP disk space
  - One copy in EP Pelican Cache, then copied to job scratch dir
- AP knowledge not captured

Share Common Files

- You can try it out as of HTCSS 24.9!

  Shares explicitly-listed common files between jobs in the same job list (cluster) running at the same time.
    - Transfers them only once.
    - Makes only one copy on-disk.
    - Controlled by the AP.
    - Details in talk Fri @ 9:20am by Brian Bockelman

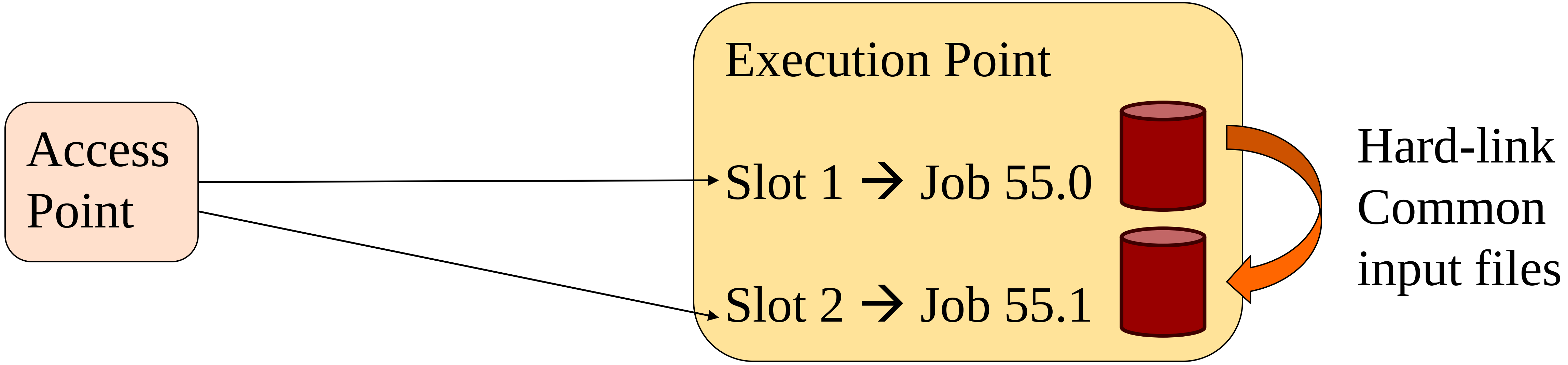


Share Input Common Files

- You can try it out as of HTCSS 24.9!

  Shares explicitly-listed common files between jobs in the same job list (cluster) running at the same time.

  - Transfers them only once.
  - Makes only one copy on-disk.
  - Controlled by the AP.
  - Details in talk Fri @ 9:20am by Brian Bockelman



Speaking of giving control to the AP…

We want the AP to get complete *information* from the EP, but no *decisions.*

Let the AP draw its own conclusions, and then tell the EP what to do.

New Job Attribute: VacateReason[Code]

- Set whenever execution attempt is interrupted (in job ad and and evict event)
- VacateReason
  - Human-readable string
- VacateReasonCode

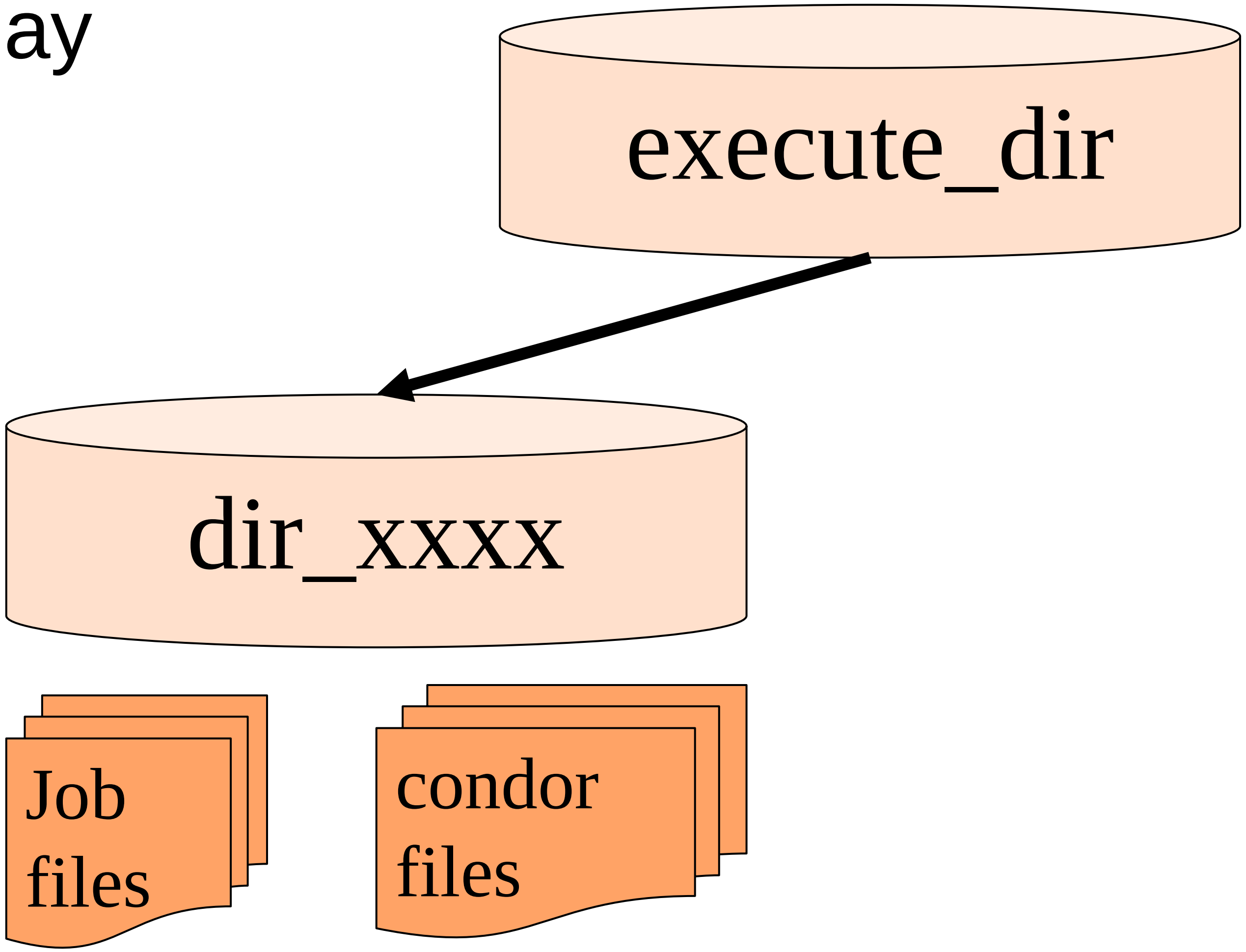
  By convention:
  - 1-999: There was a problem with the job
  - 1000+: Interruption not the job’s fault

Job Cool Down

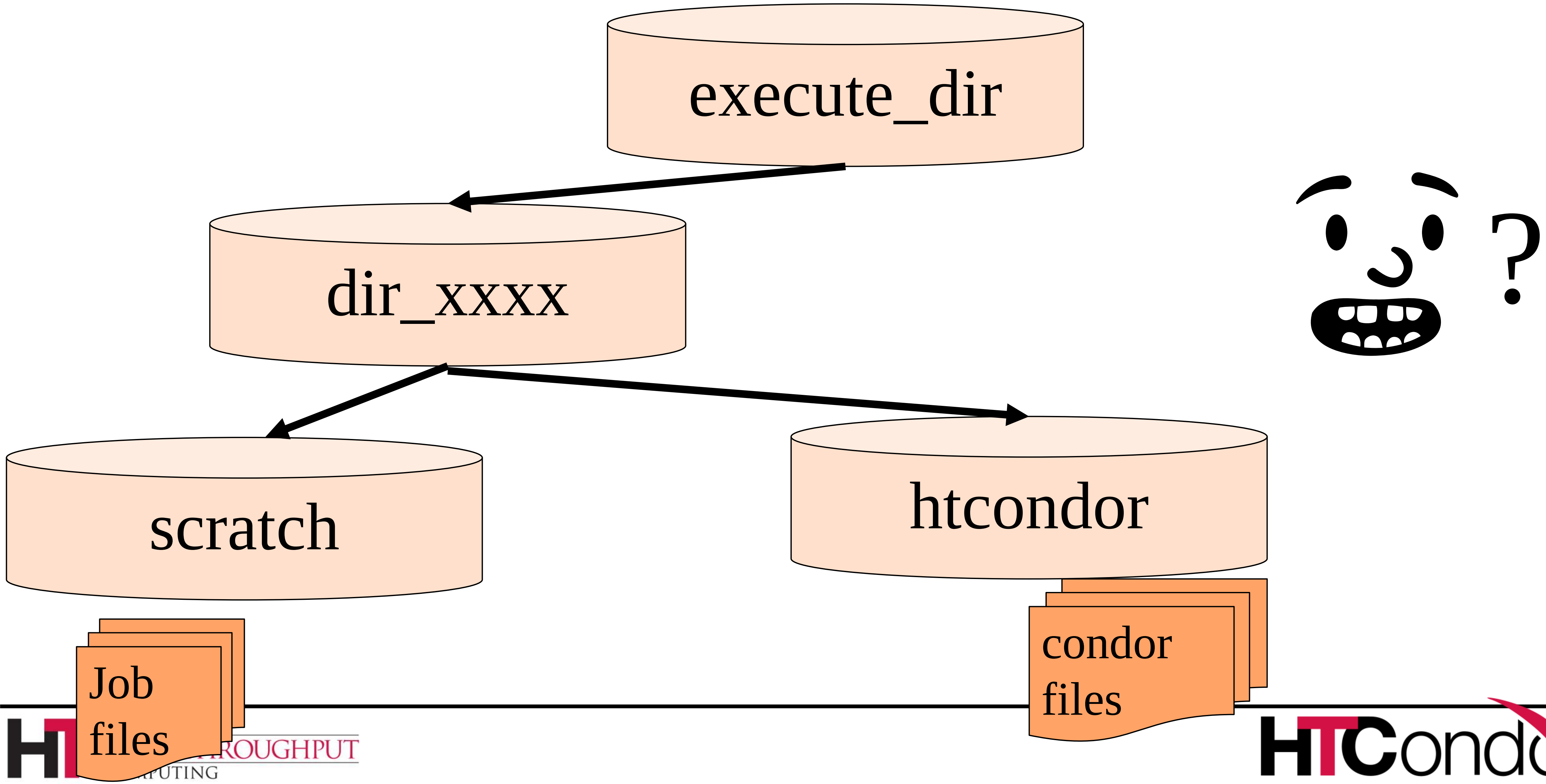
- SYSTEM_ON_VACATE_COOL_DOWN
- ClassAd expression evaluated at the AP in the context of a job that is leaving Running state (for Idle or Held)
- At vacate, if >0
  - Ignore EP's request to hold job
  - Don't run job for given number of seconds
- JobCoolDownExpiration
  - Time when most recent cool-down ends
- NumJobCoolDowns
  - Count of cool-downs for this job

STARTER_NESTED_SCRATCH

- Today



STARTER_NESTED_SCRATCH



EP Disk Reservation and Enforcement

Allows HTCondor to utilize the Linux Logical Volume Manager to provide isolated work environments and strong disk usage enforcement for jobs executing on EP's.

- No LVM ? EP will create a loopback filesystem on the fly!
- Introduced in v23.5.2
- Requires an HTCondor EP running on Linux as root
- Details Friday @ 10am

Look in the Manual for config knob STARTD_ENFORCE_DISK_LIMITS

No Job Networking

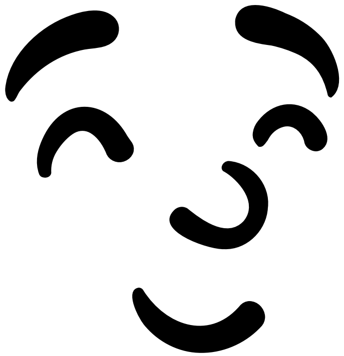
- EP admin can set
  `NO_JOB_NETWORKING = true`
- Job will only have loopback, no other network
- But condor daemons will have network
- Advertises `HasJobNetworking` for matchmaking

Cgroup Management w/o root

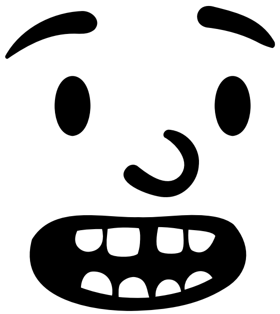
- If the base system grants permission, a glidein EP can create cgroups w/o root!
- Off by default, ENABLE_CGROUP_WITHOUT_ROOT enables, will be default soon

- HTCondor creates writeable cgroups by default
- Can also launch an EP via Docker and create writeable cgroups

At long last, Enterprise Linux 9 support in HTCSS is happy!

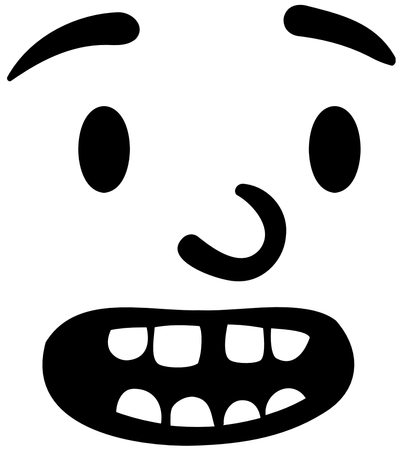


At long last, Enterprise Linux 9 support in HTCSS is happy!



- But it took a long time

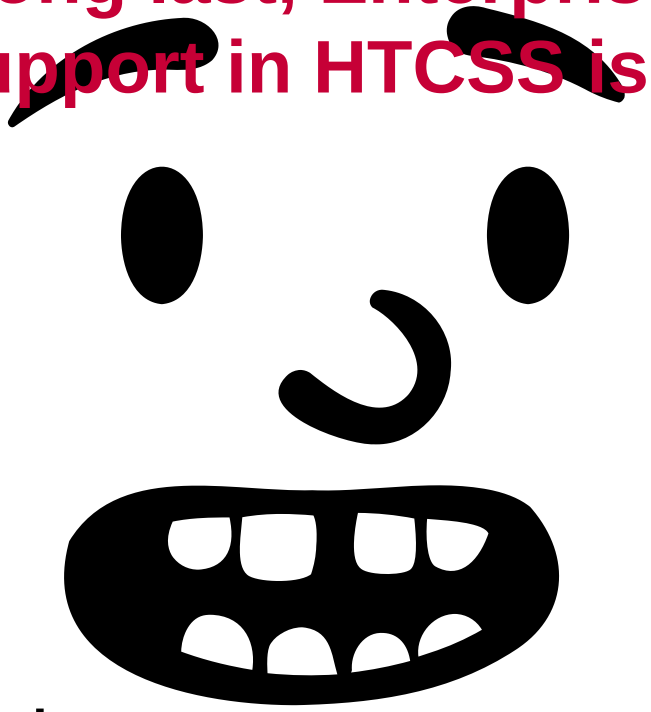
At long last, Enterprise Linux 9 support in HTCSS is happy!



- But it took a long time
- And several sites ran into problems along the way

  (mainly with memory tracking due to cgroups -> cgroup 2)

At long last, Enterprise Linux 9 support in HTCSS is happy!



- But it took a long time
- And several sites ran into problems along the way
  (mainly with memory tracking due to cgroups -> cgroup 2)
- And couple months ago, Enterprise Linux 10 was released!

What went wrong?
How can we do better with EL 10?

Reminder: Translational (Computer) Science

What went wrong?
How can we do better with EL 10?

Reminder: Translational (Computer) Science

Tracing and its results…

Couple years ago talk about tracing..

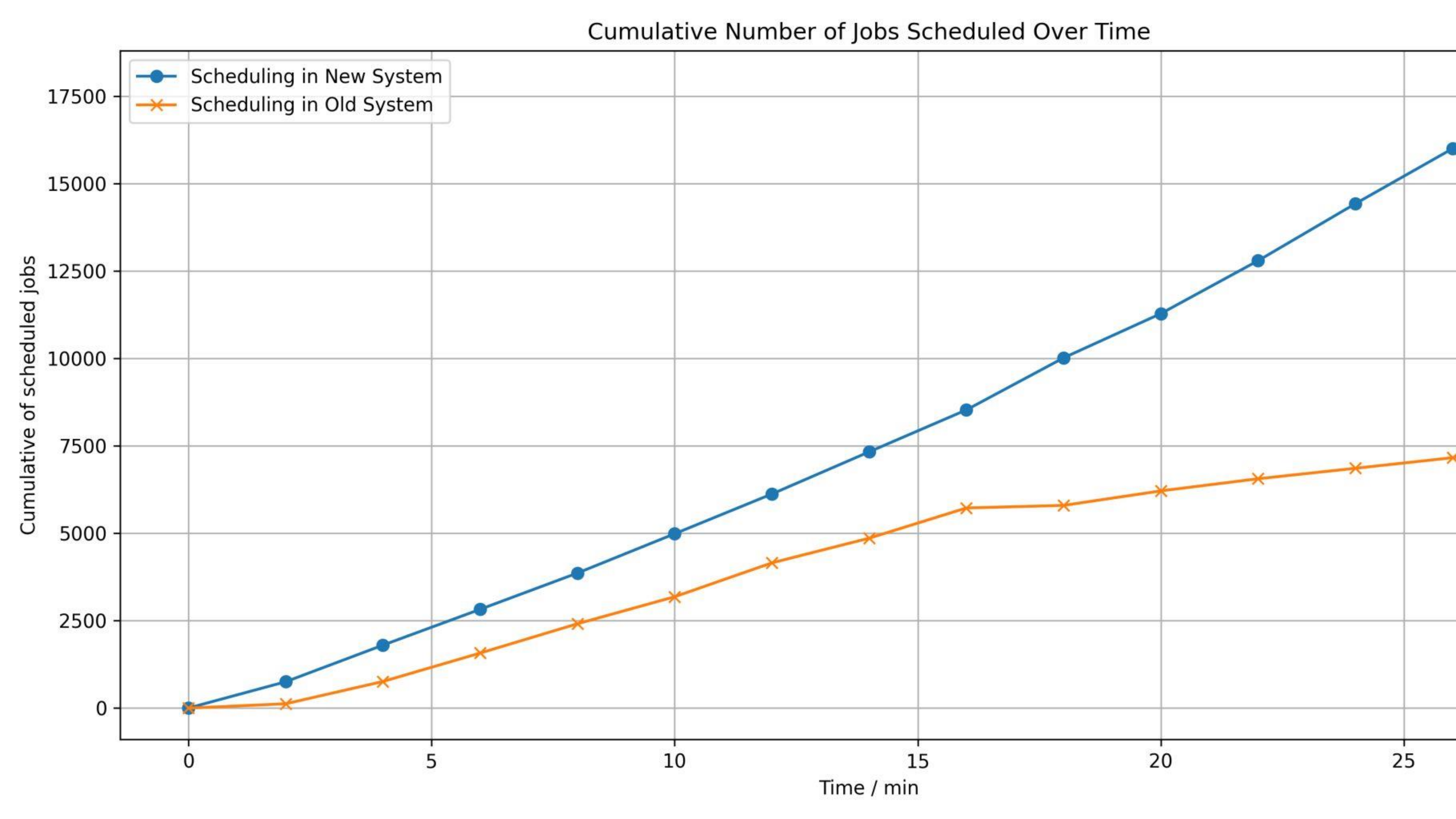
We found schedd blocking writing to pipe..

(and fixed it)

perf trace command -- after

```
# perf trace -p 3437472 -duration 10 -e '!futex'
5.3 (21.412 ms): read(fd: 3</staging/big_file>, …) = 8192
7.5 (17.578 ms): read(fd: 3</staging/big_file>, …) = 8192
7.7 (89.972 ms): read(fd: 3</staging/big_file>, …) = 8192
8.6 (28.883 ms): read(fd: 3</staging/big_file>, …) = 8192
```

And the job finished in roughly 20 minutes!



Cumulative Number of Jobs Scheduled Over Time
Scheduling in New System
Scheduling in Old System
Cumulative of scheduled jobs
Time / min
0
2500
5000
7500
10000
12500
15000
17500
0
5
10
15
20
25

DAG Checker Tool: condor_dag_checker

- Good for checking the validity of a DAG file and getting some statistics back without having to place the DAG and dig through the DAGMan debug log file.
  - Check DAG file for various failures such as invalid DAG command syntax, referencing undefined nodes, cyclic dependencies.
  - Get statistics about a DAG such as the count of nodes and arcs in a given DAG.

From command-line…

- Before…

```
$ MY_DIR=dir my_command one two three > out 2> err
```

From command-line… to submit file

```
Executable = my_command
Arguments  = one two three
Output     = out
Error      = err
Environment= MY_DIR=dir

queue
```

New “Shell” submit command

› With shell…

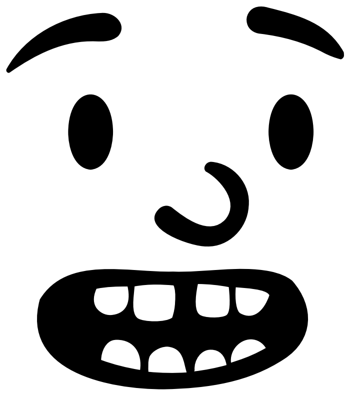
```
$ MY_DIR=dir my_command one two three > out 2> err
```

New “Shell” submit command

```
shell = MY_DIR=dir my_command one two three > out 2> err

queue
```

Dealing with Job Memory

- Current default behavior: Hold jobs that use more memory than requested.



Dealing with Job Memory

- New *first class* way for users to retry with more memory.
- Examples:

Ex 1

```
RequestMemory = 1 GB
RetryRequestMemory = 4 GB
```

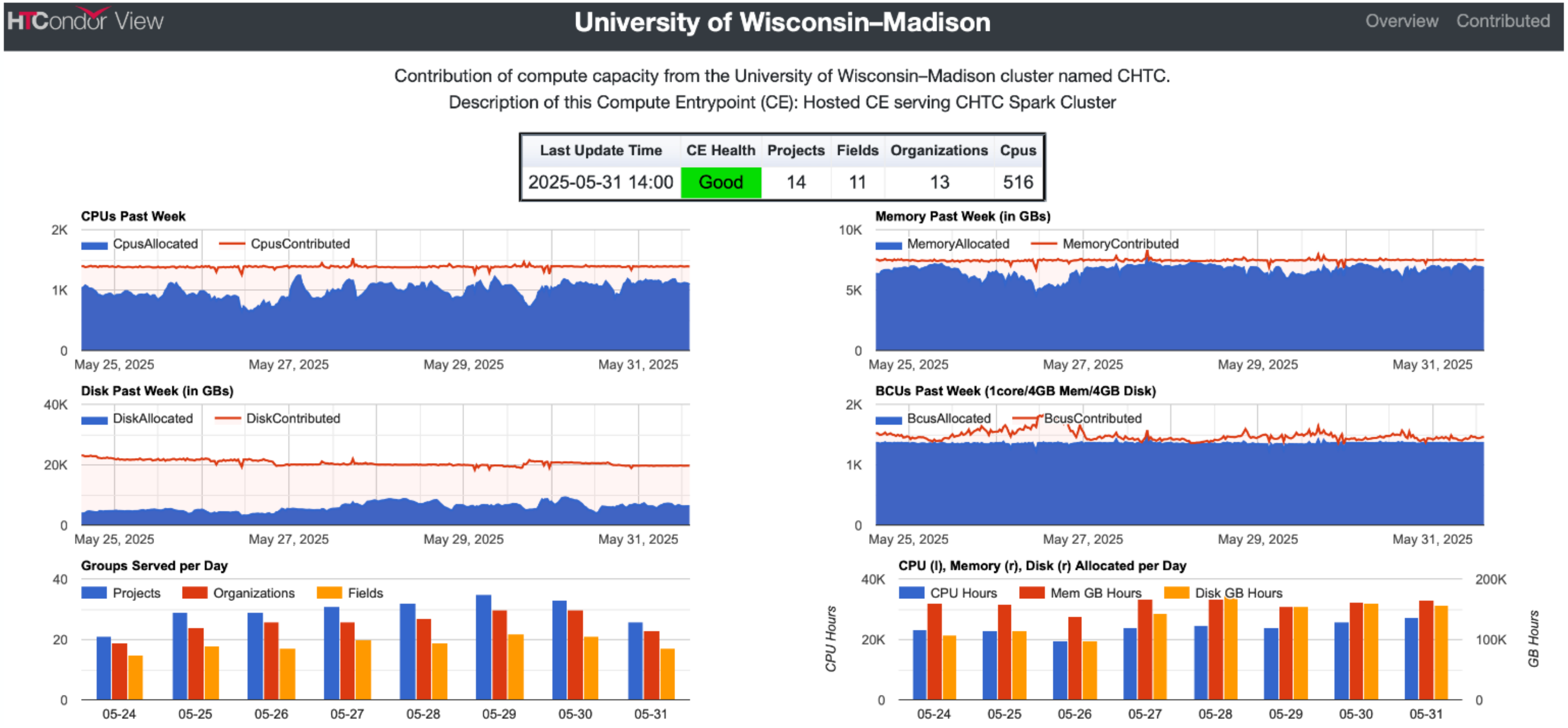
Ex 2

```
RequestMemory = 1 GB
RetryRequestMemory = 4 GB, 16 GB
```

Ex 3

```
RequestMemory = 1 GB
RetryRequestMemoryIncrease = RequestMemory * 4
RetryRequestMemoryMax = 16 GB
```

CE Dashboard



HTCondor View **University of Wisconsin–Madison** Overview Contributed

Contribution of compute capacity from the University of Wisconsin–Madison cluster named CHTC.
Description of this Compute Entrypoint (CE): Hosted CE serving CHTC Spark Cluster

Last Update Time	CE Health	Projects	Fields	Organizations	Cpus
2025-05-31 14:00	Good	14	11	13	516

CE Dashboard
to
CM Dashboard
to
Prometheus Exporter

Prometheus

The condor_who Tool in Glideins

Feedback: At times, site admins want to know about OSPool researcher jobs

- TJ (HTCSS developer) updated condor_who tool to work with glideins
- Sample output (edited to fit, etc.):

PROJECT	USER	AP_HOSTNAME	JOBID	RUNTIME	MEMORY	DISK	CPUs	EFCY	PID	STARTER
KOTO	****	ap23.uc.osg-htc.org	27872550.7518	0+00:47:33	2.0 GB	6.0 GB	1	0.99	3000808	2999526
MIT_submit	****	submit06.mit.edu	764122.27	0+00:02:01	1024.0 MB	2.3 MB	1	0.99	3169488	3168950
PixleyLab	****	ap20.uc.osg-htc.org	35396737.1095	0+06:04:38	4.0 GB	4.0 GB	1	0.99	2052820	2049582
SSGAforCSP	****	ap21.uc.osg-htc.org	17594309.420	0+16:48:15	14.8 GB	4.0 GB	1	0.99	2802678	3838451
PSU_Lynch	****	ap20.uc.osg-htc.org	35421713.0	0+05:24:08	20.0 GB	2.0 GB	1	0.99	2302554	2301237
BiomedInfo	****	ap20.uc.osg-htc.org	32684753.98178	0+00:22:17	15.6 GB	32.0 GB	1	0.99	3066645	3066091
PixleyLab	****	ap20.uc.osg-htc.org	35396944.129	0+05:52:25	4.0 GB	4.0 GB	1	0.99	2199648	2165608

New defaults coming up might be surprising

- Python API (bindings) v1 being dropped -> long live v2 ! Question:

```
import htcondor

import htcondor2 as htcondor
```

CENTER FOR HIGH THROUGHPUT COMPUTING

HTCondor

New defaults coming up might be surprising

- Python API (bindings) v1 being dropped -> long live v2 ! Question:

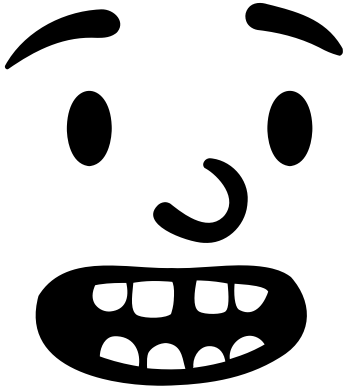
```
import htcondor
import htcondor2 as htcondor
```

- HTCondor CE Routes must use “new” syntax
- System Swap space (virtual memory) will not be used for jobs on the EP by default
- Dropping support for multiple queue statements in a single submit file
  (Use queue foreach, etc.)
- Partitionable Slots enabled by default (instead of static partitioning)
- The job’s executable will no longer be renamed to ‘condor_exec.exe’
- GPU discovery is enabled on all Execution Points by default

Last but not least…

A public service announcement…

Y 2038 Problem!



January 19th 2038

Y 2038 Problem!

- Pre 24.x, daemons hung staring > Y 2038
- Classad date functions fixed:
- In 23.x

$ classad_eval '[]' 'formatTime(int(abstime("2025-01-01")) + (13 * 365 * 24 * 3600))'

"Tue Dec 29 00:00:00 2037"

$ classad_eval '[]' 'formatTime(int(abstime("2025-01-01")) + (14 * 365 * 24 * 3600))'

"Sat Nov 22 17:31:44 1902"

Y 2038 Problem!

- Pre 24.x, daemons hung starting > Y 2038
- Classad date functions fixed:
- In 24.x

$ classad_eval '[]' 'formatTime(int(abstime("2025-01-01")) + (13 * 365 * 24 * 3600))'

"Tue Dec 29 00:00:00 2037"

$ classad_eval '[]' 'formatTime(int(abstime("2025-01-01")) + (14 * 365 * 24 * 3600))

"Wed Dec 29 00:00:00 2038"

Why do you care?

- Need to work on future dates now
- Software: Financial, scientific, security…
- Firmware: Problem with embedded systems,
  - Card reader
  - Physical security systems
  - Etc, etc, etc.
- Let's fix these now via RFP process

Thank You!

Please add your institution to our world map of HTCondor Users at:
https://htcondor.org/user-map
and click "Add Your Institution" on upper right

PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

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.

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
…
```

New Job Status

$ 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.

[###########========----------------] 34% complete.

CENTER FOR HIGH THROUGHPUT COMPUTING

HTCondor

HTCondor Python Bindings Version 2

- The version 1 bindings depend on an unsupported library (boost.python), so we needed to do something to make sure the bindings would remain available.
- Bindings are intended to be generally compatible; `import htcondor2 as htcondor` should mostly just work.
  - However: We removed some of the things marked as deprecated in the version 1 bindings, and will be deprecating a few other APIs that are not widely used
- Win for users: Compared to version 1, the new version 2 bindings will be much less picky about the specific version of Python interpreter being used ☺

Simpler User Experience for jobs w/ Tokens:

Let the AP fetch tokens from issuers on behalf of the user