Contribution ID: **135** Type: **not specified**

Cgroup v1/v2 Abstraction Layer

Tuesday 10 September 2019 19:10 (20 minutes)

Abstract

We have cgroup v1 users who want to switch to cgroup v2, but there currently isn't an upstream migration story for them. (Previous LPC talks have focused on the issues of migrating from v1 to v2, but no substantial upstream solution has come to fruition.)

The goal of this talk is to discuss the cgroup v1 to v2 migration path and gauge community interest in a cgroup v1/v2 abstraction layer.

Problem Statement

Several Oracle products have very, very long product lifetimes and are designed to run on a wide range of Linux kernels and systemd versions. These products are encountering difficulties as cgroups continues to grow and change. Older kernels only support v1, but v2 is the future in newer kernels with v1 effectively in maintenance mode. Newer versions of systemd have started to abstract the cgroup interface, but upgrading older systems to newer versions of systemd is often not feasible. Ultimately, long-lifespan products are spending an increasing and inordinate amount of time and effort managing their cgroup interfaces.

There is interest within Oracle to create a cgroup abstraction layer that will allow long-lived products to utilize the most advanced cgroups features available on every supported system. Ideally these products will be able to rely upon a library to abstract away the low-level cgroup implementation details on that system.

Audience

Anyone interested in cgroups

Why Should the Audience Attend and/or Care

- We would like to develop a cgroups abstraction layer in the next year or so. We would love to collaborate with others to build and design a solution that can help the entire community
- Do other people/companies have interest in an abstraction layer? We want to hear other use cases and needs to better serve as many people as possible
- Is there already something out there that we can utilize and build on?
- Given the wide array of users and use cases, the library will likely need to have bindings for today's most popular languages - python, go java, etc.

- There are a multitude of API possibilities. What level(s) of abstraction are of interest to the community? e.g.
  GiveMeCpus(cgname=foo, cpu_count=2, exclusive=True, numa_aligned=True, ...)
  CgroupCreate(cgname=foo, secure_from_sidechannel=True, ...)

I agree to abide by the anti-harassment policy

Yes

I confirm that I am already registered for LPC 2019

Primary author: HROMATKA, Tom

Presenter: HROMATKA, Tom

Session Classification: Containers and Checkpoint/Restore MC