



Classifying processes into control groups

Michal Koutný
L3 Support Engineer
mkoutny@suse.com

Agenda

- Kernel APIs
- libcg
- systemd
- Problems
- Conclusion

Kernel APIs

Control groups

- Multiple hierarchies

```
host:~ # grep cgroup /proc/mounts
cgroup /sys/fs/cgroup/systemd cgroup rw,nosuid,nodev,noexec,
    relatime,xattr,name=systemd 0 0
cgroup /sys/fs/cgroup/memory cgroup rw,nosuid,nodev,noexec,
    relatime,memory 0 0
cgroup /sys/fs/cgroup/pids cgroup rw,nosuid,nodev,noexec,relatime,
    pids 0 0
...
```

- Multiple membership

```
host:~ # cat /proc/self/cgroup
12:pids:/user.slice/user-0.slice/session-2.scope
...
6:memory:/
...
1:name=systemd:/user.slice/user-0.slice/session-2.scope
```

Control groups – continuation

- Cgroup sysfs API

```
host:~ # ls -g /sys/fs/cgroup/systemd/
drwxr-xr-x  2 root 0 Sep 12 19:48 init.scope
drwxr-xr-x 68 root 0 Sep 12 19:48 system.slice
drwxr-xr-x  4 root 0 Sep 12 19:48 user.slice
-rw-r--r--  1 root 0 Sep 12 19:48 cgroup.procs
-rw-r--r--  1 root 0 Sep 12 19:48 notify_on_release
-rw-r--r--  1 root 0 Sep 12 19:48 release_agent
-rw-r--r--  1 root 0 Sep 12 19:48 tasks
...
```

- Cgroup v2

```
host:~ # ls -g /sys/fs/cgroup/unified/
drwxr-xr-x  2 root 0 Sep 12 10:24 init.scope
drwxr-xr-x 68 root 0 Sep 12 19:31 system.slice
drwxr-xr-x  4 root 0 Sep 12 19:08 user.slice
-r--r--r--  1 root 0 Sep 12 19:48 cgroup.controllers
-rw-r--r--  1 root 0 Sep 12 19:48 cgroup.procs
-rw-r--r--  1 root 0 Sep 12 19:48 cgroup.subtree_control
```

Cgroup v2

- Tejun Heo, Johannes Weiner: Resource Control @FB
- <https://goo.gl/7JCgja>

Process events API

- Netlink socket address group CN_IDX_PROC

```
/* /usr/include/linux/cn_proc.h */
struct proc_event {
    enum what {
        PROC_EVENT_NONE = 0x00000000,
        PROC_EVENT_FORK = 0x00000001,
        PROC_EVENT_EXEC = 0x00000002,
        PROC_EVENT_UID = 0x00000004,
        PROC_EVENT_GID = 0x00000040,
        PROC_EVENT_SID = 0x00000080,
        PROC_EVENT_PTRACE = 0x00000100,
        PROC_EVENT_COMM = 0x00000200,
        PROC_EVENT_COREDUMP = 0x40000000,
        PROC_EVENT_EXIT = 0x80000000
    } what;
    /* ... */
}
```

libcg

libcg – configparser

- Constructs hierarchies per description

```
# /etc/cgconfig.conf

group daemons/www {
    cpu {
        cpu.shares = 1000;
    }
}

group daemons/ftp {
    cpu {
        cpu.shares = 500;
    }
}

mount {
    cpu = /mnt/cgroups/cpu;
    cpuacct = /mnt/cgroups/cpuacct;
}
```

libcg – cgrulesengd

- Uses CN_IDX_PROC API
- Puts processes into groups according to rules

```
# /etc/cgrules.conf
```

```
@students    cpu,cpuacct  /students/%u  
student:cp   *            /usergroup/students/cp
```

libcg – helpers

- Start a process in the cgroup

```
cgexec -g cpu,memory:test1 ls -l
```

- Apply rules in batch mode

```
cgclassify $PID ...
```

Question

Question

- What is the difference between `cgroup.procs` and `tasks` control files? (cgroup v1)

```
host:~ # ls -lg /sys/fs/cgroup/systemd/  
...  
-rw-r--r--  1 root 0 Sep 12 19:48 cgroup.procs  
-rw-r--r--  1 root 0 Sep 12 19:48 tasks  
...
```

Question

- What is the difference between `cgroup.procs` and `tasks` control files? (cgroup v1)

```
host:~ # ls -g /sys/fs/cgroup/systemd/  
...  
-rw-r--r--  1 root 0 Sep 12 19:48 cgroup.procs  
-rw-r--r--  1 root 0 Sep 12 19:48 tasks  
...
```

- Answer: Processes vs threads granularity.

systemd

systemd and cgroups

- systemd mounts cgroup filesystems
- systemd places processes in specific cgroups
 - .service
 - .scope
- tracking running processes
- realizing resource limits

systemd and cgroups

- systemd mounts cgroup filesystems
- systemd places processes in specific cgroups
 - .service
 - .scope
- tracking running processes
- realizing resource limits

"512 PIDs should be enough for everyone."

systemd cgroups hierarchy

- `systemd{,-logind,-machined}`

```
host:~ # systemd-cgls
-.slice
|-system.slice
|  |-dbus.service
|  |  `--712 /usr/bin/dbus-daemon --system --address=syste...
|  |-sshd.service
|  |  `--1660 /usr/sbin/sshd -D
|  `--...
|-user.slice
|  |-user-0.slice
|  |  `--...
|  `--user-471.slice
|     `--...
|-machine.slice
|  `--machine-qemu\x218machine\x2dname.scope
|     `--7499 /usr/bin/qemu-system-x86_64 -name guest=mach...
`--init.scope
   `--1 /usr/lib/systemd/systemd --switched-root --system...
```

systemd user.slice

- Managed by pam_systemd
- Subpartitioning with Delegate=yes

```
host:~ # systemd-cgls /user.slice
user.slice:
`-user-0.slice
  |-session-5.scope
  | |-8645 sshd: root@pts/1
  | `--8647 -bash
  |-session-8.scope
  | |- 1961 login -- root
  | `--13151 -bash
  `--user@0.service
    |-dbus.service
    | `--7330 /usr/bin/dbus-daemon --session --address=s...
    `--init.scope
      |-6648 /usr/lib/systemd/systemd --user
      `--6649 (sd-pam)
```

Problems

Race condition of CN_IDX_PROC

- Process 1 (cgrulesengd)
 - receive PROC_EVENT_FORK/PROC_EVENT_EXEC
 - read process metadata (UID, GID)
 - classify process via sysfs API
- Process 2 (to be classified)
 - fork, exec
 - do the work

Race condition of CN_IDX_PROC

- Process 1 (cgrulesengd)
 - receive PROC_EVENT_FORK/PROC_EVENT_EXEC
 - read process metadata (UID, GID)
 - classify process via sysfs API
- Process 2 (to be classified)
 - fork, exec
 - do the work
- No ordering between classification and process 2 execution

Race condition of CN_IDX_PROC

- Process 1 (cgrulesengd)
 - receive PROC_EVENT_FORK/PROC_EVENT_EXEC
 - read process metadata (UID, GID)
 - classify process via sysfs API
- Process 2 (to be classified)
 - fork, exec
 - do the work
- No ordering between classification and process 2 execution
- Solution
 - classify first, execute second
 - example

```
systemd-run --slice=dest.slice --scope $COMMAND $ARGS
```

Conflict between systemd and libcg

- *"cpu quota of cgconfig doesn't work after hostnamedctl execution"*

```
ExecStart=/usr/bin/cgexec -g cpu:group1 /usr/bin/daemon
```


Conflict between systemd and libcg

- *"cpu quota of cgconfig doesn't work after hostnamed/ execution"*

```
ExecStart=/usr/bin/cgexec -g cpu:group1 /usr/bin/daemon
```

```
int unit_realize_cgroup(Unit *u) {
    unit_queue_siblings(u);
    return unit_realize_cgroup_now(u, manager_state(u->manager
    ));
}

void unit_queue_siblings(Unit *u) {
    SET_FOREACH(m, u->siblings) {
        if (unit_has_mask_realized(m, unit_get_target_mask
            (m)))
            continue;
        unit_add_to_cgroup_queue(m);
    }
}
```

- processes of seemingly unrelated services may be reclassified

Conflict between systemd and libcg – solution

- libcg is fading out
- systemd takes over cgroup management

Conflict between systemd and libcg – solution

- libcg is fading out
- systemd takes over cgroup management
 - but...

libcg and systemd feature parity

- classifying based on running executable
 - racy anyway
- classifying into templated cgroups
 - configured a slice tree
 - user slices attached *arbitrarily*
- more unspecified use cases



EVERY CHANGE BREAKS SOMEONE'S WORKFLOW.

Source: <https://xkcd.com/1172/>

sudo in init scripts

```
# /etc/pam.d/sudo
...
session include common-session
...
```

```
# /etc/pam.d/common-session
...
session optional pam_systemd.so
...
```

- daemon processes are extracted from service cgroup

sudo in init scripts

```
# /etc/pam.d/sudo
...
session include common-session
...
```

```
# /etc/pam.d/common-session
...
session optional pam_systemd.so
...
```

- daemon processes are extracted from service cgroup
- port init script into systemd service
- use `setpriv`
- modify PAM configuration

Conclusion

- cgroups are meant to be generic
- libcg uses interesting but nonfunctional API
- systemd usurps cgroups (but allows delegation)
- systemd does not solve everything
- libcg can be dropped (in favor of cgroup v2)

Q & (A)

The end