US ATLAS HTCondor dev meeting notes:

Useful Links:
Indico page for event: https://indico.bnl.gov/event/25916/
Zoom link: https://bnl.zoomgov.com/j/1614360980?pwd=AQm6x3reOaNtEze9H7GjadACjvWaBE.1
Condor release plan: https://htcondor.org/htcondor/release-plan/
Contact us page for HTCondor - Link to Page

26 Nov 2025

- Quantize bug?
- EP became black hole on condor upgrade (acas0608)
- How to set tmp to be inside of condor scratch directory for jobs?

15 Oct 2025

- User job executable ended up in uninterruptible sleep, slot was “poisoned” because cgroup could not be destroyed
    - STARTD_LEFTOVER_PROCS_BREAK_SLOTS

23 July 2025:

- Can use this construct in routes/transforms if you want the biggest/smallest of a number in an array (useful for setting a floor/ceiling), or in my case to bully the single core Low memory jobs into requesting a bit more memory, while still letting the bigger requests pass through unbothered:

    max({RequestMemory, 1500}) ← Use this one
    min({RequestMemory, 1500}) ← don't use this in this case, but this construct exists and can be used elsewhere

    min/max passed an array, put in an EVALSET, not SET

    Useful to keep a copy/record of the original request if overriding:
    Copy RequestMemory ActualRequestMemory

--

- Command useful for finding CE jobs that are Completed (status 4), but for some reason are still in the queue, but not running/held/idle/removed or otherwise (zombies!?):

    condor_ce_q -constraint "JobStatus == 4" -constraint "EnteredCurrentStatus < (time() - 600)" -af:tj 'FormatTime(EnteredCurrentStatus)'

- Command to view CE routes and pre/post transform:

    condor_ce_job_router_info -config

- JIRA ticket for pilot cgroup memory limits: https://its.cern.ch/jira/browse/ATLASPANDA-1251

25 June 2025:

Condor CE issues (Zach)

- Completed jobs are sticking around for a very long time in the condor_ce_q, and this is causing issues
- Might need a followup meeting to view logs and such with Jamie

Setting CONDORCE_MAX_JOBS to lower number works with condor_reconfig, but going to bigger number seems to need a daemon restart?

Mail list:
usatlas-wbs23-l@lists.bnl.gov
https://lists.bnl.gov/sympa/info/usatlas-wbs23-l

28 May 2025:

Scaling issues for Kaushik

- Possible issue? Throttle limit on number of jobs per second, put in to limit queries that can kill slurm clusters
- Fred reports that Kaushik may have solved the problem
    - Kaushik - there was a (soft) limit on how many jobs on average are sent by Harvester. Increasing that limit solved the main problem.
- COMPLETED_JOB_EXPIRATION was changed (lowered?) from 30 to 10
- GRIDMANAGER_JOB_PROBE_RATE condor parameter to be changed (default 5) can probably set it to 20 or even higher
- OSG Ticket for this issue: https://support.opensciencegrid.org/support/tickets/public/40db7c0ac0faace4e90d56c5cb12da8101460f9540d8dde2d5c3b657c1f46f7d

30 Apr 2025:

Doug: set up a condor cluster where CM, CE is at BNL and workers are located elsewhere (SLURM HPCs)

- https://htcondor-wiki.cs.wisc.edu/index.cgi/wiki?p=RunCmsJobsAtBsc
- https://docs.google.com/document/d/1GGOc3pgidfv_qunaKqzjbvKPF8CD_RF30KkqMtwBC0U/edit?tab=t.0#heading=h.jlzxkv88ic14
- Need to follow up in subsequent meeting

Regarding schedd's not running enough jobs, there are a few potential bottlenecks we've seen

- MAX_RUNNING_JOBS is a potential first hard limit (check this)
- Potentially running out of system memory (plan for ~2MB per running job) for example, if your target is 15k running jobs from a schedd, plan for 30GB of RAM for the AP
- On a higher scale (~29k or so running, depending on your linux system defaults or config) you may hit default ephemeral port limits. You can check port range with `cat /proc/sys/net/ipv4/ip_local_port_range`
- I have also seen users accidentally set an artificial limit of running jobs on themselves in their JDF. This was especially hard to find because it was transient and seemingly only affected one person

Cgroups (Thanks Aiden)

```
BASE_CGROUP = htcondor
CGROUP_IGNORE_CACHE_MEMORY = true
CGROUP_MEMORY_LIMIT_POLICY = hard
CGROUP_POLLING_INTERVAL = 5
```

2 Apr 2025:

On EP drain, it is possible to allow backfill jobs to fill the time during the drain

Yum update on condor package will set the EP to "retirement" status, effectively draining jobs

-drain Option to condor_off that tells the master if you have a startd to end, do it with a drain command (-deadline option does not work) canceling condor_off doesn't cancel draining

5 Mar 2025:

Cgroups discussion: Email from Aidan:

I wanted to suggest another topic for discussion regarding jobs implementing cgroups. I created a program [1] that spawns a child process, assigns it to a cgroup with a memory limit, and makes the child process allocate a specified amount of memory before sleeping, freeing the memory, and exiting. When testing the program on the mwt2 condor cluster (24.0.2) I have observed some interesting behavior. When the child process exceeds the cgroup memory limit condor evicts the job and puts it on hold. This will occur even if the total memory being used by the job is less than the value set in the request_memory attribute of the submit file. The job log claims that the job used the exact amount of memory set in the request_memory attribute i.e

Error from slot1_9@uct2-c581.mwt2.org: Job has gone over cgroup memory limit of 4096 megabytes. Last measured usage: 4096 megabytes. Consider resubmitting with a higher request_memory.

But I do not believe that is an accurate account of how much memory the job is actually using. Despite the job being evicted and put on hold however, the parent process (which is in a separate cgroup as the child process), is kept alive and able to exit on its own. I am not sure how long the parent process is kept alive after the job has been evicted but I have tested up to 60 seconds after the child process has been killed due to memory limits and the parent process has been able to exit on its own and files produced by the job (stdout) have been transferred back to the submit host.

So what I wanted to discuss is if this is the expected behavior of condor and potential issues it could cause with jobs (i.e the pilot) implementing cgroups.

[1] https://github.com/arosberg/memory_allocator

- possible bug found? TJ will follow up with Greg and others
- Files transferred after is expected behavior

RESERVED_MEMORY knob, memory set aside for not condor

8 JAN 2025:

Ways to kill job without killing pilot using cgroup subgroups - Fred and team working on this with Greg

Opensearch v2 support included in condor 24.0.3+

Condor ssh to job:
Condor_starter creates sshd subprocess (magic) sends traffic through starter/shadow connection. Security for this is condor security (Greg is expert on this)

Pnfs issue:
Pnfs inaccessible from within job: looks like user's jdf was missing something required for the auth (X509UserProxy)

Job router:
HTCondor-CE job router conversion tool: problem when running from versions of condor that are too low (no output). Confirmed working in condor 24

condor_transform_ads -convert:file old_route_file > new_route_file

I found in some cases the “JOB_ROUTER_ROUTE_NAMES = <names>” line was omitted from the converted file, so I needed to include that line in the final version of the file. Names should be listed in order they are evaluated, order of the route blocks themselves does not matter.

11 DEC 2024:

This is what we are doing at MWT2
# SIGTERM to kill jobs, SIGKILL after 5 minutes
GRACEFULLY_REMOVE_JOBS = true
MachineMaxVacateTime = 5 * 60

# cgroup additions for limiting memory to 1.1x the job request for ATLAS and 3x for non-ATLAS
CGROUP_MEMORY_LIMIT_POLICY = custom
CGROUP_HARD_MEMORY_LIMIT_EXPR = ifThenElse(regexp("usatlas[1-4]", Owner), 1.1 * RequestMemory, 3 * RequestMemory)
CGROUP_SOFT_MEMORY_LIMIT_EXPR = ifThenElse(regexp("usatlas[1-4]", Owner), 1 * RequestMemory, 1.1 * RequestMemory)

MWT2 testing of cgroups:

1. sudo su usatlas1
2. condor_submit memory_allocator.submit

condor_allocator.submit contains:

```
universe = vanilla
executable = memory_allocator
arguments = 1500 30
request_memory = 1024M
log = memory_job.$(Cluster).$(Process).log
output = memory_job.$(Cluster).$(Process).out
error = memory_job.$(Cluster).$(Process).err
should_transfer_files = yes
when_to_transfer_output = ON_EXIT_OR_EVICT
transfer_executable = True
JobPrio = 100000
```

```
requirements = regexp("cit2", Machine)

queue
```

The executable arguments are:

1. 1500 is the number of MiB to allocate.
2. 30 is the number of seconds to wait after the memory is allocated before exiting
    a. There is a signal handler that will log a message on SIGINT or SIGTERM and wait the same amount of seconds before exiting

The memory request is 1024 MiB.
The requirement to run on a machine with a name containing cit2 ensures that the job runs on a server that is updated to condor 24.0.2.

Judith put the cgroups config related to memory above. The entire contents of /etc/condor/config.d/02-cnode.conf is:

```
use role:execute
use feature:partitionableslot

MWT2_CpuUsed    = int((CondorLoadAvg / TotalLoadAvg) *
(ifthenelse((TotalLoadAvg < TotalCpus), TotalLoadAvg, TotalCpus)) *
100) / 100.0
MWT2_CpuUsage   = ifthenelse(((TotalLoadAvg > 0.0) && (Activity !=
"Idle")), MWT2_CpuUsed, 0)
MWT2_CpuExceeded  = (MWT2_CpuUsage > (Cpus + 0.8))
MWT2_CpuMemory  = int(TotalMemory / TotalCpus)

START = TRUE
HAS_CVMFS = TRUE
TRUST_UID_DOMAIN = TRUE

STARTD_ATTRS = $(STARTD_ATTRS) HAS_CVMFS MWT2_CpuUsed MWT2_CpuUsage
MWT2_CpuExceeded MWT2_CpuMemory

# SIGTERM to kill jobs, SIGKILL after 5 minutes
GRACEFULLY_REMOVE_JOBS = true
MachineMaxVacateTime = 5 * 60

# cgroup additions for limiting memory to 1.1x the job request for
ATLAS and 3x for non-ATLAS
CGROUP_MEMORY_LIMIT_POLICY = custom
CGROUP_HARD_MEMORY_LIMIT_EXPR = ifThenElse(regexp("usatlas[1-4]",
Owner), 1.1 * RequestMemory, 3 * RequestMemory)
```

```
CGROUP_SOFT_MEMORY_LIMIT_EXPR = ifThenElse(regexp("usatlas[1-4]",
Owner), 1 * RequestMemory, 1.1 * RequestMemory)

DISABLE_SWAP_FOR_JOB = true

IGNORE_LEAF_OOM = false
```

Opensearch / Adstash

24.0.3/ 23.0.19 includes some improvements to opensearch 2.0 implementation

- Possible to run a condor 24.0.3 vm with adstash to talk to 23.X cluster

New for late 23.X 24+ in addition to machine ad they now have singular ad for each EP startd daemon ad which has aggregate view of whole machine

Looking for overloaded slots

The ATLAS production system occasionally gets two or more processes running on a slot. This is the script I wrote to detect these bad boys:

```
#!/bin/sh

EffCut=1.4
TimeCut=3600
NumberOfLoops=60
SleepTime=120

touch large_cpu_loop.txt
for i in $(eval echo "{1..$NumberOfLoops}")
do
  date >> large_cpu_loop.txt
  condor_q -global usatlas1 usatlas3 -run \
    -attributes ClusterId GlobalJobId RemoteHost CpusUsage
     CpusProvisioned JobCurrentStartExecutingDate ServerTime \
    -autoformat ClusterId GlobalJobId RemoteHost CpusUsage
     CpusProvisioned JobCurrentStartExecutingDate ServerTime | \
  sort -n | \
  awk -v effcut=$EffCut '$7-$6 < timecut {next} $5 ~ /[a-z]/ {next}
$5 == 0 {next}  $4/$5 > effcut {print $2 " " $3 " " $4/$5}' \
  >> large_cpu_loop.txt

  sleep $SleepTime
```

```
done
```

Any suggestions, improvements, corrections sent luehring@iu.edu are welcome.

Other/ Misc.

Can make use of backfill slots for lower priority jobs that get eviction when higher priority jobs come through. Progress will be lost on evicted jobs

HTCONDOR release schedule
HTCondor Release Plans

Contact us page for HTCondor - Link to Page