facebook

cgroupv2: Linux's new unified control group system

Chris Down (cdown@fb.com)
Production Engineer, Web Foundation

In this talk

- Short intro to cgroups
- v1, what went well, what didn't
- Design decisions in v2
- New features
- Future work

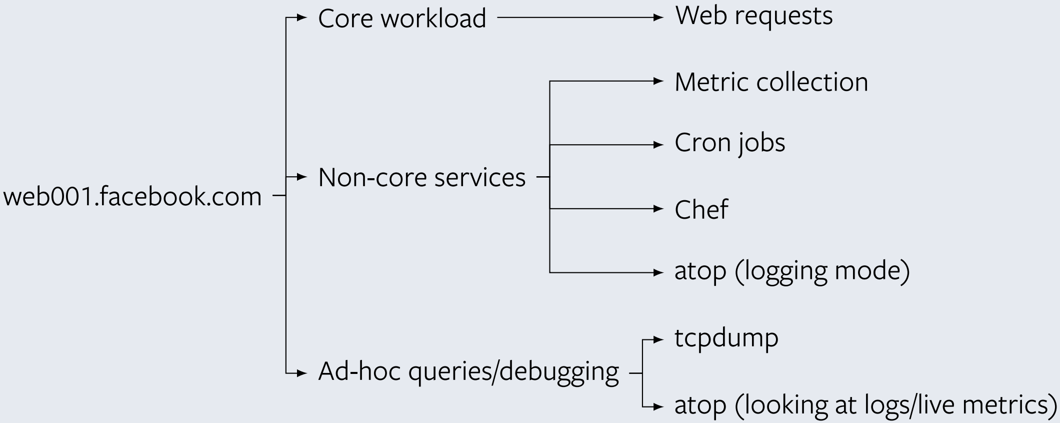
About me

- Chris Down
- Production Engineer, Web Foundation
- Working at Facebook London
- `cdown@fb.com`

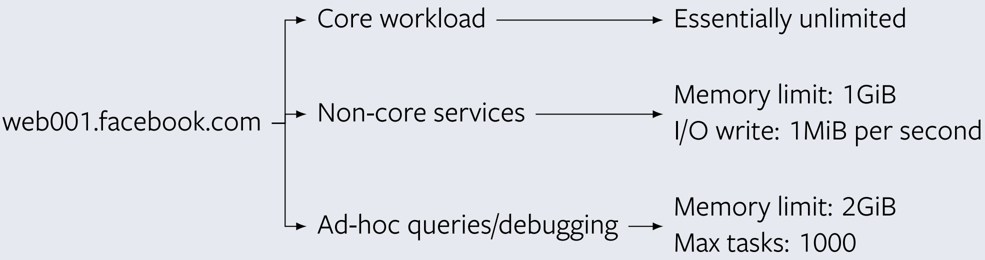
Why WF cares about cgroupv2

- >100,000 servers
- Many thousands of services
- Want to limit failure domains

The problem


web001.facebook.com
Core workload
Web requests
Non-core services
Metric collection
Cron jobs
Chef
atop (logging mode)
Ad-hoc queries/debugging
tcpdump
atop (looking at logs/live metrics)

Limits



web001.facebook.com

Core workload → Essentially unlimited

Non-core services → Memory limit: 1GiB
I/O write: 1MiB per second

Ad-hoc queries/debugging → Memory limit: 2GiB
Max tasks: 1000

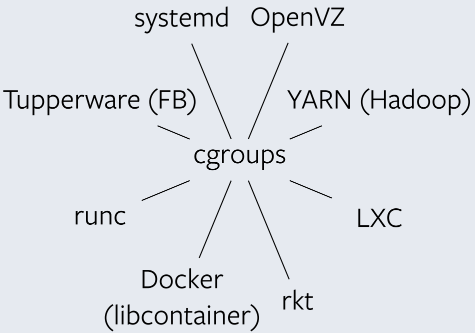
What are cgroups?

- cgroup ≡ control group
- System for resource management on Linux
- Directory hierarchy at `/sys/fs/cgroup`
- Limit, throttle, and account for resource usage per control group
- Each resource interface is provided by a *controller*

Practical uses

- Isolating core workload from background resource needs
  - Web server vs. system processes (eg. Chef, metric collection, etc)
  - Time critical work vs. long-term asynchronous jobs
- Don’t allow one workload to overpower the others

Who uses cgroups?



systemd

OpenVZ

Tupperware (FB)

YARN (Hadoop)

cgroups

runc

LXC

Docker (libcontainer)

rkt

How did this work in cgroupv1?

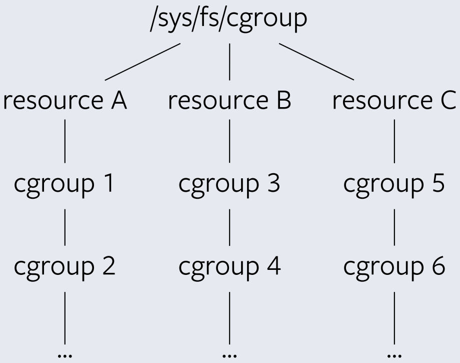
cgroupv1 has a hierarchy per-resource, for example:

```
% ls /sys/fs/cgroup
cpu/ cpuacct/  cpuset/  devices/  freezer/
memory/  net_cls/  pids/
```

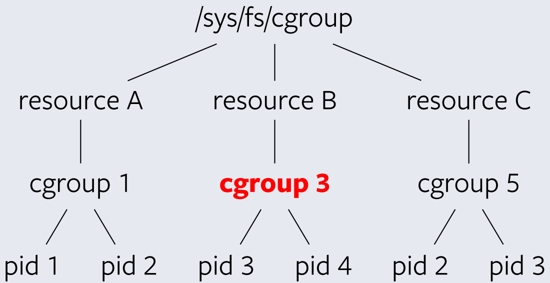
Each resource hierarchy contains cgroups for this resource:

```
% find /sys/fs/cgroup/pids -type d
/sys/fs/cgroup/pids/background.slice
/sys/fs/cgroup/pids/background.slice/async.slice
/sys/fs/cgroup/pids/workload.slice
```

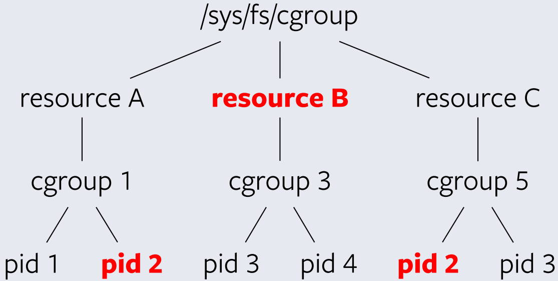
How did this work in cgroupv1?



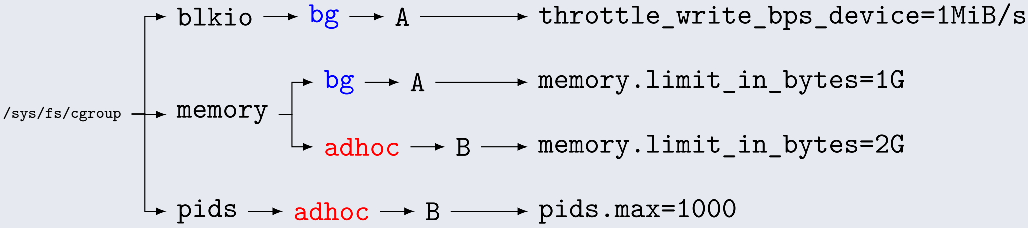
How did this work in cgroupv1?



How did this work in cgroupv1?



Hierarchy in cgroupv1



/sys/fs/cgroup → blkio → bg → A → throttle_write_bps_device=1MiB/s

/sys/fs/cgroup → memory → bg → A → memory.limit_in_bytes=1G

/sys/fs/cgroup → memory → adhoc → B → memory.limit_in_bytes=2G

/sys/fs/cgroup → pids → adhoc → B → pids.max=1000

How does this work in cgroupv2?

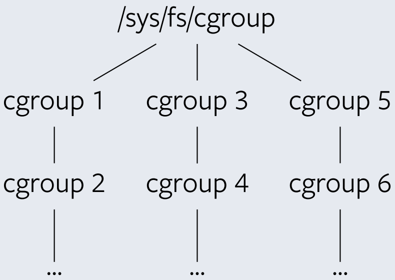
cgroupv2 has a *unified hierarchy*, for example:

```
% ls /sys/fs/cgroup
background.slice/  workload.slice/
```

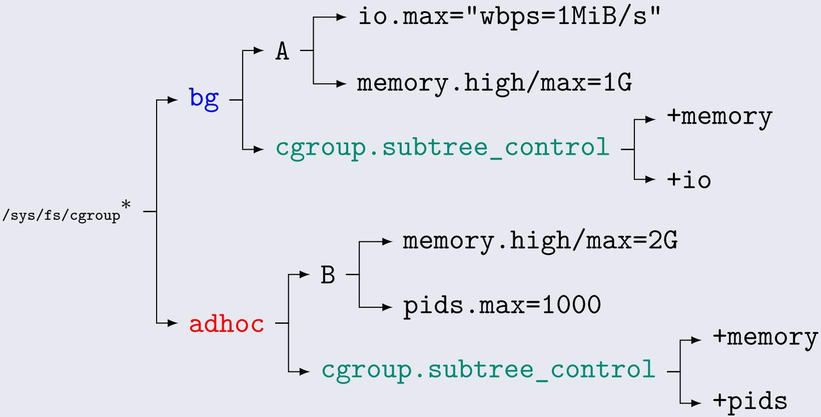
Each cgroup can support multiple resource domains:

```
% ls /sys/fs/cgroup/background.slice
async.slice/  foo.mount/  cgroup.subtree_control
memory.high  memory.max  pids.current  pids.max
```

How does this work in cgroupv2?

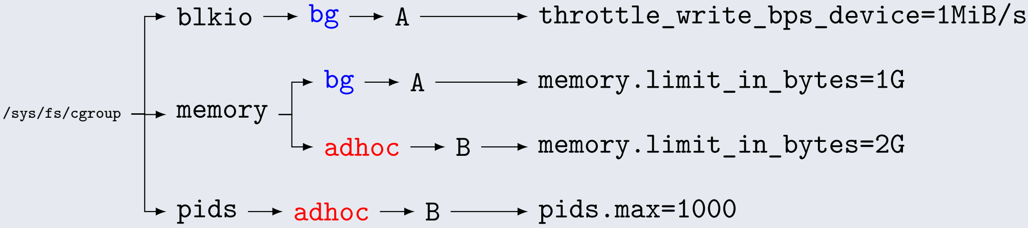


Hierarchy in cgroupv2

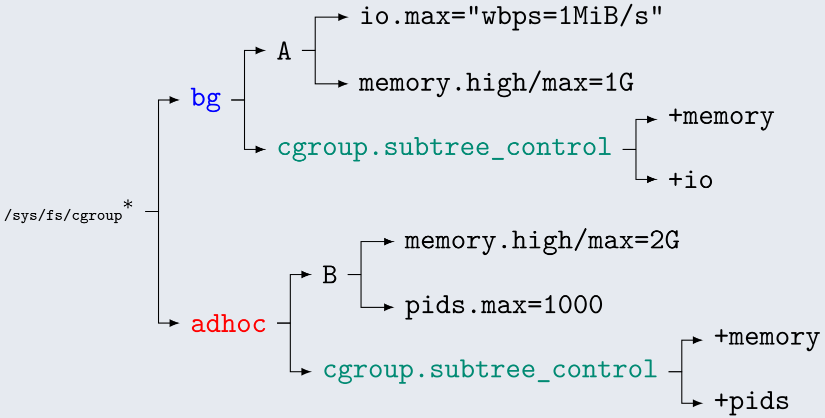


* in real life, we must enable memory/pids/io controllers for children here too

Hierarchy in cgroupv1



Hierarchy in cgroupv2

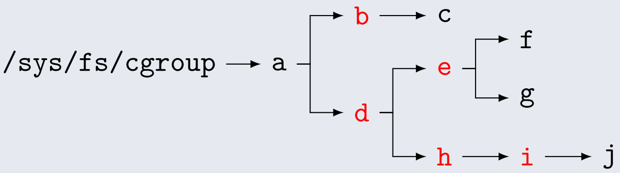


* in real life, we must enable memory/pids/io controllers for children here too

Fundamental differences between v1 and v2

- Unified hierarchy — resources apply to cgroups now
- Granularity at TGID (PID), not TID level
- Focus on simplicity/clarity over ultimate flexibility

"No internal process" constraint



Practical improvements in v2

v2 improvements: Tracking of non-immediate/multi-resource charges

In v1:

- No tracking of non-immediate charges
- Charged to root cgroup, essentially unlimited

In v2:

- Page cache writebacks and network are charged to the responsible cgroup
- Can be considered as part of cgroup limits

v2 improvements: Communication with backing subsystems

In v1:

- Most actions for non-share based resources reacted crudely to hitting thresholds
- For example, in the memory cgroup, the only option was to OOM kill or freeze*

In v2:

- Many cgroup controllers negotiate with subsystems before real problems occur
- Subsystems can take remediative action (eg. direct reclaim with `memory.high`)
- Easier to deal with temporary spikes in a resource's usage

* soft_limit_in_bytes exists in v1, but it's too overloaded and abstract to be useful

v2 improvements: Saner notifications

In v1:

- One `clone()` per event for cgroup release, expensive
- `eventfd()` support for others

In v2:

- inotify support everywhere
- `eventfd()` support still exists
- One process to monitor everything, if you like

v2 improvements: Utility controllers make sense now

In v1:

- Utility controllers have their own hierarchies
- We usually want to use processes from another hierarchy
- As such, we end up manually synchronising 😟

In v2:

- We have a single unified hierarchy, so no sync needed

v2 improvements: Consistency between controllers

In v1:

- Inconsistencies in controller APIs
- Some controllers don't inherit values

In v2:

- Our crack team of API Design Experts have ensured your sheer delight*

* well, at least we can pray we didn't screw it up too badly

v2 improvements: Unified limits

In v1:

- Some limitations could not be fixed due to backwards compatibility
- `memory.{,kmem.,kmem.tcp.,memsw.,[...]}limit_in_bytes`

In v2:

- Less iterative, more designed up front
- We now have universal thresholds (eg. `memory.{high,max}`)

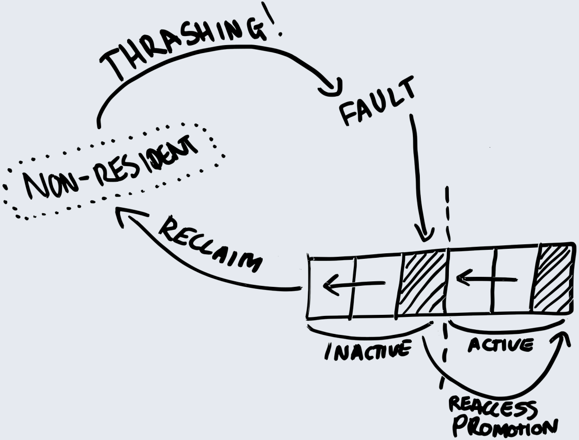
Use at FB

- ~10% of web servers
- Already getting better results on spiky workloads
- Separation of workload services from system services (eg. Chef, metric collection)
- Running managed with systemd (see “Deploying systemd at scale” talk)

Current support

- Merged:
  - I/O
  - Memory
  - PID
- Complete, but pending merge:
  - CPU (thread: `https://bit.ly/cgroupv2cpu`)

v2 big bets: Real memory pressure detection



Future work

- CPU accounting for "tracked" events (eg. page cache writeback)
- Work-conserving I/O controller
- Accounting for filesystem metadata updates
- Better metrics around memory pressure (eg. refault metrics)
- Freezer for v2

How can I get it?

- `systemd.unified_cgroup_hierarchy=1`
  - systemd will mount `/sys/fs/cgroup` as cgroupv2
  - Available from systemd v226 onwards
- `cgroup_no_v1=all`
  - The kernel will disable all v1 cgroup controllers
  - Available from Linux 4.6 onwards

Manual mounting:

```
% mount -t cgroup2 none /sys/fs/cgroup
```

Links

- Docs at https://bit.ly/cgroupv2
- Come ask me questions 😊

facebook