Effective use of cgroups with HTCondor

Tom Downes

Center for Gravitation, Cosmology and Astrophysics
University of Wisconsin-Milwaukee
LIGO Scientific Collaboration

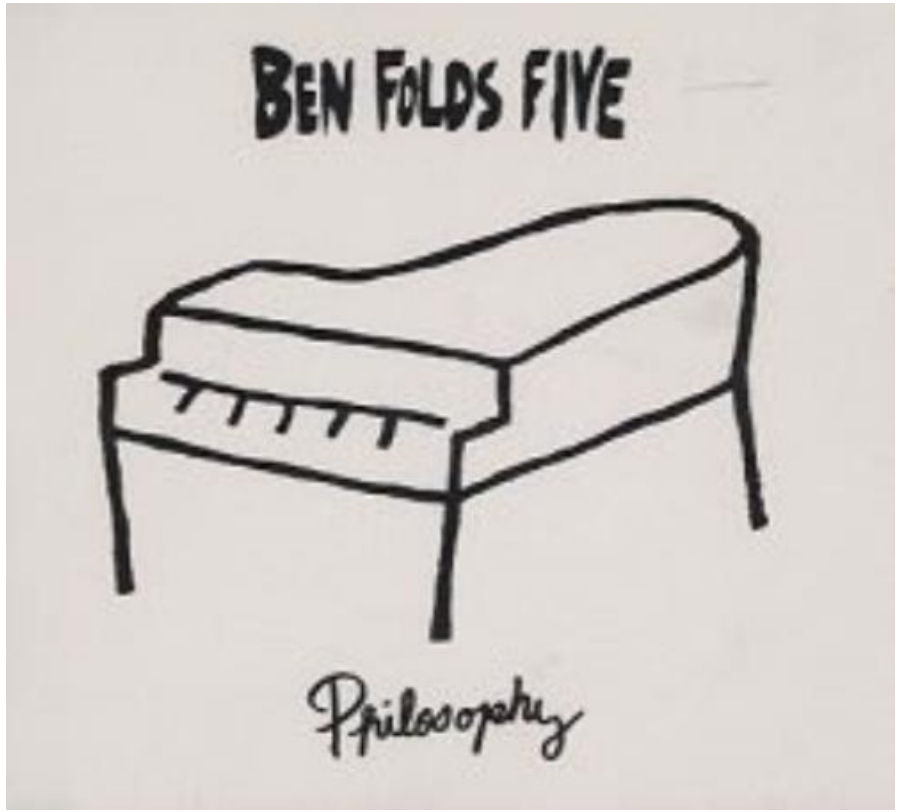
HTCondor Week 2017

What are Control Groups (cgroups)

- Condor is a fault tolerant system for running jobs
- Control Groups provide fault tolerance for systems by managing access to resources like CPU / memory / devices / network
- systemd tightly coupled: RHEL 7+, Debian 8+, Ubuntu 15.04+
- Broadly coupled to movement to isolate processes from one another
- Vastly improve measurements of job resource utilization

I got my philosophy

- I “speak for the systems” not jobs
- Services working should be normal
- Need system autonomy to live life
  - Go on vacation
  - Work on important things



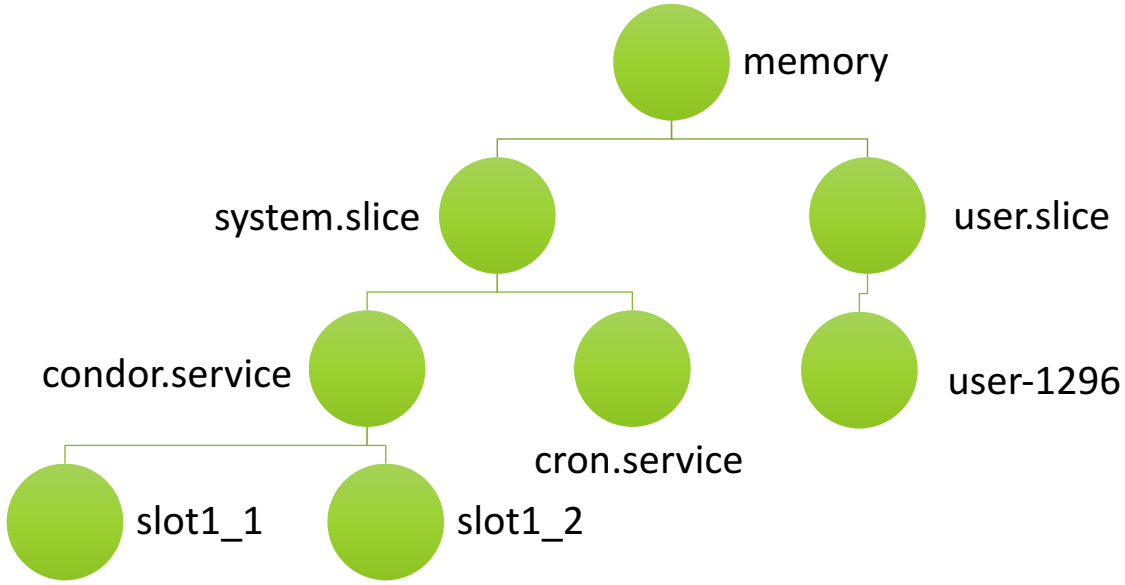

```
2000345.0   scaudill         4/18 23:58 Error from
slot1_3@execute1068.nemo.uwm.edu: Job has gone over memory limit of 5120
megabytes. Peak usage: 5320 megabytes.
```

The punchline

```
$ cat /etc/condor/config.d/cgroups
BASE_CGROUP=/system.slice/condor.service
CGROUP_MEMORY_LIMIT_POLICY=soft
```

How cgroups work

- Controllers manage a single resource (CPU, memory, etc.) hierarchically
- Each controller is K-ary tree structure exposed in directory structure
- Processes are assigned to nodes
- Condor daemons in systemd cgroup
- Condor adds leaf nodes for jobs!




```
$ cat /sys/fs/cgroup/memory/system.slice/condor.service/tasks
1640
...
```

Memory controller

- Measures RAM usage (Resident Set Size)
- Separately measures **combined** RAM and swap usage
- Actual swap usage determined by `swappiness`, a cgroup setting!
- RHEL7 docs misleadingly suggest that you can **prevent** swap usage
  - the only unused swap is no swap!
- Hierarchical accounting: descendant cgroups count toward ancestors

Debian / Ubuntu

- By default, Debian does not enable memory controller
- Neither Debian/Ubuntu enable swap features within controller

```
$ grep GRUB_CMDLINE_LINUX_DEFAULT /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="quiet cgroup_enable=memory swapaccount=1”
$ update-grub
$ shutdown –r now
```

- Preseed at install (avoid first boot problems!) with

```
grub-pc grub2/linux_cmdline_default string quiet cgroup_enable=memory swapaccount=1
```

Limiting Condor execute nodes

- In addition to job limits, let's limit total memory used by Condor daemons and processes
  - Use ExecStartPost to limit RAM+swap (thanks, RHEL7 docs!)

```
$ cat /etc/systemd/system/condor.service.d/memory.conf
[Service]
MemoryAccounting=true
MemoryLimit=4G
# this value must be greater than or equal to MemoryLimit
ExecStartPost=/bin/bash -c "echo 4G >
/sys/fs/cgroup/memory/system.slice/condor.service/memory.memsw.limit_in_bytes"
```

Limiting users on a submit node

- Add same condor.service limits as on execute node
- Configuration below limits **total RAM by all users** at command line
- For users who login via ssh, sets per-process virtual memory limit

```
$ cat /etc/systemd/system/user.slice.d/50-MemoryLimit.conf
[Slice]
MemoryAccounting=true
MemoryLimit=6G


$ cat /etc/systemd/system/openssh-server.service.d/memory.conf
[Service]
LimitAS=2147483648
```

Details of enforcement (hard limit on Condor)

When **hard limit** on whole condor service is reached…

1. Kernel attempts to reclaim memory from cgroup + descendants
   - swap out / delete file cache until below hard limit or soft limit (if enabled)
2. If that fails, OOM killer invoked on cgroup + descendants
3. OOM killer targets jobs with high `/proc/[pid]/oom_score`
   - “bad” jobs are ones that are closest to their limit
   - jobs have `oom_score_adj` set to appear, at minimum, at 80%+ of their limit

Details of enforcement (soft limits on jobs)

- Soft limits on jobs == ”OOM event will occur above the slot cgroup”
- With hard limit on Condor, see prior slide
- Otherwise, it will occur when all system RAM and swap are exhausted
- In both cases, Condor intercepts OOM to perform kill and cleanup

Differences in behavior

- Hitting Condor service hard limit is different from exhausting system
- If system resources exhausted, global OOM “outside” of cgroups
  - In this case, the `oom_score_adj` set by Condor is very important!
- If triggered by Condor hard limit, *every job sees OOM event!*
  - In this case, the `oom_score_adj` set by Condor doesn’t matter because all jobs have same value (-800)
  - Condor < 8.6: responded by killing every job on node!
  - Condor >= 8.6: with default value of `IGNORE_LEAF_OOM=True`, examines job before killing. Don’t kill if <90% of its request.

cgroups-v2

- Controllers unified into much simpler tree structure
- Processes live **only** in leaf nodes
- Memory controller eschews soft/hard limits in favor of low/high/max
  - Built-in understanding that job memory usage is hard to predict: be flexible
  - Eliminates userspace OOM handling
  - Encourages applications to monitor memory and change limit dynamically
- Memory controller measures swap as its own resource
- Possible to use memory v2 interface while using v1 for others
- cgroups author: https://www.youtube.com/watch?v=PzpG40WiEfM

Thoughts

- Define system stability as the goal rather than constraining jobs
- Memory management
  - Soft limits don’t really do much except encourage a bit of swapping
  - Soft and hard limits are separate settings. Could set both for jobs!
  - There **must** be a swappiness knob
  - Re-consider IGNORE_LEAF_OOM behavior for jobs at 90% level
  - Consider alternative approach in spirit of cgroups-v2
  - cgroups-v2 will be production opt-in on Debian 9 in next few months
- Swap existence is a matter of religion
  - Might be able to use systemd/cgroups to really make backfill work nicely

Increase use of systemd?

- Condor could mimic systemd by creating many scope units
- In prepping this talk, I concluded that one still needed to write directly to cgroups API, but might examine long-term benefits of using systemd as stable interface to cgroups
- If able to be made compatible with `/etc` configuration files or templates, allows user to use cgroups for other resources without developing new Condor knobs