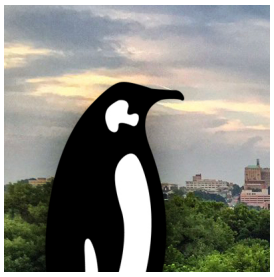

Contribution ID: **278** Type: **not specified**

Cgroups and Enterprise Users

Monday, 13 November 2023 17:00 (30 minutes)

Enterprise distributions are finally transitioning to cgroup v2 as the default [1][2]. But as has been discussed in previous Linux Plumbers Conferences [3][4], the transition from cgroup v1 to cgroup v2 has not been seamless for userspace applications.

Some (simpler) enterprise applications have been able to utilize Systemd service files to manage their cgroups needs, but larger and more complex programs require more granular control. The Oracle database has started to utilize libcgroup's cgroup v1/v2 abstraction layer, and this has solved some of their easier cgroup challenges.

Topics we're interested in discussing. (We're definitely open to other areas of discussion - let us know if you have areas that pique your interest, and we'll gladly consider them. But this is what is of interest within Oracle for the next year or so.)

- Processes in the root cgroup v1 cgroup are not subject to cgroup resource restrictions (like cpusets), but cgroup v2 does enforce this. Some applications have chosen to use cgrulesengd to solve this, but this risks running afoul of the single-writer rule. And honestly it feels antiquated. How can we provide a more effective solution to these customers whose products will run on either a v1 or v2 system? (To further complicate things, these products also run on many different kernel versions and systemd versions and not all of the latest and greatest features are available.)
- Some products have relied heavily on realtime for their high-priority (and/or low-latency) processes. Older versions of Oracle Linux allowed them to set realtime quotas and periods on a per-cgroup basis (via the CONFIG_RT_GROUP_SCHED kernel config), but this feature isn't available on cgroup v2 systems. (And I think this is the right choice.) With EEVDF coming soon, I would be curious to hear what are now the best practices for prioritizing certain processes - nice, EEVDF slice length, realtime scheduler class, etc.
- Now that libcgroup plays nicely with systemd (see release v3.1.0 [5]), we're encouraging distros to again provide it as a package. Yes, users can use libcgroup to violate the single-writer rule, but used properly it can greatly simplify cgroup management, including the creation of delegated systemd scopes. And of course all of its standard cgroup reading/writing capabilities remain. I wouldn't mind giving a quick rundown of where libcgroup is and where it's going. It has proven invaluable for those applications that need to straddle both cgroup v1 and v2.
- We would like to hear the community's thoughts on settings that are unmappable from cgroup v1 to cgroup v2, like cpuset.sched_relax_domain_level. Currently libcgroup will raise an UNMAPPABLE error in cases like this; note the user can silence this error. Is this the best we can do?
- Is there interest in adding a libcgroup abstraction/mapping of cpu.stat and other multiline cgroup files?

[1] https://docs.oracle.com/en/operating-systems/oracle-linux/9/relnotes9.0/ol9-features-changes.html
[2] https://www.redhat.com/en/blog/whats-new-rhel-90-beta
[3] https://lpc.events/event/4/contributions/524/
[4] https://lpc.events/event/11/contributions/930/
[5] https://github.com/libcgroup/libcgroup/releases/tag/v3.1.0

Primary authors: BABULAL, Kamalesh; HROMATKA, Tom

Presenters: BABULAL, Kamalesh; HROMATKA, Tom

Session Classification: Containers and checkpoint/restore MC

Track Classification: LPC Microconference: Containers and checkpoint/restore MC