Cgroups - Control Groups

All cgroups are found under the ‘cgroup’ pseudo filesystem here:
/sys/fs/cgroup

```
$ mount |grep cgroup
cgroup2 on /sys/fs/cgroup type cgroup2
(rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursiveprot)
```

The modern variant’s called cgroups2 or cgroups v2.

Populated in the usual manner – as dirs and files...

```
$ ls /sys/fs/cgroup/
cgroup.controllers      cpu.stat              memory.reclaim
cgroup.max.depth        dev-hugepages.mount/  memory.stat
cgroup.max.descendants  dev-mqueue.mount/     misc.capacity
cgroup.pressure         init.scope/           misc.current
cgroup.procs            io.cost.model         proc-sys-fs-binfmt_misc.mount/
cgroup.stat             io.cost.qos           sys-fs-fuse-connections.mount/
cgroup.subtree_control  io.pressure           sys-kernel-config.mount/
cgroup.threads          io.prio.class         sys-kernel-debug.mount/
cpu.pressure            io.stat               sys-kernel-tracing.mount/
cpuset.cpus.effective   memory.numa_stat      system.slice/
cpuset.mems.effective   memory.pressure       user.slice/
$
```

https://utcc.utoronto.ca/~cks/space/blog/linux/CgroupV2FairShareScheduling
...
In cgroup v2, there is a single ('unified') hierarchy of cgroups and processes in them. However, a given spot in the hierarchy may not have all of the available resource controllers enabled in it, as covered in the "enabling and disabling" section of the documentation. Enabling a controller is potentially important because, as described there:

Enabling a controller in a cgroup indicates that the distribution of the target resource across its immediate children will be controlled. [...]

The inverse is true; if a controller is not enabled, the distribution of the resource to the cgroup's children is not being controlled.

The list of controllers that are being applied to immediate children is in cgroup.subtree_control. The presence of a particular controller in that causes settings files related to the controller to show up in the immediate children, which means that looking for those settings files in a child (such as 'user.slice/user-1000.slice') is a reliable indicator that the parent (ie, 'user.slice') has the controller enabled.

...

Control of CPU scheduling is handled by the cgroup v2 'cpu' controller. Fair share scheduling is handled through a weight-based CPU time distribution model (although you can also impose usage limits). When the CPU controller is enabled for a cgroup, a cpu.weight file appears in all children (with a default value of 100). When distributing CPU time to the children, all of their cpu.weight values are summed up and then each active child gets CPU in proportion to their weight relative to the total. This means that if all cpu.weight files have the same value, all children will get equal shares of the CPU time. The actual cpu.weight values only matter if they're different; if they're all the same, the value is arbitrary.

(All of this is assuming that the CPU is saturated. If not all of the CPU is being used, everyone gets as much of it as they want.)

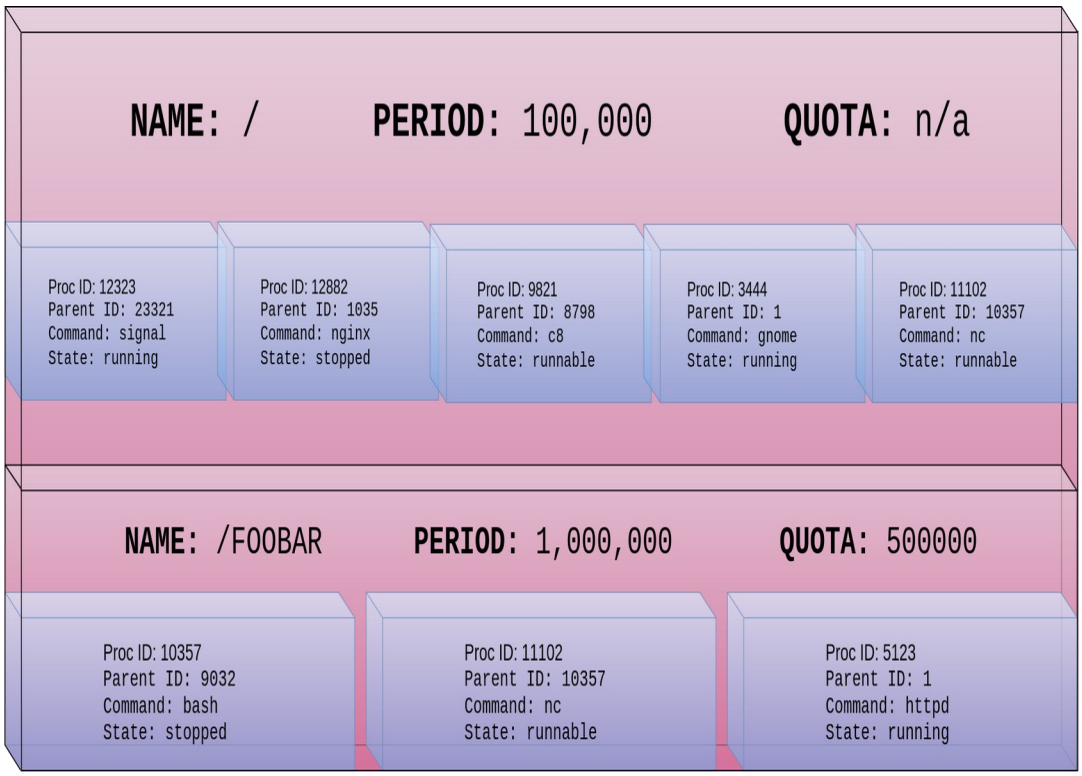
...

https://utcc.utoronto.ca/~cks/space/blog/linux/SystemdCgroupV2FairScheduling

Found on Fedora 38
/sys/fs/cgroup/user.slice/user-1000.slice/cpu.weight:500

so the user uid 1000 is getting more cpu than others...

https://www.grant.pizza/blog/understanding-cgroups/
...
This is best explained by example:



A CPU cgroup “FOOBAR”, a child of the root CPU cgroup

In the diagram above we see that there are three processes in a cgroup called /Foobar. There are also many processes in the / cgroup. As we see in that root cgroup, a quota of -1 is a special value to indicate there is an unlimited quota. In other words, no limit.

Now let’s think about the /Foobar cgroup. A period of 1000000 microseconds (or one second) has been specified. The quota of 500000 microseconds (or a half second) has also been set. Everytime a process in the cgroup spends a microsecond of time running on the CPU, the quota is decremented. Every process in the cgroup shares this quota. As an example let’s say all three processes run at the same time (each on their own core) starting at the beginning of a period. After around .17 of a second the processes in the cgroup will have spent their entire quota. At that point the scheduler will opt to keep all three of those processes paused until the period is over. At that point the quota is refreshed.

...

They are not the mechanism by which resources are limited but rather just a glorified way of collecting arguments for those resource limits. It’s up to the individual subsystems to read those arguments and take them into consideration. The same goes for every other cgroup implementation.

...

OFFICIAL DOCS!

https://docs.kernel.org/admin-guide/cgroup-v2.html

At least 1 cgroup on boot; the ‘root’ one, /

...

https://opensource.com/article/20/10/cgroups
good article.

https://www.redhat.com/sysadmin/cgroups-part-three
...
Remember that the CPUShares are based on the top-level cgroup, which is unconstrained by default. This means that should a process higher up in the tree demand CPUShares, the system will give that process priority. This can confuse people. It's vital to have a visual representation of the cgroup layout on a system to avoid confusion.
...

https://www.redhat.com/sysadmin/cgroups-part-four
...
The answer has to do with the way that systemd interprets nested cgroups. Children are declared in the following fashion: -.slice. Since systemd attempts to be helpful, if a parent does not exist, systemd creates it for you. If I had used underscores _ instead of dashes - the result would have been what you would have expected:

Control group /:
├─my_beautiful_slice.slice
│ └─cat.service
│ └─4123 /usr/bin/cat /dev/urandom

<<See
systemd.resource-control(5) >>

...

Instead of showing you the output from top, now is a good time to introduce you to systemd-cgtop. It works in a similar fashion to regular top except it gives you a breakdown per slice, and then again by services in each slice. This is very helpful in determining whether you are making good use of cgroups in general on your system. As seen below, systemd-cgtop shows both the aggregation for all services in a particular slice as part of the overall system and the resource utilization of each service in a slice:
...