SUSE

How is kernel getting along with many cgroups

Michal Koutný <mkoutny@suse.com>

LPC 2024, Wien

Outline

- Assumed use cases
- Considered aspects
- Changes done in the past
- Changes (not) done ~ proposals
- Other ideas
- Discuss (anytime), complain

Assumed use cases

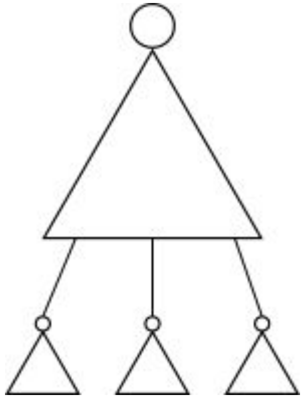
- No cgroups (singular trees)
- Single server
- Container host
- Desktop
- (v1 setups)
- Specific setups

```
CONFIG_CGROUPS=y
CONFIG_CGROUP_*=y
CONFIG_BLK_CGROUP=y
CONFIG_CPUSETS=y
CONFIG_RT_GROUP_SCHED=n
CONFIG_SCHED_AUTOGROUP=n

CONFIG_MEMCG=y
CONFIG_MEMCG_KMEM=y
CONFIG_CGROUP_DEBUG=n
```

Considered aspects

- Locking
  - cgroup_mutex
  - cgroup_threadgroup_rwsem
  - controllers' locks
- Full (sub)tree operations
  - stats
  - offlined objects
  - memory reclaim
  - (userspace iterations?)
- Full depth operations
  - stats, charging
  - group scheduling
- Memory footprint
  - data overhead
  - fragmentation



Changes done in the past

- cgroup_mutex
  - `9067d90006df0` ("cgroup: Eliminate the need for cgroup_mutex in proc_cgroup_show()") v6.8-rc1~182^2~16
  - `822bc9bac9e9a` ("cgroup: no need for cgroup_mutex for /proc/cgroups") v5.16-rc1~146^2~2
  - `bb758421416fd` ("cgroup: remove cgroup_mutex from cgroupstats_build") v5.16-rc1~146^2~3
  - `be288169712f3` ("cgroup: reduce dependency on cgroup_mutex") v5.16-rc1~146^2~4
- cgroup_threadgroup_rwsem
  - `6a010a49b63ac` ("cgroup: Make !percpu threadgroup_rwsem operations optional") v6.0-rc1~157^2~2
- rstat improvements
  - precision vs overhead tradeoff: conditional and periodic flushing
  - `3b8cc62987240` ("blk-cgroup: Optimize blkcg_rstat_flush()") v6.2-rc1~129^2~68
  - `7bd5bc3ce9632` ("mm: memcg: normalize the value passed into memcg_rstat_updated()") v6.7-rc1~90^2~208
  - `8d59d2214c236` ("mm: memcg: make stats flushing threshold per-memcg") v6.8-rc1~180^2~203
  - `21c38a3bd4ee3` ("cgroup/rstat: add cgroup_rstat_cpu_lock helpers and tracepoints") v6.10-rc1~138^2
  - `ff48c71c26aae` ("memcg: reduce memory for the lruvec and memcg stats") v6.10-rc1~105^2~40

Changes not done ~ proposals

- Cleaning up of offlined memcgs traversal
  - offlined memcgs should be only memory not time garbage
  - mem_cgroup_scan_tasks may skip offline memcgs (zombies at most)
  - v1 only
    - mem_cgroup_mark_under_oom needn't process offline memcg
    - mem_cgroup_oom_trylock
    - mem_cgroup_oom_notify
  - writeback on behalf of offlined memcgs and blkcgs?
- Does damon_sysfs_memcg_path_to_id need traversal?
- BPF cgroup iterator's locking?
- More cond_rescheds?
- [PATHC v3 -next 0/3] Some optimizations about freezer

Other proposals (broader scope)

- More VMs on one physical machine
  - partitioning whole kernels
- Getting out of way in latency sensitive paths
  - sched_ext group scheduling
  - memcg deferred charging

SUSE

How is kernel getting along with many cgroups?

The latest kernel – well enough (until anyone notices).

References

- LPC 2022, cgroup rstat's advanced adoption
- [RFC] memcg rstat flushing optimization
- Authors of referenced commits: Yosry Ahmed, Shakeel Butt, Jesper Dangaard Brouer, Tejun Heo, Waiman Long, Chen Ridong, Yafang Shao,...
- UATC 2022, RunD: A Lightweight Secure Container Runtime for High-density Deployment and High-concurrency Startup in Serverless Computing

SUSE

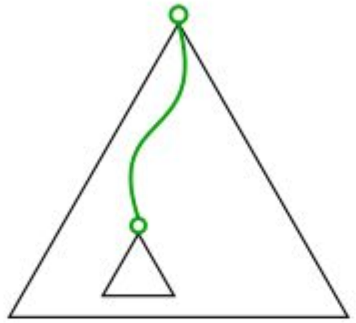
Extra slides

cgroup_threadgroup_rwsem

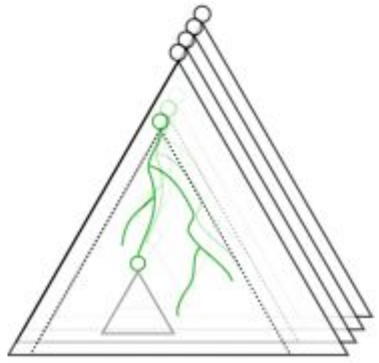
- Inverted lock
- Readers: fork, exit (~invisible)
- Writers: cgroup migration (exclusive, stability)
- Conveniently unnecessary with CLONE_INTO_CGROUP
- Migrations vs fork/exit trade-off
  - `favordynmods` mount option to favor migrations at expense of fork/exit
- Implemented as percpu_rw_semaphore
  - Cheap for readers (this_cpu_inc)
  - Expensive for writers (rcuwait, ~RCU(?))

rstat

- local-only writers (per-cpu)
  - `cgroup_rstat_updated(cgrp, cpu)`
  - building and update tree
  - `cgroup_rstat_cpu_lock`

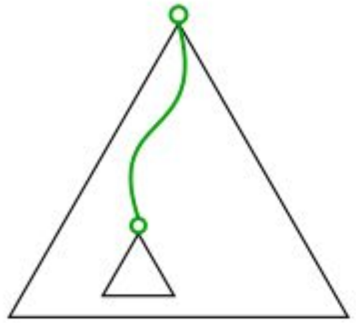


- aggregating readers (flushing)
  - `->css_rstat_flush(css, cpu)`
  - only processing cases from the update tree
  - `cgroup_rstat_lock`
  - with inter cpu `cond_resched`



rstat – memcg

- writers (per-cpu)
  - per-cpu and memcg error tracking
  - MEMCG_CHARGE_BATCH



- flushing
  - periodic flushing (0.5/s)
  - conditional subtree flushing (based on error)
  - rate-limited (sub)tree flushing (based on delay)
  - No flushing on CPU hotplug (Bug? Fixes: 7e1c0d6f58207)