Opportunities and Challenges Courtesy Linux Cgroups Version 2

“Putting jobs in a box”

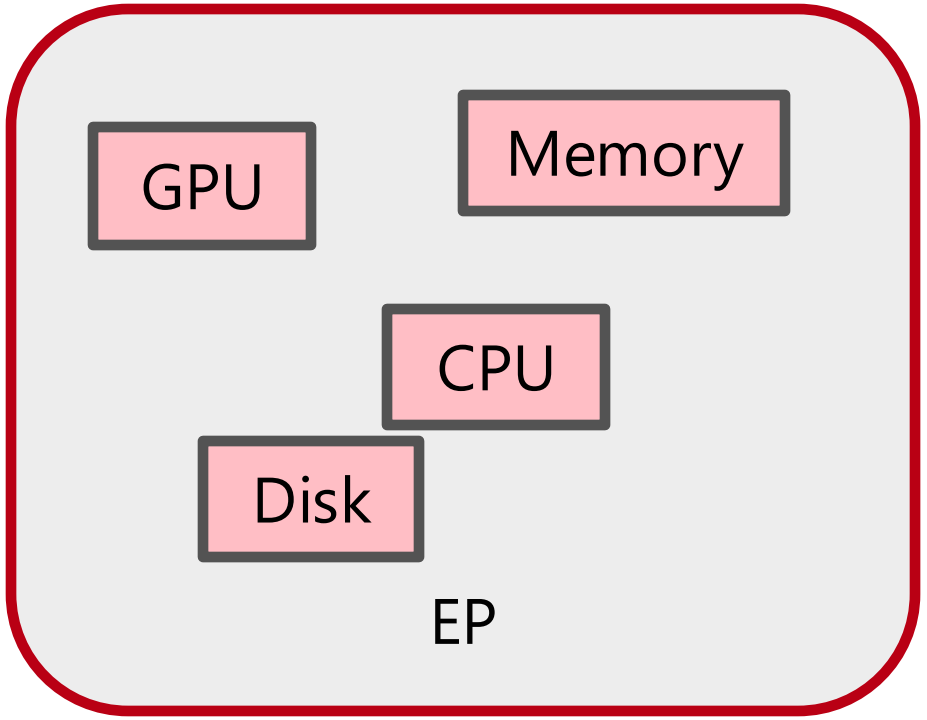
Brian Bockelman, 26 September 2024

MORGRIDGE
INSTITUTE FOR RESEARCH

Long Search for managing EP resources

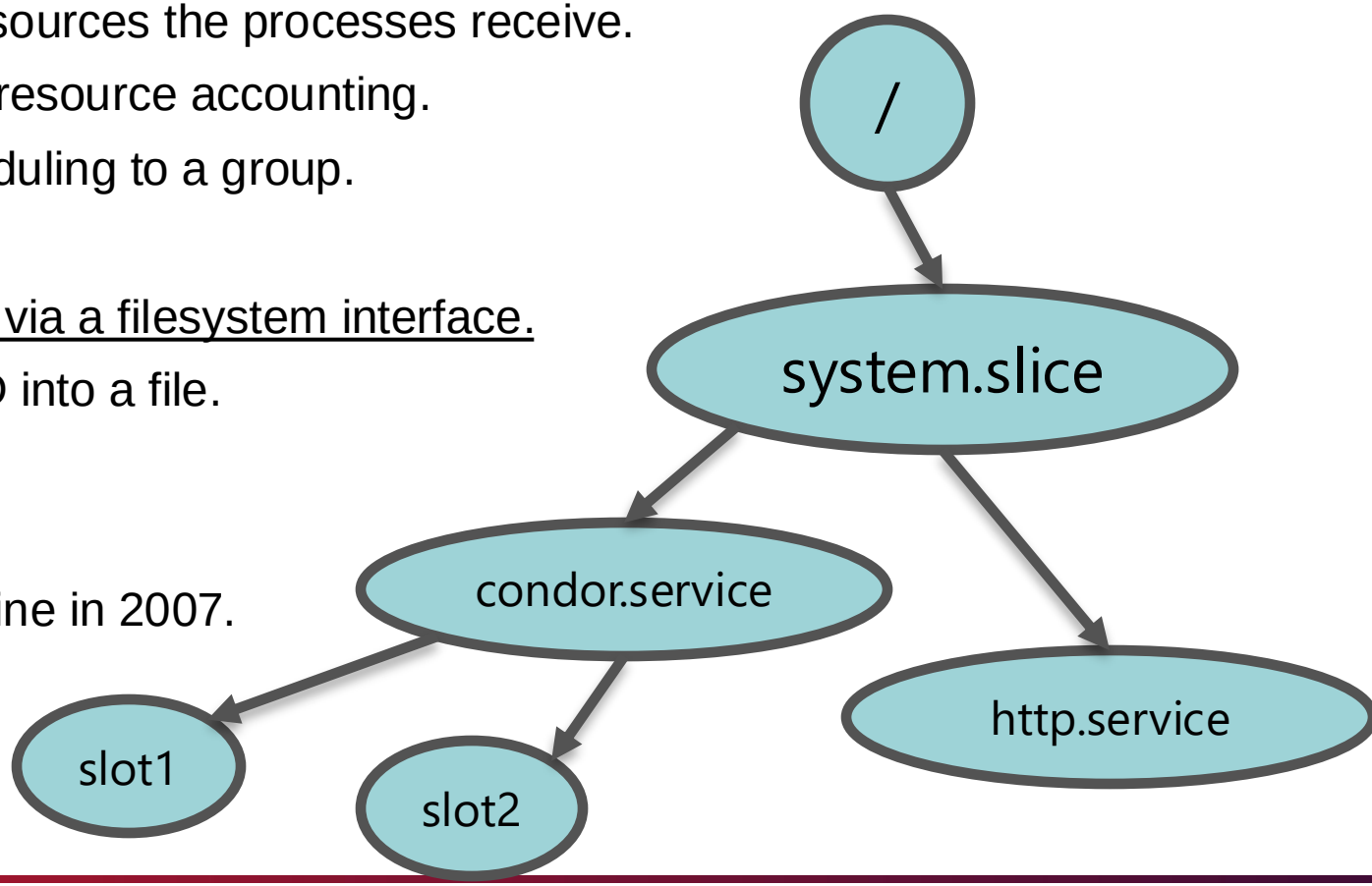
- HTCSS has long searched for an ideal “box” to place jobs in:
  - Manage finite resources (CPU, GPU, memory, disk, network).
  - Recursive.
  - Unprivileged.
  - Unable to “break out”
- What do we mean by **manage**?
  - Accurate account for & report usage.
  - Prevent job from overrunning assigned resources
  - Execute policy based on resource consumption.
- How to do this in Linux?
  - Traditional Unix process management assumes well-behaved, cooperative processes. If we make this assumption, then there’s no problem to solve!
  - HTCSS accumulated about 10 years of “tricks”, mostly involving rooting around in /proc, secondary GIDs, environment variables. Whole daemon for process tracking (condor_procd)!

Not close to our ideal box!



Linux to the rescue!

- The Linux kernel came up with the idea of “control groups”: a **set of processes** whose properties or resources were managed by a kernel **controller**.
  - The **CPU controller** manages the scheduler resources the processes receive.
  - The **memory controller** manages the memory resource accounting.
  - The **freezer controller** will deny any CPU scheduling to a group.
- This is Unix, I know this!
  - Kernel decided to expose and manage cgroups via a filesystem interface.
  - Want to add a thread to a cgroup? Write its PID into a file.
  - Want to create a child group? `mkdir`
  - Want to read a counter? `cat` the right file!
- Efforts began in 2006 and merged into Linux mainline in 2007.



Ok, so we didn't do a good job with git in 2011

commit 321489d95346fbaa3c0a2c9daf15df74aac9664c
Author: Brian Bockelman <bbockelm@cse.unl.edu>
Date: Thu May 5 10:13:46 2011 -0500

===GT=== #1831

This is the final patch from Brian Bockelman which implements cgroups on linux for the condor_procd. In building the patch, I've found some build errors which I'll fix in subsequent patches.

===VersionHistory===

When Condor can utilize the CGroup functionality in Linux, the accuracy of process tracking, accounting, and management can be made much more accurate.

4 years from mainline commit to HTCondor using it in production

Some wins, some losses

- How good are cgroups for our ideal box?
  - Management: ✅ … mostly
  - Unable to break out: ✅
  - Recursive: 😒
  - Unprivileged: ❌
- Pitfalls:
  - Doesn't address disk or I/O management. See separate talk by Cole.
  - Impractical to use recursively (as a nested job is unprivileged).
  - Never implemented GPU management.

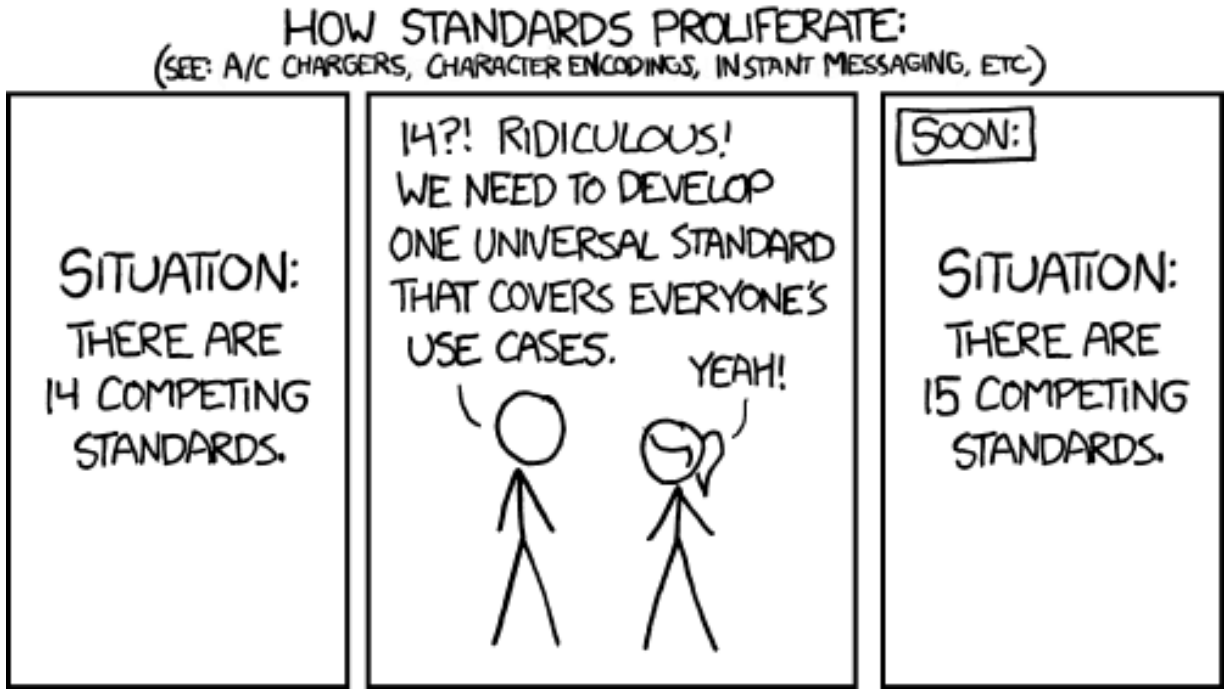
After 13 years, mostly "it just works"

Meanwhile, in Linux…

- Cgroups V1 was a great donation from Google and IBM to the Linux kernel.
- However, it was a first attempt and not everything was successful:
  - **Overly complex**: Each thread can be in a separate control group.
  - **Overly complex**: Each thread can be in a separate control group per controller.
  - **Overly complex**: Threads are allowed to be in “internal nodes” in the cgroup hierarchy; assigning resources to sibling is complicated & difficult to understand.
  - **Cruft**: Inconsistent naming scheme within the mounted filesystem.
  - **Unpredictable**: multiple writers to the same cgroup can cause unpredictable results.

Cgroups to the rescue … again?

- Cgroups V2 was added to the mainline kernel in 2014, a decade ago.
  - HTCSS added support this year – why a decade?
- Upgrading is harder than creating:
  - **Need to keep V1 and V2 working side-by-side**.
    - Will be like this for several years.
  - Dropped libcgroup (dropped from RHEL9), switched to interacting directly via the filesystem.
  - Moved cgroup code from separate procd into the HTCondor daemons.
  - Making behavior consistent between V1 and V2.
    - Esp. difficult in interpreting memory usage.
  - Plus typical bug smashing!



Over the hump: new features! GPUs

- The picture of multi-tenancy and GPUs is … ugly. Users can trivially stomp on each other's running jobs

Any job can read OR WRITE to ANY gpu on the system!

Cause of crashes, hangs, incorrect data in CHTC, elsewhere!

```
$ nvidia-smi

+-----------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.14              Driver Version: 550.54.14      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla P100-PCIE-16GB           On  |   00000000:3B:00.0 Off |                    0 |
| N/A   37C    P0             33W /  250W |    1030MiB /  16384MiB |      1%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  Tesla P100-PCIE-16GB           On  |   00000000:D8:00.0 Off |                    0 |
| N/A   33C    P0             31W /  250W |    4496MiB /  16384MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

GPUs and HTCondor

- The first thing HTCondor did was set the CUDA_VISIBLE_DEVICES environment variable, which hides a GPU from the CUDA libraries.
  - This worked until the first user copy/pasted `CUDA_VISIBLE_DEVICES=0`, thanks to Stack Overflow.

```
$ export CUDA_VISIBLE_DEVICES=0
$ nvidia-smi
Fri Jul  5 12:21:43 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.14              Driver Version: 550.54.14      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla P100-PCIE-16GB           On  |   00000000:3B:00.0 Off |                    0 |
| N/A   37C    P0             33W /  250W |    1030MiB /  16384MiB |      1%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

GPUs, HTCondor, and Cgroups

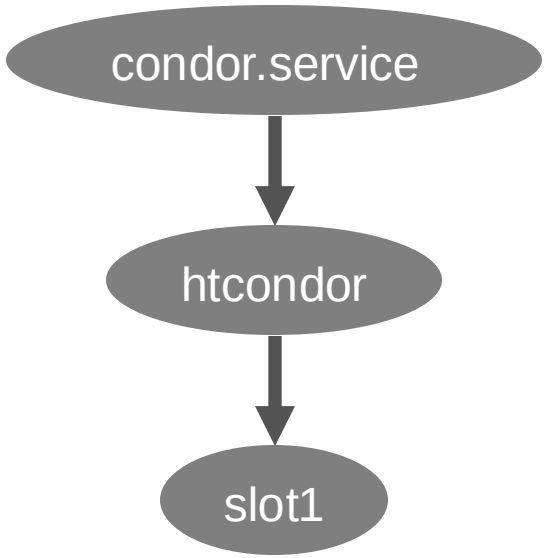
- Cgroups V2 has a new controller, the device controller, that allows HTCondor to install a small program that is executed by the kernel when devices files are opened.
  - HTCondor installs a program that returns “permission denied” if a job ever tries to open a GPU device that it wasn’t assigned.

Finally, isolation for GPUs.

```
$ nvidia-smi

+-----------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.14              Driver Version: 550.54.14      CUDA Version: 12.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE-16GB           On  |   00000000:3B:00.0 Off |                    |
| N/A   37C    P0              33W /  250W |    1030MiB /  16384MiB |      1%      |
|                                         |                        |                    |
+-------------------------------+----------------------+----------------------+
```

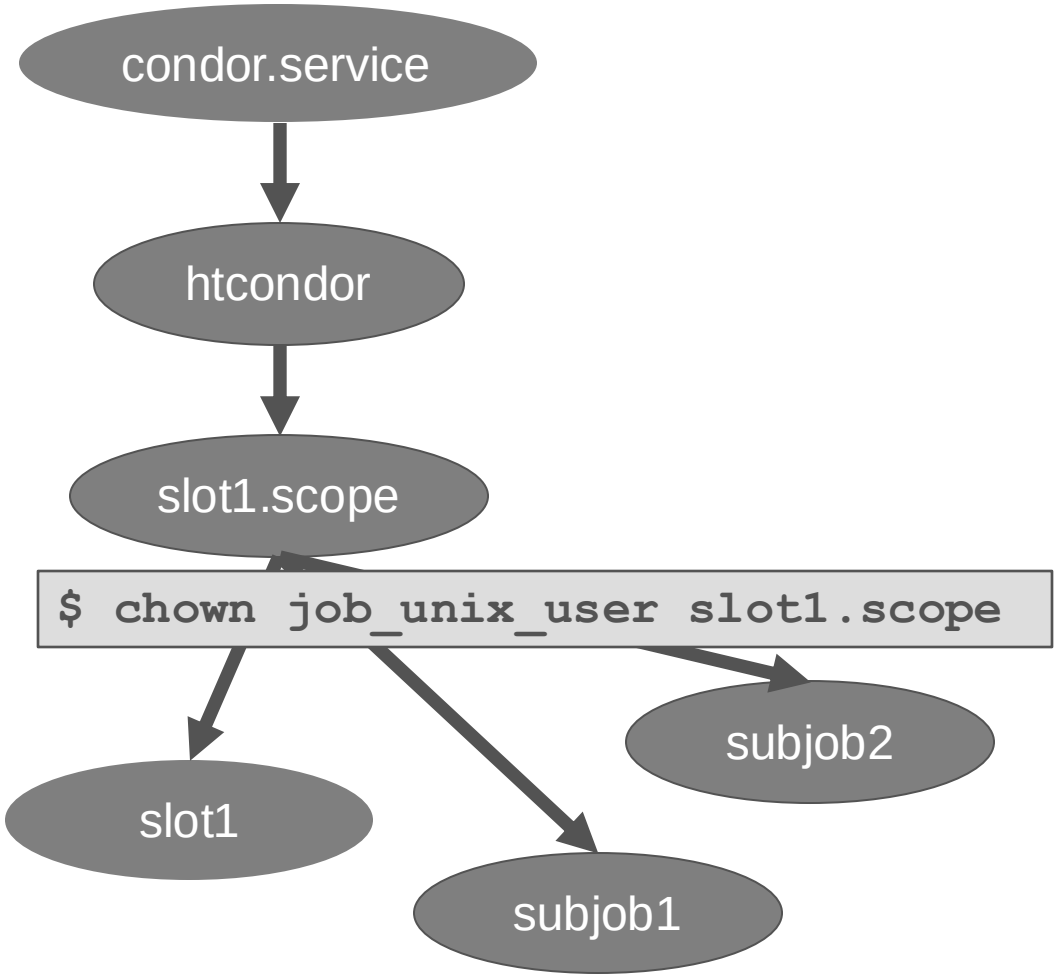
New features: Delegation




```
$ chown job_unix_user slot1
```

- Remember our properties of an “ideal box”? One missing piece was “recursive”!
  - In cgroups V2, it’s more practical to delegate (chown!) to the job user.
- Hiccup: in Cgroups V2, tasks can only live in hierarchy “leaf” nodes.
  - Can’t simply chown the slot cgroup.

Delegation – small tweak



- Instead, we create a deeper hierarchy and chown the parent to the job's unix user.
- Suddenly, the glidein can manage the cgroup hierarchy!
- We can start considering the matrix of local scheduler & overlays, pushing delegation in as many cases as possible.

Local Batch	HTCondor	SLURM	K8S	HTCondor
Overlay	HTCondor	HTCondor	HTCondor	Alice Pilots

Help us advocate to enabling delegation on SLURM and K8S!

ALICE is doing this today !

with pilots over HTCondor (and SLURM)

Whither systemd?

- The filesystem is a fine API for interacting with the kernel cgroups.
- However, systemd *also* has an API for managing cgroups.
  - Currently, we tell systemd to delegate a cgroup to HTCondor - *and be hands off*.
  - This means the various systemd tools do *not* have visibility into the HTCondor jobs.
- If we used systemd to launch create processes, we:
  - Would benefit from systemd's cgroup handling knowledge.
  - Administrators would have visibility into what's happening using the systemd toolset.
  - **Would open the door to restarting (upgrading?) the EP *without* losing running jobs**.

What do you think?

Final thoughts

- For over a decade, cgroups have been essential to “building a better box” and managing jobs.
- We’re in the middle of a transition from V1 to V2
  - Hopefully squashing the last few bugs! TBD…
- Cgroups V2 opens up some new doors:
  - GPUs are finally managed! Combined with the LVM work … do we have all the resources managed?
  - Can finally delegate and use cgroups from unprivileged EPs (glideins)!

Help us spread use of unprivileged cgroups!

Questions?

This project is supported by the National Science Foundation under Cooperative Agreements OAC-2030508. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

FEARLESS SCIENCE

MORGRIDGE INSTITUTE FOR RESEARCH