Cgroups: The Next Generation

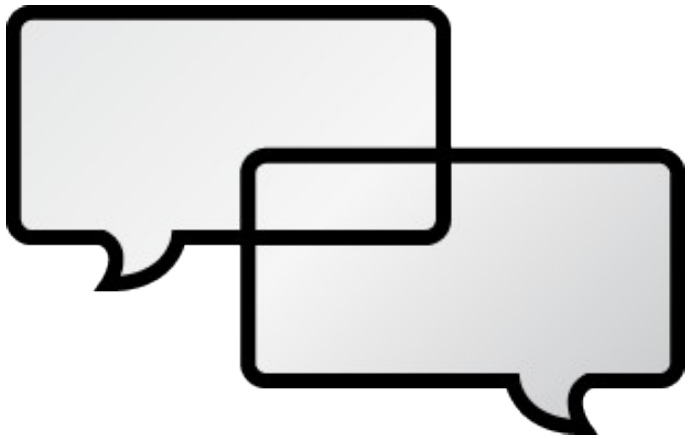
RHEL 8 & Cgroups v2

Marc Richter

Principal Technical Account Manager

Red Hat

What we'll be discussing today



Intros

Cgroup Basics

What's New in v2

Demo

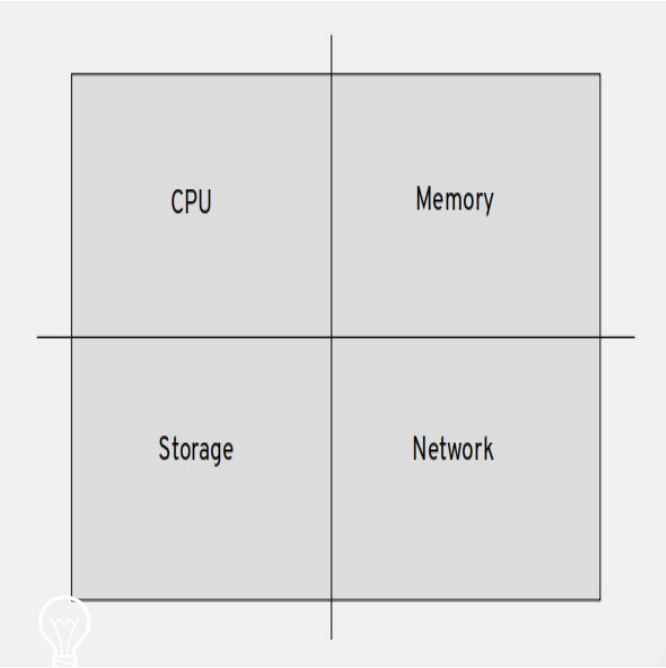
Next Steps

/whois unclemarc

Linux Nerd since 1998
RHCE, cause that’s fun
Red Hatter since 2015
    - Principal Technical Account Manager
Scout Leader since 2009
1 wife, 4 kids, 3 dogs = mild chaos

Red Hat

Why Cgroups?

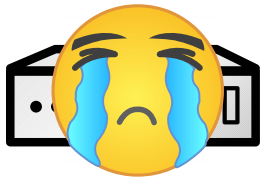


Moving Parts

Modern computers are pretty dang busy. Even so-called “serverless” applications are but one of many jobs running on a single piece of hardware. We need tools to manage the balance of all the critical resources on the system. Cgroups are an important tool for performance tuning.

Performance Tuning? Hard! Do not want!

A Real World Problem

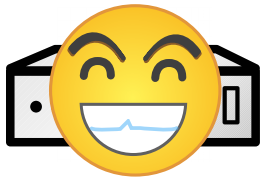


Database Server
Clustered database server, load would vary during the workday.

Mandatory Security Scanner
Periodic job to look for malware and other nastiness. Takes all the processor time it can get.

Cluster Agent
If on the same core as the scanner, would get starved for CPU during busy times. This would cause a fencing event

A Real World Problem. Fixed



Create a control group

System was RHEL 6, so we used the libcgroup tools to create a cgroup under the CPU controller.

Set CPU Quota for new cgroup

We set the maxium CPU allowed to 60% of a single core.

Scanning Agent

When launched, the agent gets placed into the new control group. It can NEVER exceed the CPU Quota

Profit!

The server stopped being fenced, as the cluster agent was never starved for CPU time.

“Tell me more about these wondrous cgroups…”

```
def parse_block_three(block, url)
  result = []
  case_array = block.split(",")
  index = 0

  puts block
  case_array.each do |caseid|
    if index == 0 then
        parsed = caseid.match(/(\w*) (\w*) (\w*)/)
        caseid = parsed[3]
    end
    result << { :text => caseid, :url => url + "#{caseid}" }
    index += 1
  end
  result
end

def parse_hub_block(block, url)

  #puts "Input: " +  block
  result = []
  case_array = block.split(",")
  index = 0
```

Kernel Based Controllers

Officially appeared in RHEL 6. Required manual configuration to enable and use.

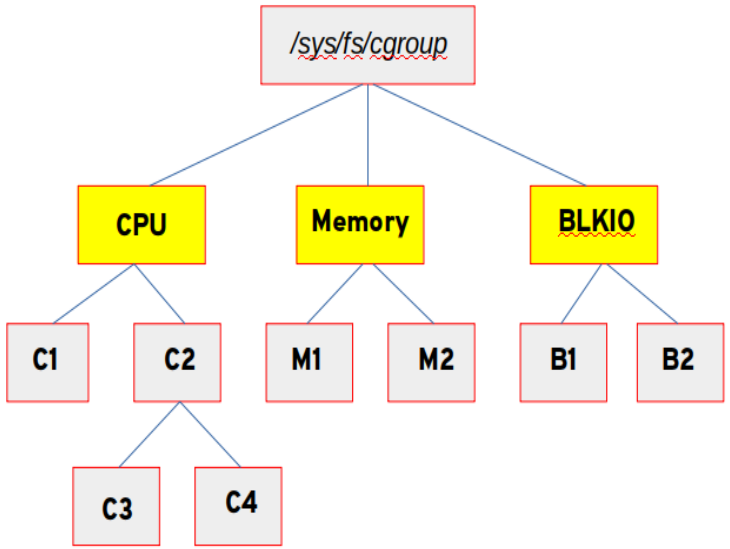
Core systemd component

RHEL 7 and 8 both use cgroups v1 (same concept as cgroups in RHEL 6) as their default in systemd. Not optional, required for proper system operation. Not all controllers used by systemd.

Required for containers

Cgroups are a foundational component for containers, along with kernel namespaces and SELinux.

Version 1



Default in all RHEL

It's the only version available in RHEL 6 and RHEL 7 and is the default in RHEL8

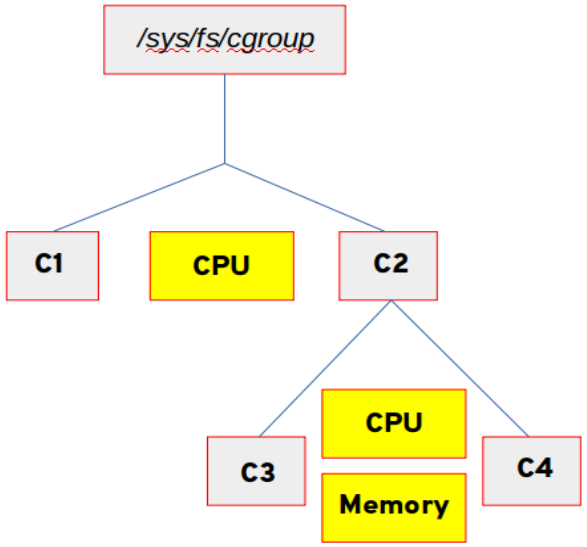
Controlled via a virtual filesystem

Mounted at /sys/fs/cgroup – this can be analyzed for current state and modified to change state. Many actions have commands or APIs rather than manually manipulating this filesystem

Cgroups are arranged under controllers

Each controller has a hierarchy under it. A process can end up existing in one or more cgroups at the same time. This can lead to some confusion

Version 2



Optional in RHEL 8

Can be enabled with a kernel boot option. Most commands are supported, some use cases are not yet in place.

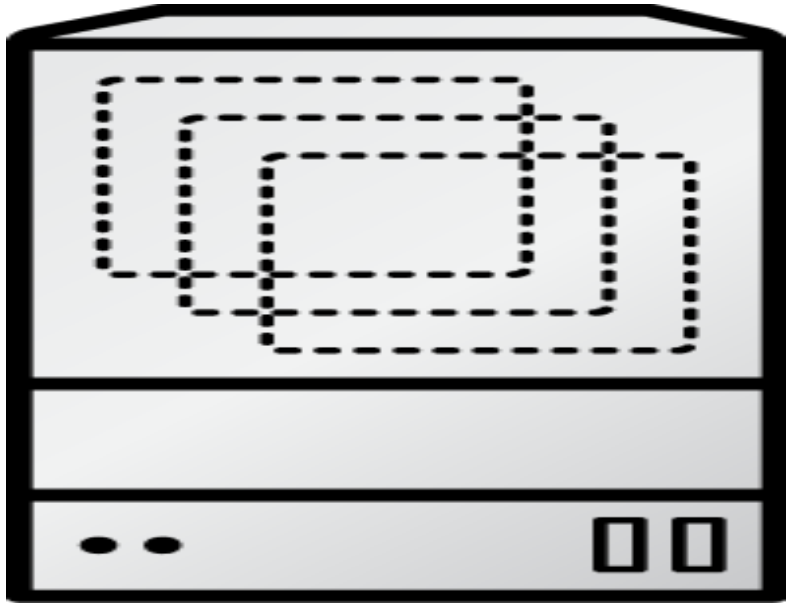
Controlled via a virtual filesystem

Mounted at /sys/fs/cgroup – this can be analyzed for current state and modified to change state. Many actions have commands or APIs rather than manually manipulating this filesystem

Single hierarchy

There is a single hierarchy for all cgroups. Controllers are enabled for sub trees in the hierarchy. A process can only exist in one cgroup at a time. This simplfies managing the processes a bit.

Version 2 Controllers in RHEL 8



Four currently supported

CPU
Memory
BLKIO
PIDs

Exploring the virtual filesystem

```
[root@roland ~]# cd /sys/fs/cgroup/
[root@roland cgroup]# ls -la
total 0
dr-xr-xr-x.  5 root root 0 Jun  5 18:34 .
drwxr-xr-x.  7 root root 0 Jun  5 18:34 ..
-r--r--r--.  1 root root 0 Jun  5 18:36 cgroup.controllers
-rw-r--r--.  1 root root 0 Jun  5 18:36 cgroup.max.depth
-rw-r--r--.  1 root root 0 Jun  5 18:36 cgroup.max.descendants
-rw-r--r--.  1 root root 0 Jun  5 18:36 cgroup.procs
-r--r--r--.  1 root root 0 Jun  5 18:36 cgroup.stat
-rw-r--r--.  1 root root 0 Jun  5 18:36 cgroup.subtree_control
-rw-r--r--.  1 root root 0 Jun  5 18:36 cgroup.threads
drwxr-xr-x.  2 root root 0 Jun  5 18:34 init.scope
drwxr-xr-x. 37 root root 0 Jun  5 18:34 system.slice
drwxr-xr-x.  3 root root 0 Jun 13 08:37 user.slice
[root@roland cgroup]#
```

Exploring the virtual filesystem

```
[root@roland system.slice]# pwd
/sys/fs/cgroup/system.slice
[root@roland system.slice]# ls
 atd.service                    dev-mqueue.mount          polkit.service
 auditd.service                 firewalld.service         puppet.service
 boot.mount                     io.bfq.weight             qemu-guest-agent.service
 cgroup.controllers             io.max                    rhsmcertd.service
 cgroup.events                  io.stat                   rngd.service
 cgroup.max.depth               irqbalance.service        rsyslog.service
 cgroup.max.descendants         libstoragemgmt.service    smartd.service
 cgroup.procs                   mcelog.service            sshd.service
 cgroup.stat                    memory.current            sssd.service
 cgroup.subtree_control         memory.events             sys-kernel-config.mount
 cgroup.threads                 memory.high               sys-kernel-debug.mount
 cgroup.type                    memory.low                systemd-journald.service
 chronyd.service                memory.max                systemd-logind.service
 cockpit.socket                 memory.min                systemd-udevd.service
 cpu.max                        memory.stat               system-getty.slice
 cpu.stat                       memory.swap.current      'system-lvm2\x2dpvscan.slice'
 cpu.weight                     memory.swap.events       'system-sshd\x2dkeygen.slice'
 cpu.weight.nice                memory.swap.max          'system-systemd\x2dhibernate\x2dresume.slice'
 crond.service                  NetworkManager.service   'system-user\x2druntime\x2ddir.slice'
 dbus.service                   pids.current              tuned.service
 dev-hugepages.mount            pids.events
'dev-mapper-rhel\x2dswap.swap'  pids.max
```

Exploring the virtual filesystem

```
[root@roland system.slice]# pwd
/sys/fs/cgroup/system.slice
[root@roland system.slice]# cat cgroup.subtree_control
memory pids
[root@roland system.slice]#
```

Exploring the virtual filesystem

```
[root@roland system.slice]# cd sshd.service/
[root@roland sshd.service]# ls
cgroup.controllers      cgroup.procs            cgroup.type     memory.high  memory.stat          pids.current
cgroup.events           cgroup.stat             cpu.stat        memory.low   memory.swap.current  pids.events
cgroup.max.depth        cgroup.subtree_control  memory.current  memory.max   memory.swap.events   pids.max
cgroup.max.descendants  cgroup.threads          memory.events   memory.min   memory.swap.max
[root@roland sshd.service]# cat cgroup.procs
727
[root@roland sshd.service]# ps aux | grep 727
root       727  0.0  0.9  92248  7656 ?        Ss   Jun05   0:00 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chac
ha20-poly1305@openssh.com,aes256-ctr,aes256-cbc,aes128-gcm@openssh.com,aes128-ctr,aes128-cbc -oMACs=hmac-sha2-256-etm@op
enssh.com,hmac-sha1-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha2-256,hmac-sha1,umac-
128@openssh.com,hmac-sha2-512 -oGSSAPIKexAlgorithms=gss-gex-sha1-,gss-group14-sha1- -oKexAlgorithms=curve25519-sha256@li
```

Demo

Using systemctl and cgroups v2

In this demonstration, we'll use the systemctl command to change the CPU quota of user "mrichter" on the fly. We'll see what changes happen in the virtual filesystem. You'll also meet Mr. Scope, an interesting character.

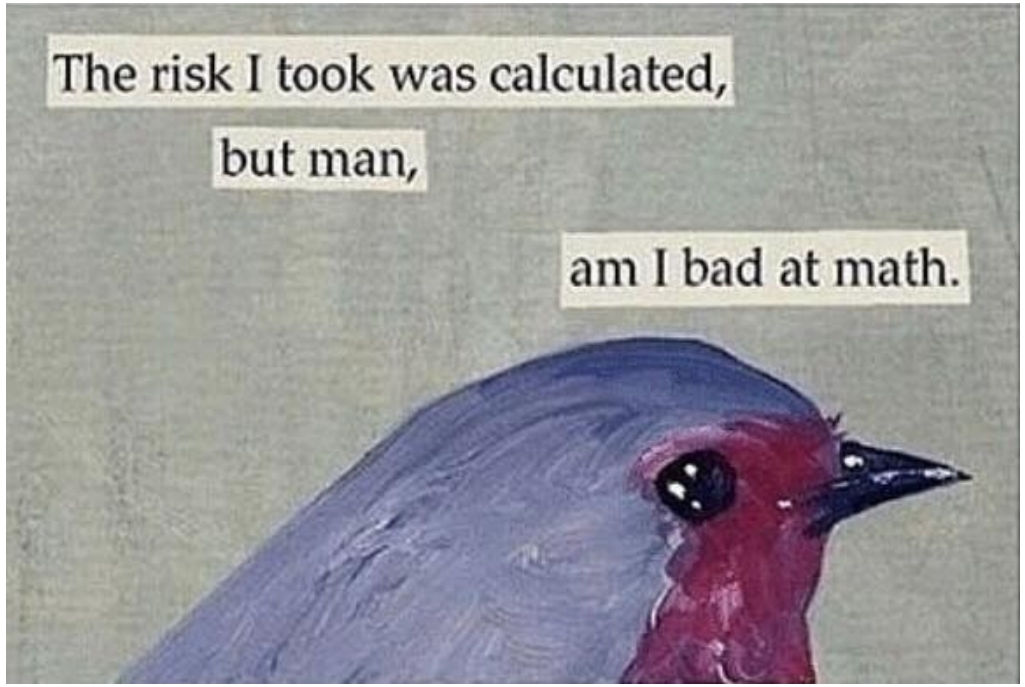
If Marc Forgot to Mention During the Demo...

```
[root@roland user-1000.slice.d]# pwd
/etc/systemd/system.control/user-1000.slice.d
[root@roland user-1000.slice.d]# ls
50-CPUQuota.conf
[root@roland user-1000.slice.d]# cat 50-CPUQuota.conf
# This is a drop-in unit file extension, created via "systemctl set-property"
# or an equivalent operation. Do not edit.
[Slice]
CPUQuota=100%
[root@roland user-1000.slice.d]#
```


Persistence

Setting a property writes it to the /etc/systemd/system.control directory. This overrides drop-ins in /etc/systemd/system/

So unclemarc, should we be using cgroups v2?



Not fully implemented for all use cases

- libvirt
- runc
- Kubernetes

Version 1 remains the default

Support for v1 will remain for lifespan of RHEL 8 and will always be the default

libcgroup tools ARE most likely going away

It is time to move off of tooling that relies on the old-school RHEL 6 flavored libcgroup packages

Next Steps

Cgroups Blog Series
https://www.redhat.com/en/blog/authors/marc-richter

RHEL 8 System Documentation
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/

Cgroups Kernel Doc
https://www.kernel.org/doc/Documentation/cgroup-v2.txt

Questions?

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat