facebook

5 years of cgroup v2

The future of Linux resource control

Chris Down
Kernel, Facebook
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

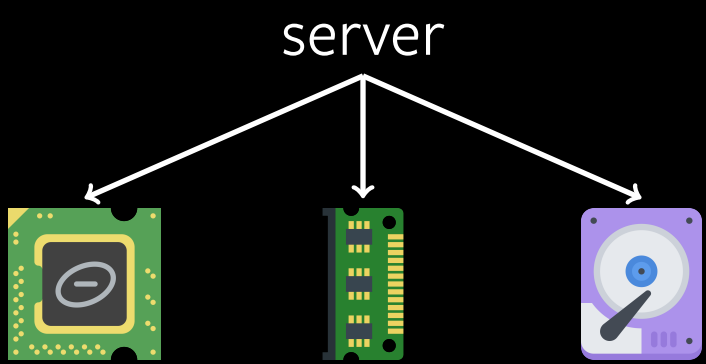
Overview

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

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

Download Now

Image: Spc. Christopher Hernandez, US Military Public Domain


server

Filmed at QCon London 2017

Brought to you by InfoQ

facebook

cgroupv2: Linux's new unified control group system

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

How did this work in cgroupv1?

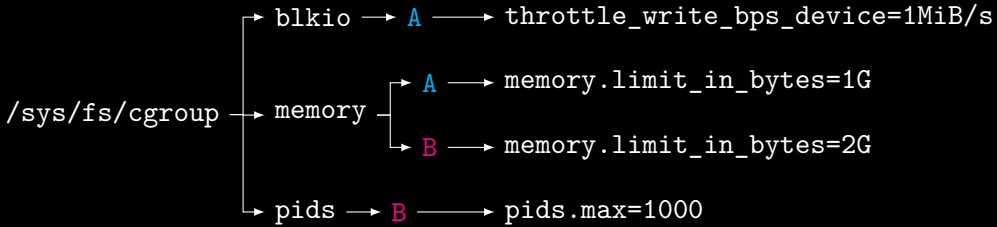
cgroupv1 has a hierarchy per-resource, for example:

```
% 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/memory -type d
/sys/fs/cgroup/memory/background.slice
/sys/fs/cgroup/memory/background.slice/sshd.service
/sys/fs/cgroup/memory/workload.slice
```

Hierarchy in cgroupv1



How does this work in cgroupv2?

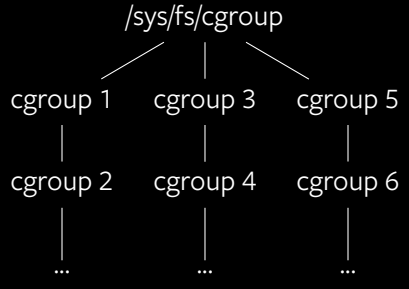
cgroupv2 has a *unified hierarchy*, for example:

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

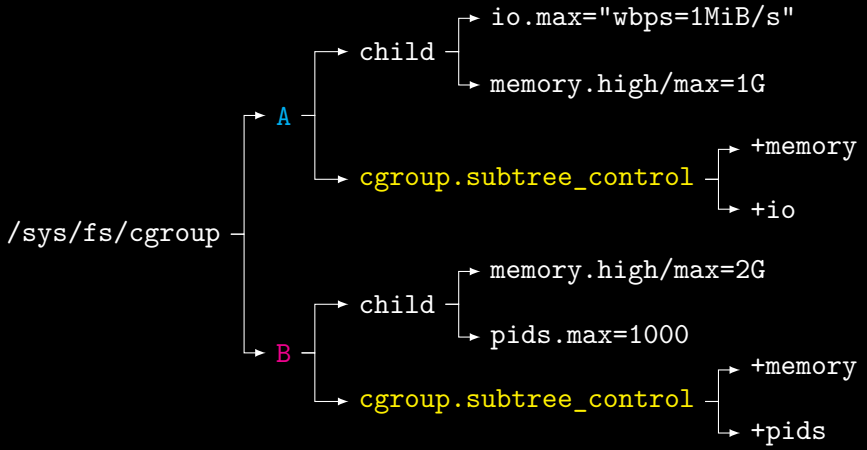
Each cgroup can support multiple resource domains:

```
% 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



Multi-resource actions

In v1:

- No tracking of actions which span multiple resources
- No tracking of asynchronous actions

In v2:

- Page cache writebacks, network, etc are charged to the responsible cgroup
- Can be considered as part of cgroup limits and dealt with accordingly

From: Linus Torvalds <torvalds@linux-foundation.org>
To: linux-kernel@vger.kernel.org
Date: Sun, 13 Mar 2016 21:53:34 -0700
Subject: Linux 4.5

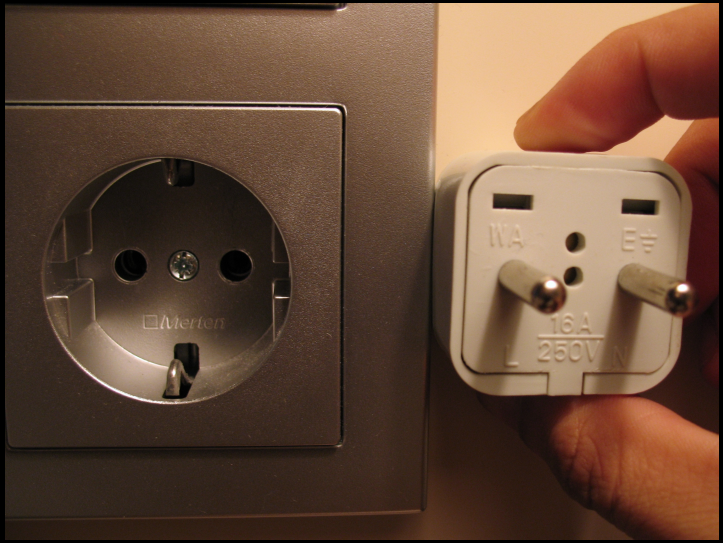


Image: Simon Law on Flickr, CC-BY-SA

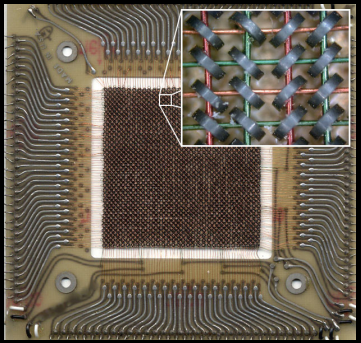
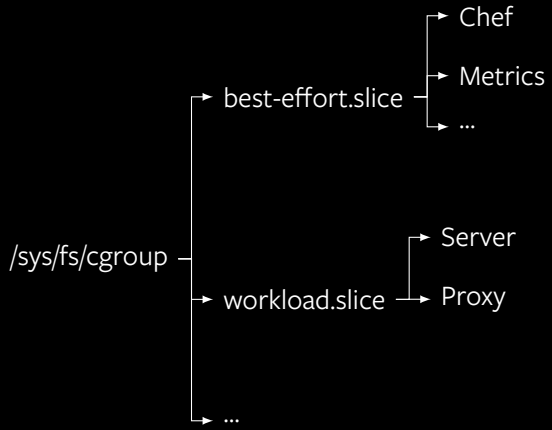
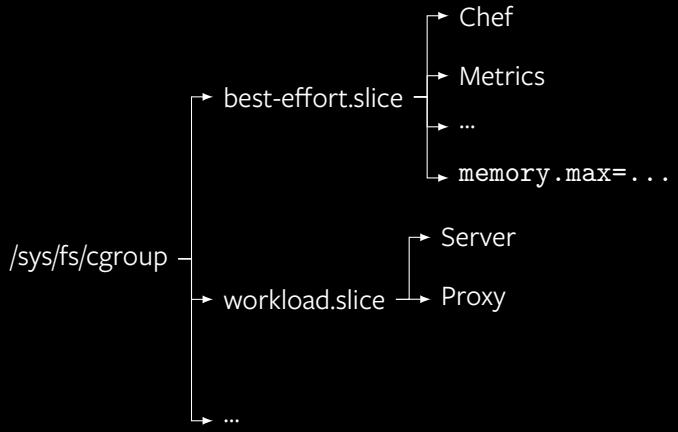


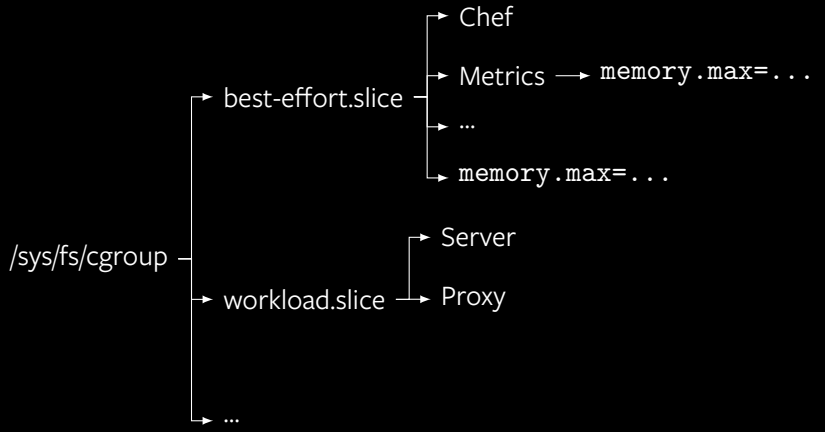
Image: Orion J on Wikimedia Commons, CC-BY

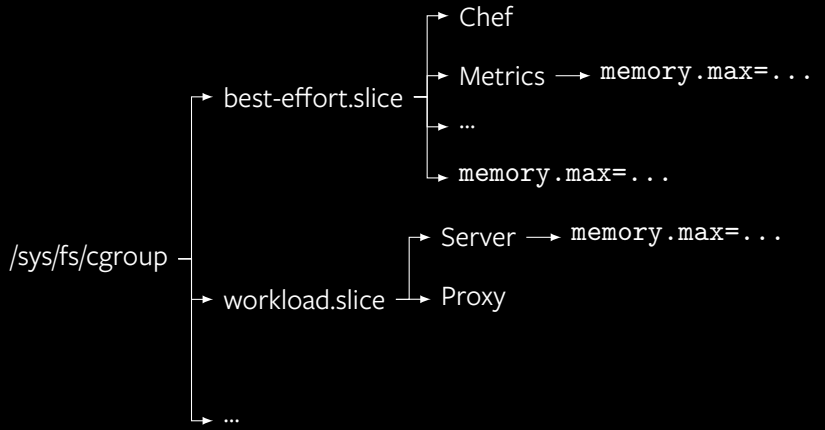
- Memory is divided in to multiple “types”: anon, cache, buffers, etc
- “Reclaimable” or “unreclaimable” is important, but not guaranteed
- RSS is kinda bullshit, sorry

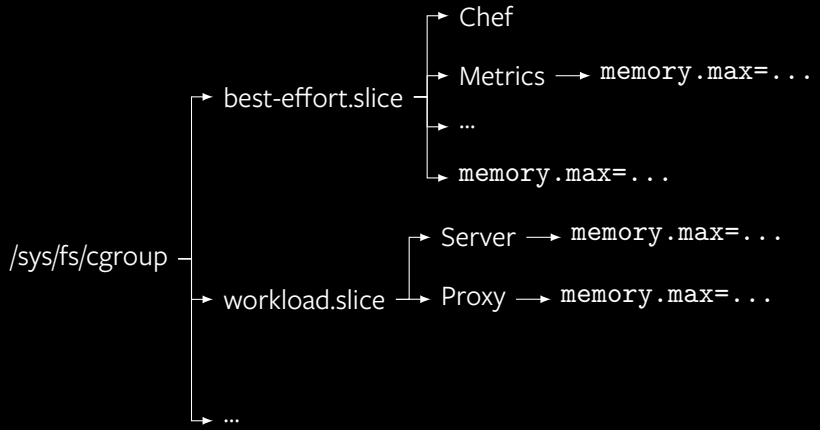
```
# cgroup v2
echo 1G > /sys/fs/cgroup/foo/memory.max
```

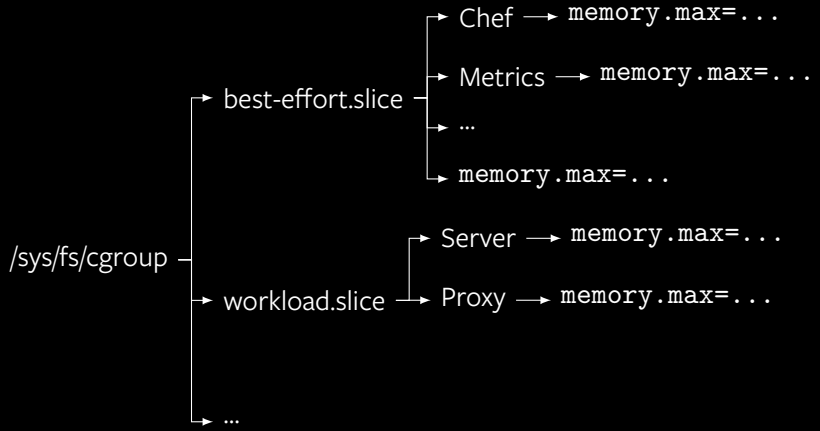

/sys/fs/cgroup
best-effort.slice
Chef
Metrics
...
workload.slice
Server
Proxy
...

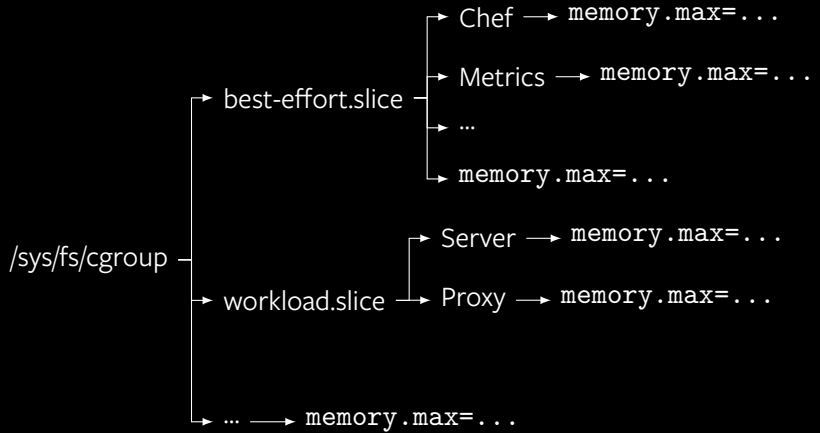

Chef
Metrics
best-effort.slice
...
memory.max=...
Server
/sys/fs/cgroup
workload.slice
Proxy
...

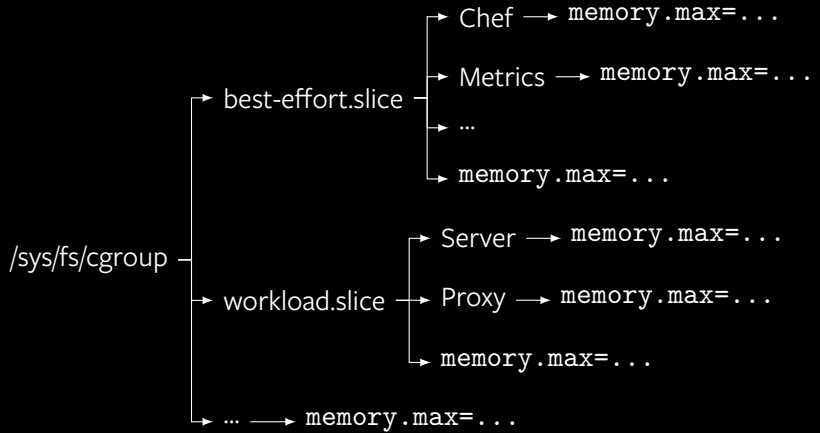

/sys/fs/cgroup
best-effort.slice
Chef
Metrics
memory.max=...
...
memory.max=...
workload.slice
Server
Proxy
...

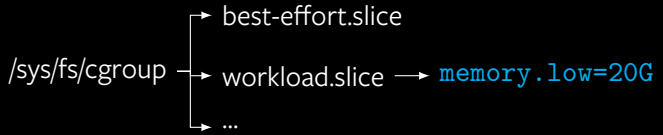

/sys/fs/cgroup
best-effort.slice
Chef
Metrics
memory.max=...
...
memory.max=...
workload.slice
Server
memory.max=...
Proxy
...

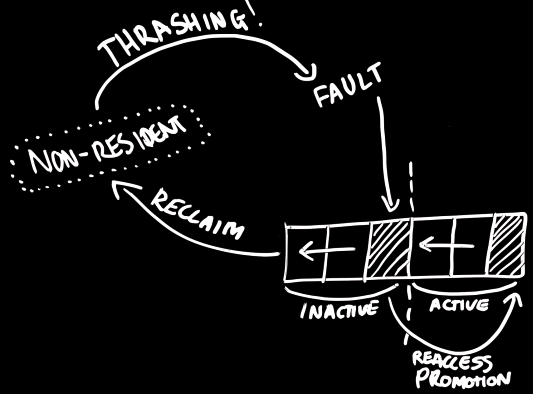

/sys/fs/cgroup
best-effort.slice
Chef
Metrics
memory.max=...
...
memory.max=...
workload.slice
Server
memory.max=...
Proxy
memory.max=...
...


/sys/fs/cgroup
best-effort.slice
Chef
memory.max=...
Metrics
memory.max=...
...
memory.max=...
workload.slice
Server
memory.max=...
Proxy
memory.max=...
...


/sys/fs/cgroup
best-effort.slice
Chef
memory.max=...
Metrics
memory.max=...
...
memory.max=...
workload.slice
Server
memory.max=...
Proxy
memory.max=...
...
memory.max=...


/sys/fs/cgroup
best-effort.slice
Chef
memory.max=...
Metrics
memory.max=...
...
memory.max=...
workload.slice
Server
memory.max=...
Proxy
memory.max=...
memory.max=...
...
memory.max=...


/sys/fs/cgroup
best-effort.slice
workload.slice
memory.low=20G
...



- memory.low and memory.min bias reclaim away from a cgroup
- Reclaim can still be triggered when protected on global memory shortage

```
% 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

How should we detect memory pressure?

How should we detect memory pressure?

- Free memory?

How should we detect memory pressure?

- Free memory?
- ...without caches and buffers?

How should we detect memory pressure?

- Free memory?
- ...without caches and buffers?
- Page scanning?

How should we detect memory pressure?

- Free memory?
- ...without caches and buffers?
- Page scanning?
- Something else?

psi

“If I had more of this resource, I could probably run *N*% faster”

- Find bottlenecks
- Detect workload health issues before they become severe
- Used for resource allocation, load shedding, pre-OOM detection

```
% cat /sys/fs/cgroup/system.slice/memory.pressure
some avg10=0.21 avg60=0.22 total=4760988587
full avg10=0.21 avg60=0.22 total=4681731696
```

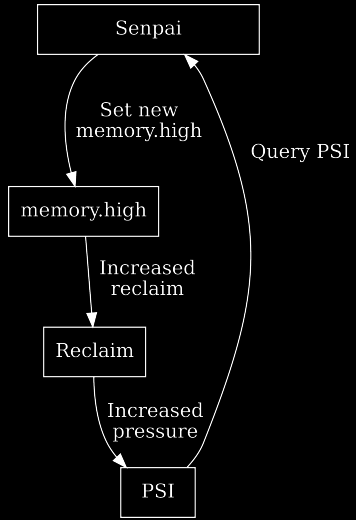
```
% time make -j4 -s
real    3m58.050s
user    13m33.735s
sys     1m30.130s

# 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
```

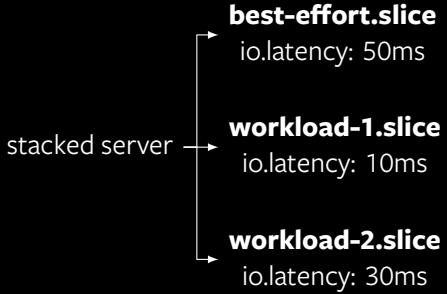


bit.ly/cgsenpai

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

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

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


stacked server
best-effort.slice
io.latency: 50ms
workload-1.slice
io.latency: 10ms
workload-2.slice
io.latency: 30ms

bit.ly/iocost & bit.ly/resctlbench

All the cool kids are using it

Control group 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™

Next-gen resource management

bit.ly/kdecgv2

Try it out:

`cgroup_no_v1=all` on kernel command line

Docs: `bit.ly/cgroupv2doc`

Whitepaper: `bit.ly/cgroupv2wp`


facebook