Lab BSY PROC-2

Introduction & Prerequisites

This laboratory is to learn how to:

- Find and understand the subsystems managed by the kernel
- Identify cgroups hierarchies created by systemd
- Create cgroups and associate processes to them

The following resources and tools are required for this laboratory session:

- A ZHAW VPN session
- Any modern web browser
- Any modern SSH client application
- OpenStack Horizon dashboard: https://ned.cloudlab.zhaw.ch
- OpenStack account details
  - See Moodle
- Username to login with SSH into VMs in ned.cloudlab.zhaw.ch OpenStack cloud from your laptops
  - **`Ubuntu`**

Time

The entire session will take 90 minutes.

Assessment

No assessment foreseen

Task 1 – Understand Cgroups Version One

Subtask 1.1 – Default System Configuration

Analyze the default cgroup configuration of Ubuntu. Which **subsystems** are supported by the Ubuntu kernel? Explain the output.

Which **hierarchies** are provided by default? Which subsystems are configured at which hierarchy?

`Navigate to the default CPU hierarchy and check how many cgroups are present. What may be the purpose of these cgroups? Who created them? You may find the command systemd-cgls useful.`

`How many processes are in cgroup user.slice/system.slice/init.slice?`

Run the following commands and explain the output: ps xawf -eo pid,user,cgroup,args

Check the configuration for a process called “cron” using “ps” and “grep”. Explain both, the configuration for the “cron” and “ps” process, in terms of systemd and cgroup configuration.

Verify your understanding by using the following commands: systemd-cgtop and systemd-cgls

Can you identify the “cron” service?

Subtask 1.2 – Default System Configuration by Systemd

Obviously, systemd creates a number of cgroups. Identify the respective unit files and explain their configuration.

Modify the user.slice configuration such that all processes created by users do not receive more than 20% of CPU share. Verify your configuration by creating more load than your system can handle, e.g. by creating a number of “wc /dev/zero &” background processes (& send the process directly into the background). You may find man `man systemd.resource-control useful. What happened to the “cron” process?`

Free one wc process from the limits imposed by the user.slice related cgroup.

Task 2 – Create a Custom Control

Create an environment to control CPU access (CPU affinity) from scratch, i.e. not using any preconfigured cgroups. Create a tmpfs under /mnt and use this directory for your cgroup hierarchy.

How many CPUs can be controlled on your system per default?

Create M processes, with M being the number of CPUs (N) on your system plus one (M=N+1). Those processes shall consume 100% CPU load. Now configure your cgroups to only allow these processes to use half of the CPUs available on your system. Mind the following prerequisite (command to be executed once the controller is mounted and the cgroup created): **`echo 0 > cpuset.mems`**

Verify and explain the effect by looking at the processes via "top". Also double-check the CPU assignments (affinity) with the command "taskset". Disable those CPUs, via "chcpu", that you allowed in your cpuset and explain what happens with those processes pinned onto them.

Cleanup

IMPORTANT: At the end of the lab session:

- **Delete** all -unused - OpenStack VMs, volumes, security group rules that were created by your team.