



7 years of cgroup v2

The future of Linux resource control

Chris Down

Kernel, Meta

<https://chrisdown.name>

Downloads

Please select the amount of RAM to download:

1GB



Overview

- * 1GB CT12864AA800 Memory
- * 240-pin DIMM
- * DDR2 PC2-6400, CL=6

Was: ~~\$99.99~~ Now: **FREE**

[Download Now](#)

2GB



Overview

- * 2 GB (2 x 1 GB)
- * 240-pin DIMM
- * DDR2 800 MHz (PC2-6400)

Was: ~~\$149.99~~ Now: **FREE**

[Download Now](#)

4GB



Overview

- * 4 GB (2 x 2 GB)
- * 240-pin DIMM
- * DDR2 800 MHz (PC2-6400)

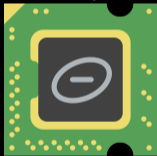
Was: ~~\$199.99~~ Now: **FREE**

[Download Now](#)



Lance Cheung, CC BY-NC-SA: bit.ly/sevimage

server





Filmed at
QCon London 2017

Brought to you by
InfoQ
UNIQUE

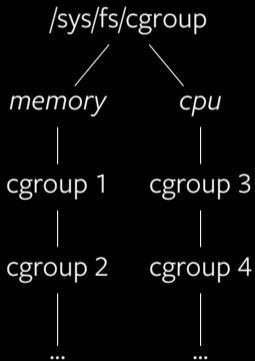
facebook

cgroupv2: Linux's new unified control group system

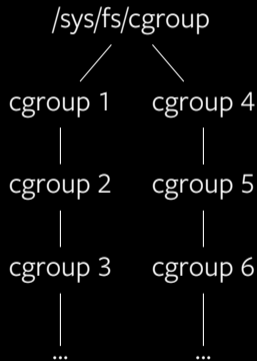
Chris Down (cdown@fb.com)
Production Engineer, Web Foundation

bit.ly/cgv2qcon

v1, each cgroup exists in the context of only one resource:



v2, {en,dis}able resources per-cgroup using `cgroup.subtree_control`:



Why do we need a single resource hierarchy?

Why do we need a single resource hierarchy?

- Memory starts to run out

Why do we need a single resource hierarchy?

- Memory starts to run out
- This causes us to reclaim page caches/swap, causing disk IO

Why do we need a single resource hierarchy?

- Memory starts to run out
- This causes us to reclaim page caches/swap, causing disk IO
- This reclaim costs sometimes non-trivial CPU cycles

Shift to “protection” mentality

- Limits (eg. `memory.{high,max}`) really don't compose well
- Prefer protection (`memory.{low,min}`) if possible
- Protections affect memory reclaim behaviour



bit.ly/fosdem20mm



USE CGROUPS

**CONTROL
RESOURCES**

**DON'T MAKE THE
WEBSITE FALL OVER**

How can you view memory usage for a process in Linux?

How can you view memory usage for a process in Linux?

- SIKE THIS SLIDE WAS A TRAP

```
% size -A chrome | awk '$1 == ".text" { print $2 }'  
132394881
```

```
% cat /proc/self/cgroup
0::/system.slice/foo.service
% cat /sys/fs/cgroup/system.slice/foo.service/memory.current
3786670080
```

- `memory.current` tells the truth, but the truth is sometimes complicated
- Slack grows to fill up to cgroup limits if there's no global pressure


```
% time make -j4 -s  
real    3m58.050s  
user    13m33.735s  
sys     1m30.130s
```

```
# Peak memory.current bytes: 803934208
```

```
% sudo sh -c 'echo 600M > memory.high'
```

```
% time make -j4 -s
```

```
real    4m0.654s
```

```
user    13m28.493s
```

```
sys     1m31.509s
```

```
# Peak memory.current bytes: 629116928
```

```
% sudo sh -c 'echo 400M > memory.high'
```

```
% time make -j4 -s
```

```
real    4m3.186s
```

```
user    13m20.452s
```

```
sys     1m31.085s
```

```
# Peak memory.current bytes: 419368960
```

```
% sudo sh -c 'echo 300M > memory.high'
```

```
% time make -j4 -s
```

```
^C
```

```
real    9m9.974s
```

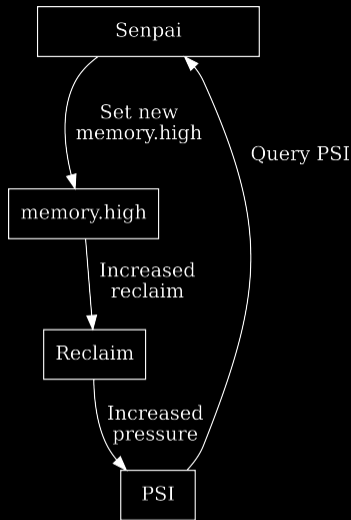
```
user    10m59.315s
```

```
sys     1m16.576s
```

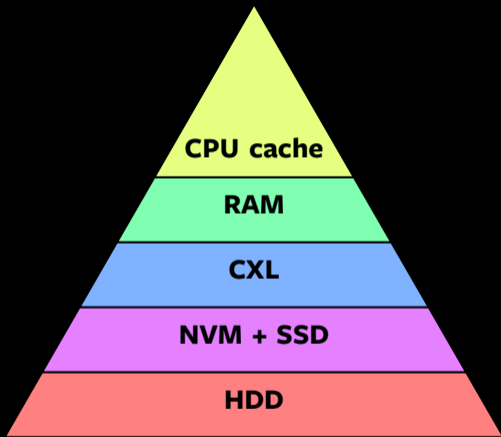
```
% sudo senpai /sys/fs/cgroup/...
2021-05-20 14:26:09
    limit=100.00M pressure=0.00
    delta=8432 integral=8432

% make -j4 -s
[...find the real usage...]

2021-05-20 14:26:43
    limit=340.48M pressure=0.16
    delta=202 integral=202
2021-05-20 14:26:44
    limit=340.48M pressure=0.13
    delta=0 integral=202
```

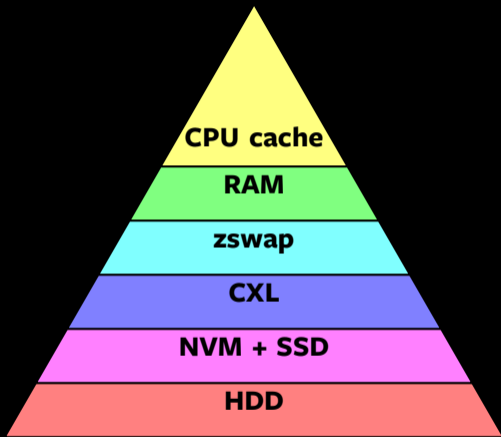


bit.ly/cgsenpai



↑ high cost, low latency

↓ low cost, high latency



↑ high cost, low latency

↓ low cost, high latency

New swap algorithm in kernel 5.8+:

- Repeatedly faulting/evicting a cache page over and over? Evict a heap page instead

New swap algorithm in kernel 5.8+:

- Repeatedly faulting/evicting a cache page over and over? Evict a heap page instead
- We only trade one type of paging for another: we're not adding I/O load

Effects of swap algorithm improvements:

Effects of swap algorithm improvements:

- Decrease in heap memory

Effects of swap algorithm improvements:

- Decrease in heap memory
- Increase in cache memory

Effects of swap algorithm improvements:

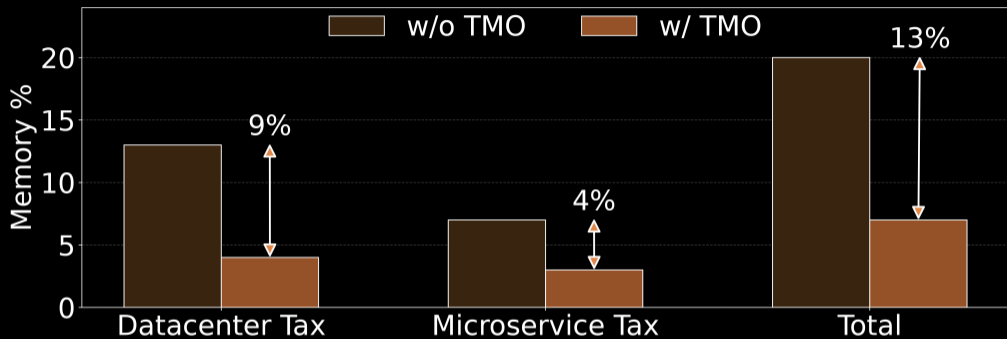
- Decrease in heap memory
- Increase in cache memory
- Increase in web server performance

Effects of swap algorithm improvements:

- Decrease in heap memory
- Increase in cache memory
- Increase in web server performance
- Decrease in disk I/O from paging activity

Effects of swap algorithm improvements:

- Decrease in heap memory
- Increase in cache memory
- Increase in web server performance
- Decrease in disk I/O from paging activity
- Increase in workload stacking opportunities



bit.ly/tmopost

- Memory starts to run out
- This causes us to reclaim page caches/swap, causing disk IO
- This reclaim costs sometimes non-trivial CPU cycles

```
% echo '8:16 wbps=1MiB wiops=120' > io.max
```

```
# target= is in milliseconds  
% echo '8:16 target=10' > io.latency
```





bit.ly/iocost + bit.ly/resctlbench

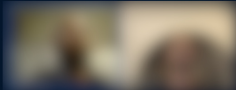
All the cool kids are using it

cgroupv2 users:

- containerd \geq 1.4
- Docker/Moby \geq 20.10
- podman \geq 1.4.4
- runc \geq 1.0.0
- systemd \geq 226

Distributions:

- Fedora uses by default on \geq 32
- Coming to other distributions by default soon™



Mapping processes to apps

- The manager tries to map up windows to .desktop files
- Hoping they report the right things
- We match up audio (by PID) to windows
- With multi processes this is a guessing game

13 / 42

bit.ly/kdecgv2



Try it out:

- `cgroup_no_v1=all` on kernel command line
- Docs: bit.ly/cgroupv2doc
- Whitepaper: bit.ly/cgroupv2wp

Feedback:

- E-mail: chris@chrisdown.name
- Mastodon: [@cdown@fosstodon.org](https://fosstodon.org/@cdown)

∞ Meta