Controlling Your Site with Condor and Cgroups

Brian Bockelman

19 March 2012

What's Wrong With your Site?

- You may not know it, but your site is due for a tune-up:
  - It's fairly easy for a job's **processes to escape** the batch system.
  - Jobs can **write files on disk** that aren't cleaned by the batch system.
  - The **accounting statistics** were designed for a 32-bit world.
  - **Resource management** is in the cavemen days.
- This talk covers what we do with Condor to fix these!

World Writable Directories

- All files created by the job should disappear when the job ends.
    - Incredibly hard to do when there are world-writable directories: which files came from which job?
- Two tricks are available:
    - A *filesystem namespace* allows for a set of processes to have a unique set of mounts.
    - A *bind mount* will make part of the filesystem available elsewhere.

In Condor 7.7.5

World-Writable Directories

Available RHEL5

- Condor will create a filesystem namespace for the job.
  - Then, it will bind mount /tmp and /var/tmp into $_CONDOR_SCRATH_DIR.
  - Writes the job performs into /tmp will actually go into the scratch dir.
  - Feature is called MOUNT_UNDER_SCRATCH, and can remove *all* world-writable directories.
- All files will not be visible by other jobs, and be deleted after the job ends.

In Condor 7.7.5

Chroots

- (See Carl's talk from earlier today)
- Chroots provide a higher level of isolation.
  - Sandbox for one job isn't visible by others.
- Useful for removing all setuid binaries: a common vector of attack.
  - Remove stuff from the user environment you don't want jobs to play with.

In RHEL6

Primer: Cgroups

- Control groups, or "cgroups", are a mechanism for managing a set of processes. Unlike POSIX process groups, you must* be root to create or move to a different group.
- The kernel does the process tracking for us.
    - Within the kernel, there exists various *controllers* that operate on cgroups. There are a wide variety: they control resource limits, fairshare policies, and accounting.

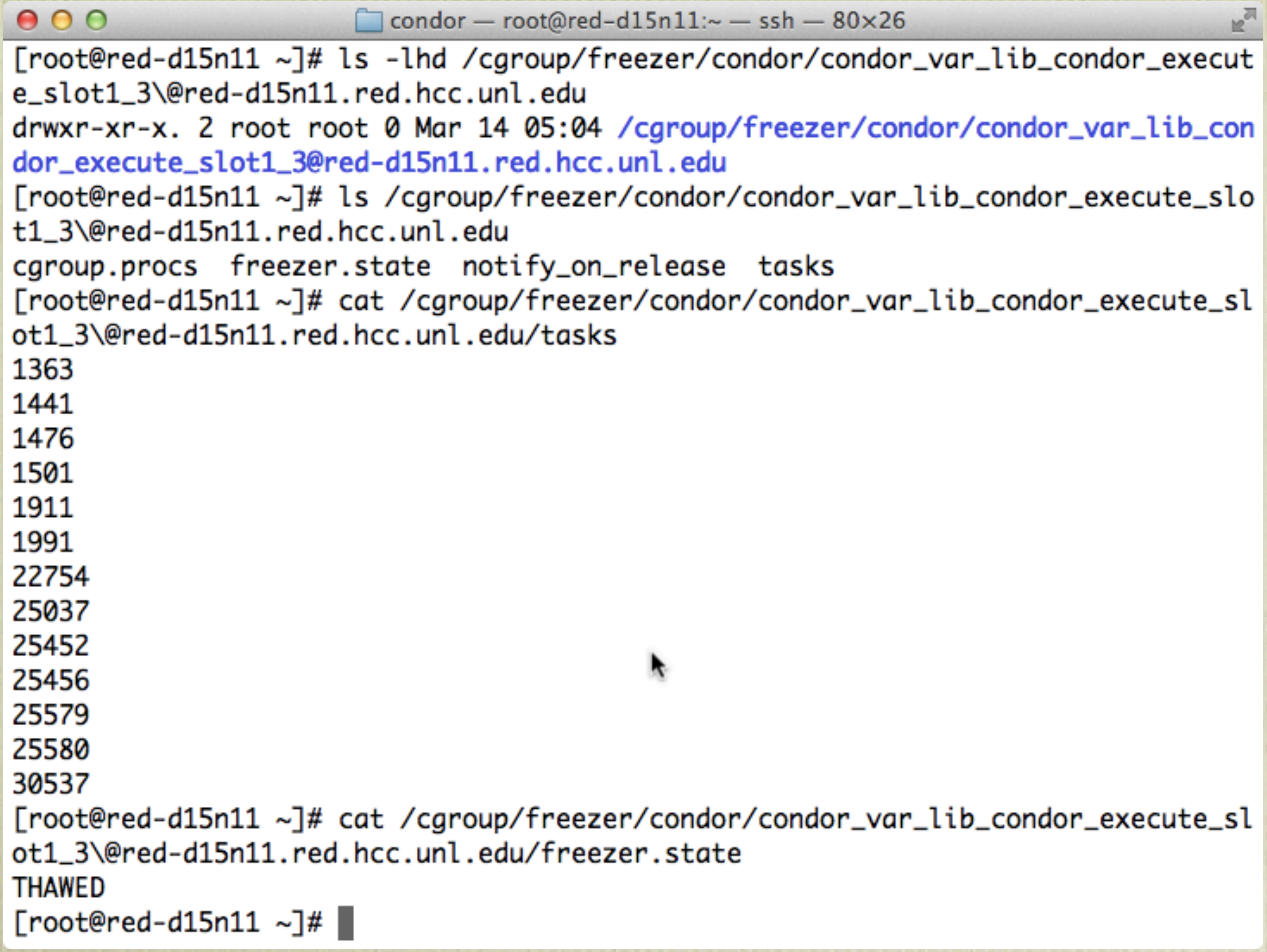
*Or have a sysadmin give your user the power.

In Condor 7.7.0

Killing Jobs

- An example is the “freezer” controller.
    - The freezer can take a cgroup of processes and prevent them from receiving further CPU cycles.
    - Similar to SIGSTOP - but delivered to multiple processes atomically.
- Condor will spawn the job into a dedicated cgroup.
    - The kernel guarantees all child processes will stay in this cgroup.
    - To kill a job, it will freeze the cgroup, send SIGKILL to all processes, then unfreeze.
    - All processes will receive SIGKILL simultaneously: no more race conditions with forking!

Cgroups are managed via the Filesystem



condor — root@red-d15n11:~ — ssh — 80×26

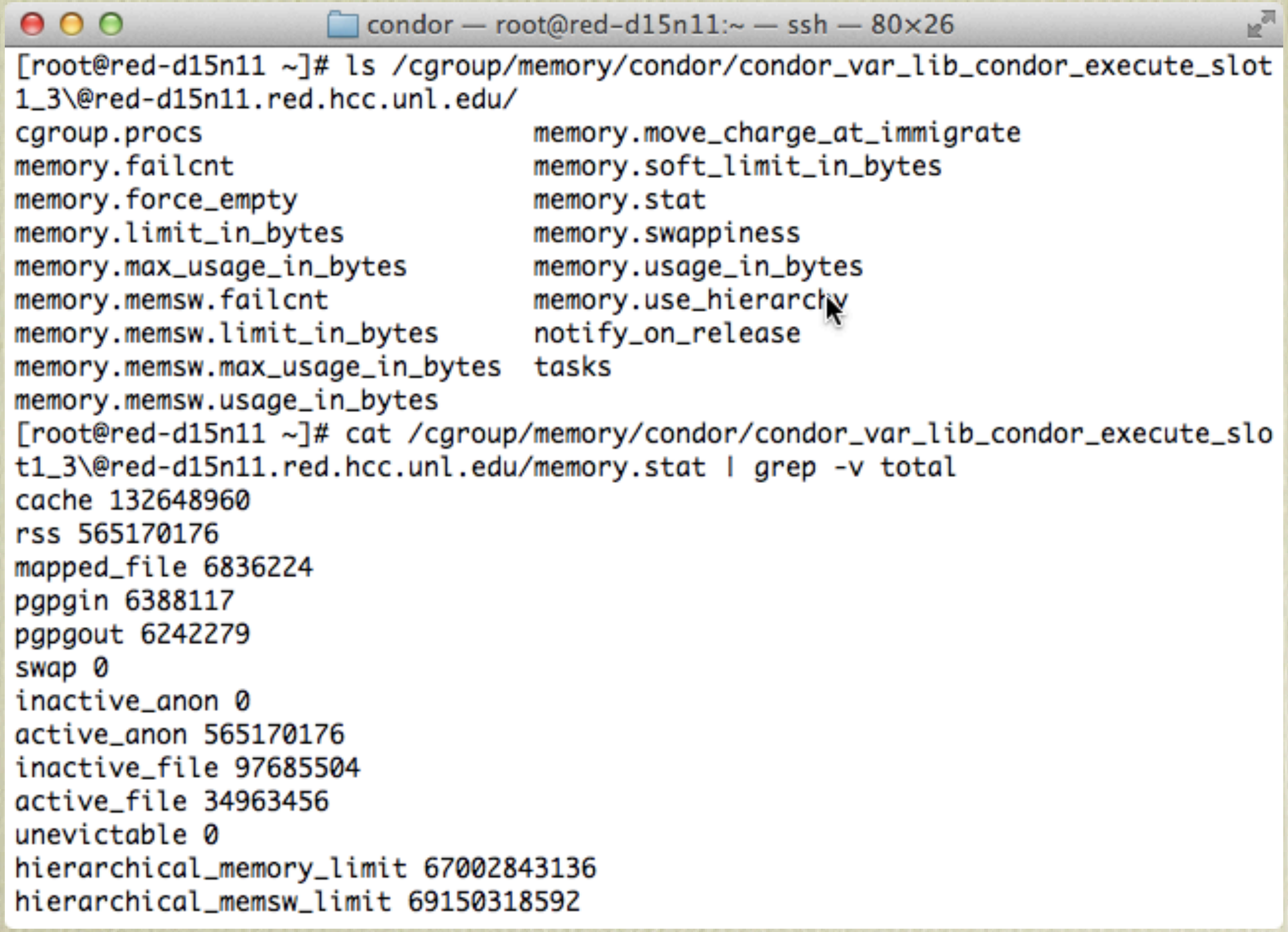
```
[root@red-d15n11 ~]# ls -lhd /cgroup/freezer/condor/condor_var_lib_condor_execut
e_slot1_3\@red-d15n11.red.hcc.unl.edu
drwxr-xr-x. 2 root root 0 Mar 14 05:04 /cgroup/freezer/condor/condor_var_lib_con
dor_execute_slot1_3@red-d15n11.red.hcc.unl.edu
[root@red-d15n11 ~]# ls /cgroup/freezer/condor/condor_var_lib_condor_execute_slo
t1_3\@red-d15n11.red.hcc.unl.edu
cgroup.procs  freezer.state  notify_on_release  tasks
[root@red-d15n11 ~]# cat /cgroup/freezer/condor/condor_var_lib_condor_execute_sl
ot1_3\@red-d15n11.red.hcc.unl.edu/tasks
1363
1441
1476
1501
1911
1991
22754
25037
25452
25456
25579
25580
30537
[root@red-d15n11 ~]# cat /cgroup/freezer/condor/condor_var_lib_condor_execute_sl
ot1_3\@red-d15n11.red.hcc.unl.edu/freezer.state
THAWED
[root@red-d15n11 ~]# 
```

Condor 7.7.0

Memory Accounting

- The memory controller is already used for memory accounting: much more accurate with modern jobs.
- When memory limits are hit, our options are either to ignore it (and let them be violated) or to kill the job.
    - Not very efficient if the job only needs to violate the limit slightly.

Memory Accounting


condor — root@red-d15n11:~ — ssh — 80×26

```
[root@red-d15n11 ~]# ls /cgroup/memory/condor/condor_var_lib_condor_execute_slot
1_3\@red-d15n11.red.hcc.unl.edu/
cgroup.procs                      memory.move_charge_at_immigrate
memory.failcnt                    memory.soft_limit_in_bytes
memory.force_empty                memory.stat
memory.limit_in_bytes             memory.swappiness
memory.max_usage_in_bytes         memory.usage_in_bytes
memory.memsw.failcnt              memory.use_hierarchy
memory.memsw.limit_in_bytes       notify_on_release
memory.memsw.max_usage_in_bytes   tasks
memory.memsw.usage_in_bytes
[root@red-d15n11 ~]# cat /cgroup/memory/condor/condor_var_lib_condor_execute_slo
t1_3\@red-d15n11.red.hcc.unl.edu/memory.stat | grep -v total
cache 132648960
rss 565170176
mapped_file 6836224
pgpgin 6388117
pgpgout 6242279
swap 0
inactive_anon 0
active_anon 565170176
inactive_file 97685504
active_file 34963456
unevictable 0
hierarchical_memory_limit 67002843136
hierarchical_memsw_limit 69150318592
```

Condor 7.7.6?

Memory Limits

- We have implemented two new policies:
  - *Soft limit:* A job can use arbitrary amounts of memory until the node starts swapping; then, only jobs over the limit will be swapped out.
  - *Hard limit:* When a job hits its memory request, it will immediately start swapping.

Condor 7.7.6?

CPU Fair-share

- Condor has had CPU affinity for quite some time.
    - Prevents idle CPUs from being used; number of jobs really must be number of cores.
- The "cpu" controller allows fine-grained fairsharing to be done between the cgroups without having to lock them to a CPU.
    - So, you can evenly fairshare 24 cores between 25 jobs.

Other Accounting Tricks

In Condor 7.7.0

- Block device I/O: Actually, not nearly as interesting as I thought it would be.
  - Maybe something for the future? At a CMS site with up to 24 cores/node, not very relevant.

In github

- Network I/O: Quite an invasive patch set, but we create a virtual ethernet device per job.
  - This allows us to have per-job firewall rules and to-the-byte network accounting.

Other cgroup tricks

- We limit all of Condor and job processes to 62GB RAM (on a 64GB host) to prevent Condor from crashing the host.
- Planning to do the same thing for the OSG CE.
- Prioritize Hadoop block I/O over scratch disk.

For the Future

- In Condor, I really want per-job or per-node swap rates and NFS activity stats.
- We're actively porting a subset of this work to PBS. MOUNT_UNDER_SCRATCH and memory limits should be available as a patch to torque.
- In SL6, Cgroups are a real game-changer for resource management without having to resort to virtual machines!