Cgroups and Enterprise Users
Linux Plumbers 2023

Tom Hromatka, Kamalesh Babulal
tom.hromatka@oracle.com, kamalesh.babulal@oracle.com
Linux Kernel Development
Oracle Corporation
Nov 13th, 2023
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.
**Kernel cgroup**

**Upstream releases**

- cgroup v1
  - v2.6.24
  - Jan 2008
  - memory, I/O, pids

- cgroup v2
  - v3.16
  - Aug 2014
  - (experimental*)
  - RDMA
  - v4.11
  - Apr 2017
  - cpu, device

- cgroup v2
  - v4.5
  - Mar 2016
  - (stable)
  - PSI
  - v4.15
  - Jan 2018
  - cpuset v2
  - v4.20
  - Dec 2018
  - freezer
  - v5.0
  - Mar 2019
  - hugeTLB
  - v5.2
  - Jul 2019
  - misc
  - v5.6
  - Mar 2020
  - v5.13
  - Jul 2021

*mount -t cgroup -o __DEVEL__sane_behavior cgroup <mount-point>*
Enterprises cgroup
Upstream vs Distribution Adoption

Enterprise Distributions
- LTS of 10 years or more

Enterprise Customers
- Applications runs on multiple distros/release
- Long term Application support
- Need to support cgroup v1/v2
Libcgroup Releases
Upstream vs Distribution adoption vs libcgroup release
Why/When to Use libcgroup

- Programmatic management of delegated scopes
- CLI management of delegated scopes
- Complex cgroup hierarchies
- Different hierarchies across v1 controllers
- Different hierarchies across v1 and unified controllers
- Threaded cgroups
- Transient cgroups
- Cgroup settings not managed by systemd, e.g. cpusets
- Support for Python bindings
Upstream libcgroup Activity

Upstream libcgroup Commits by Year

Year

Commits

Enterprises Application Example

Application A
- cpu time
- long term support
- support v1

libcgroup (v1)
v0.34

Jan 2008

libcgroup (v1)
Distro adoption

Jun 2009

Applicaion A

libcgroup (v2)
v2.0

Mar 2016

Application A

Distro adoption

libcgroup v2
v3.16
(experimental)

Aug 2014

Distro adoption

libcgroup v2
v4.5

May 2021

Application A

Distro adoption

libcgroup (v2)
Distro adoption

Jul 2021

Distro adoption

libcgroup v3.0
abstraction layer

Aug 2022

Application A

v6.6

JuLy 2023

Application A

Support v2

Oct 2023

Application A

Support v1/v2

2021

Application A

Support v1

2011

Application A

Support v2

2022

Application A

Support v2

2023

Application A

Support v2
Enterprise Application Life Cycle

Application A
Release X

- libcgroup (v1)
  - v0.34
- cgroup v2
  - v3.16 (experimental)
- cgroup v2
  - v4.5 (stable)

Application A
Release X + 1

- libcgroup (v2)
  - v2.0

Application A
Release X + 2

- libcgroup v3.0
  - abstraction layer
- kernel
  - v6.6

Application A
Release X + 1

- default boot=cgroup hybrid, supports=cgroup v1, cgroup v2

Application A
Release X + 2

- default boot=cgroup v1, supports=cgroup v1, cgroup v2

Application A
Release X

- default boot=nocgroup, supports=cgroup v1

Application A
Release X + 2

- Distro adoption

Application A
Release X + 1

- Distro adoption

Application A
Release X

- Distro adoption

Application A
Release X + 2

- Distro adoption

Application A
Release X + 1

- systemd support
Enterprise Application Life Cycle

Application A
Release X

cgroup v1
v2.6.24
Jan 2008

Application A
Release X

cgroup v2
v3.16
(experimental)
Aug 2014

Application A
Release X support ends

cgroup v2
v4.5
(stable)
Mar 2016
May 2021

Application A
Release X + 2

kernel
v6.6
Oct 2023

Application A
Release X + 1

Application A
Release X + 2 support ends

Application A
Release X + 1
support ends

2034

Enterprise Application Life Cycle
Challenges for Enterprise Users

Exclusive cpuset and root cgroup
- v2 - a process in the root cgroup cannot access a child cgroup's exclusive cpuset cpus
- v1 - root cgroup entirely ignores child cgroup's exclusive cpusets

Exclusive cpuset and child cgroups
- v6.6 - parent cgroups set as root can exclusively allocate all cpus to their children
- v5.15 - parent cgroups must keep at least 1 cpu available to themselves

CPU controller and root cgroup
- Given the default cpu.shares/cpu.weight, one process in the root cgroup has equal weight to a child cgroup
- Inconsistent spawning of processes in the root cgroup vs a child cgroup like system.slice

Single writer rule
- systemd owns the root cgroup and its children in the root cgroup
- A delegated cgroup (company.slice/app.scope) is the only approved method for a user-managed cgroup
- Generally more strictly enforced on cgroup v2 and newer versions of systemd
- This rule didn’t exist during early cgroup adoption
libcgroup features to assist moving from cgroup v1 to v2

Abstraction layer support

- Maps v1 controller settings to v2 controller settings and vice versa

```
cgxset -1 -r cpu.shares=512 <cgroup>
cgroup_convert_cgroup(out_cgroup, out_version, in_cgroup, in_version);
```

- Not all mappings are available, especially multiline controller settings.

Systemd delegation support

- Allow creation of systemd slice and scope (transient) programatically and tools to support them

```
cgcreate -S -c
  -gcpu,mem:libcgroup.slice/database.scope

cgroup_create_scope2(cgroup, ignore_ownership,
  cgroup_systemd_scope_opts);
```

Abstraction layer support cgconfig.conf

- `/etc/cgconfig.conf` will automap v1 settings to v2 settings

```
group foo {
  cpu {
    cpu.shares=512;
  }
}
```

Systemd delegation support cgconfig.conf

- `/etc/cgconfig.conf` will automap v1 settings to v2 settings

```
systemd {
  slice = database.slice;
  scope = db.scope;
  setdefault = yes;
}
```
Discussion Topics

• Re-adoption of libcgroup into distros

• Are there upcoming features (cgroups, systemd, etc.) that could break assumptions made by these legacy applications?

• Are there upcoming features that may be of interest to these legacy applications?

• Strategies for applications that must support cgroup v1/v2, new and old systemd versions, and a wide range of kernel versions
  • See inconsistencies slide from earlier

• Ways to increase “proper” usage of cgroups – single writer rule, no-inner-node processes, etc.

• Ways to discourage improper usage of cgroups
Thank You
<table>
<thead>
<tr>
<th>Controllers</th>
<th>Kernel Version</th>
<th>libcgroup Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory, I/O, pids controllers</td>
<td>- v4.5 (Mar 2016)</td>
<td></td>
</tr>
<tr>
<td>rdma controller</td>
<td>- v4.11 (Apr 2017)</td>
<td></td>
</tr>
<tr>
<td>cpu controller</td>
<td>- v4.15 (Jan 2018)</td>
<td></td>
</tr>
<tr>
<td>block I/O latency controller</td>
<td>- v4.19 (Oct 2018)</td>
<td></td>
</tr>
<tr>
<td>Pressure Stall Information</td>
<td>- v4.20 (Dec 2018)</td>
<td></td>
</tr>
<tr>
<td>cpuset v2</td>
<td>- v5.0 (Mar 2019)</td>
<td></td>
</tr>
<tr>
<td>freezer</td>
<td>- v5.2 (Jul 2019)</td>
<td></td>
</tr>
<tr>
<td>Kernel Release</td>
<td>- v5.5 (Jan 2020)</td>
<td>libcgroup v0.42</td>
</tr>
<tr>
<td>hugetlb</td>
<td>- v5.6 (Mar 2020)</td>
<td></td>
</tr>
<tr>
<td>Kernel Release</td>
<td>- v5.12 (Apr 2020)</td>
<td>libcgroup v2.0</td>
</tr>
<tr>
<td>misc</td>
<td>- v5.13 (Jun 2021)</td>
<td></td>
</tr>
<tr>
<td>Kernel Release</td>
<td>- v5.17 (Mar 2022)</td>
<td>libcgroup v2.0.1</td>
</tr>
<tr>
<td>Kernel Release</td>
<td>- v5.18 (May 2022)</td>
<td>libcgroup v2.0.2</td>
</tr>
<tr>
<td>Kernel Release</td>
<td>- v5.19 (Jul 2022)</td>
<td>libcgroup v3.0</td>
</tr>
<tr>
<td>Kernel Release</td>
<td>- v6.2 (Feb 2023)</td>
<td>libcgroup v2.0.3</td>
</tr>
</tbody>
</table>
## cgroup adoption

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Release Version</th>
<th>Release Date</th>
<th>libcgroup version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Linux</td>
<td>9</td>
<td>Jun 2022</td>
<td>3.0</td>
</tr>
<tr>
<td>RedHat Enterprise Linux</td>
<td>9</td>
<td>May 2022</td>
<td>N/A</td>
</tr>
<tr>
<td>SuSE Linux Enterprise Server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ubuntu Server</td>
<td>21.10</td>
<td>Oct 2021</td>
<td>2.0.2.2</td>
</tr>
</tbody>
</table>