

# An introduction to cgroups and cgroupspy

```
tags = ['python', 'docker', 'coreos', 'systemd']
```



## About me

- Entrepreneur
- Geek
- VP Biz Dev @ CloudSigma

## Contact info

- Email: [viktor@cloudsigma.com](mailto:viktor@cloudsigma.com)
- WWW: <http://vpetersson.com>
- Twitter: [@vpetersson](https://twitter.com/vpetersson)
- LinkedIn: <http://www.linkedin.com/in/vpetersson>

## About CloudSigma

- Public Infrastructure-as-a-Service (IaaS)
- PoPs in Europe and North America
- Support (almost) all operating systems
- Virtual data center
- Trusted by CERN, ESA and many more

## Talk outline

- Introduction to cgroups
- Using cgroups
  - Examples
- Cgroup tools
  - Filesystem
  - libcgroup
  - cgroupspy
  - systemd
  - Docker

# 1. Introduction

## What are cgroups?

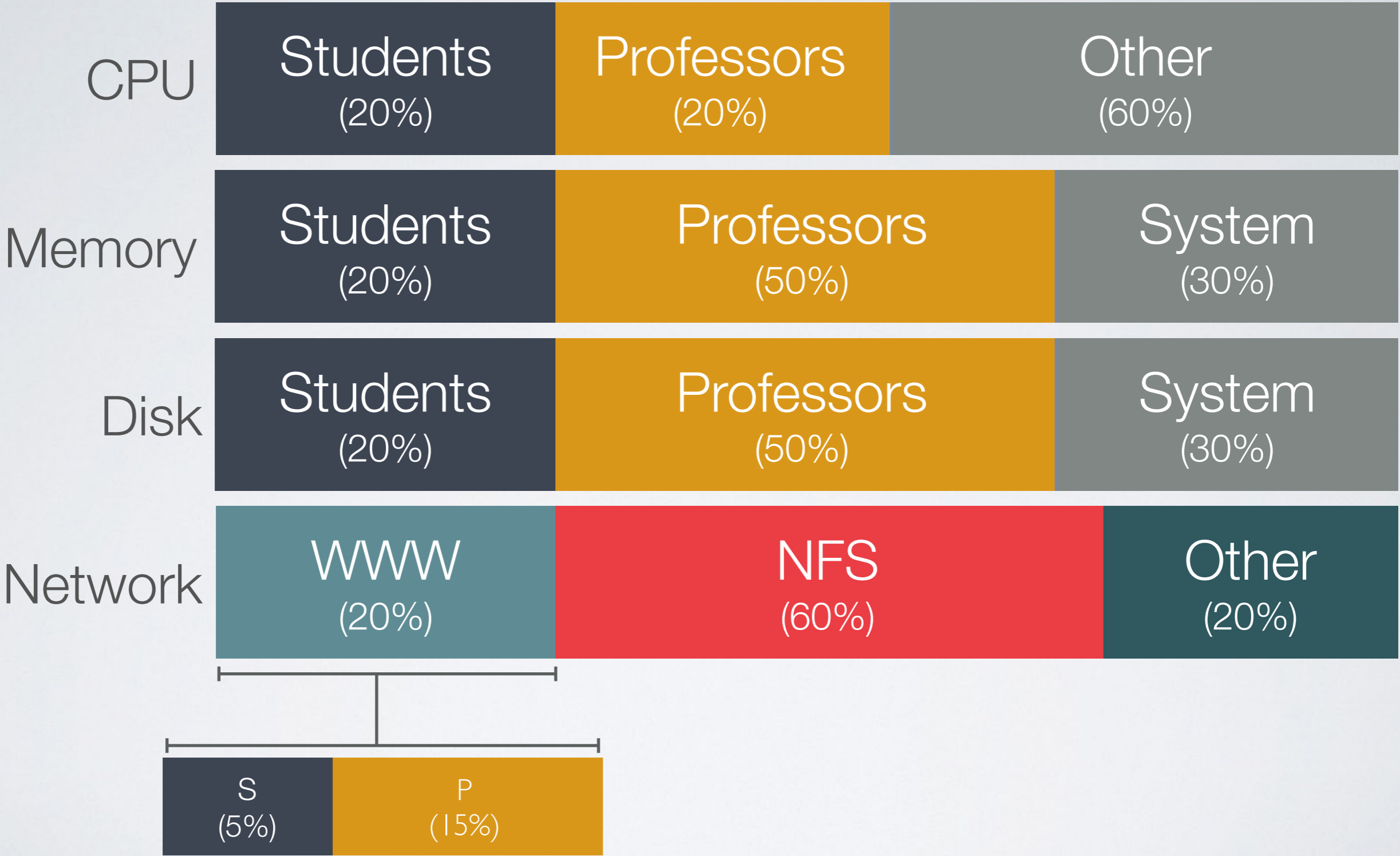
- Control groups
- Resource accounting (with hierarchy)
- Much more sophisticated than `ulimit`
- A file system

## What can you do with cgroups?

- Limit and prioritize
  - CPU consumption
  - Memory consumption
  - Disk I/O consumption
  - Network consumption
  - Device limitations
- Classify network packets
- Freeze processes
- Set up resource accounting

# Example (from the Kernel docs)

1. Introduction





## Terminology

- Resource class or Controller
- Group or Slice (in systemd)
- CPU Schedulers
  - Completely Fair Scheduler (CFS)
  - Real-Time scheduler (RT)

## Resource classes

- **Block IO** (blkio)
- **CPU Set** (cpuset)
- CPU Accounting (cpuacct)
- **CPU** (cpu)
- Devices (devices)
- Freezer (freezer)
- **Memory** (memory)
- Network Classifier (net\_cls)
- Network Priority (net\_prio)

# 1. Introduction

## Universal elements

- tasks
- notify\_on\_release
- release\_agent

# 1. Introduction

Distro	Cgroups	Systemd
CentOS/RHEL	Yes	Yes
CoreOS	Yes	Yes
Debian	Yes	Yes
Fedora	Yes	Yes
Ubuntu	Yes	Optional

## Zero to cgroups on Ubuntu 14.04

```
$ apt-get install -y cgroup-lite  
$ mkdir /sys/fs/cgroup/cpuset/test  
$ echo 0 > /sys/fs/cgroup/cpuset/test/cpuset.cpus  
$ echo $$ > /sys/fs/cgroup/cpuset/test/tasks
```

## 2. Using cgroups

## 2.1 CPU Resources

## 2.1 CPU Resources

### CPU resources

- cpu
- cpuset



## cpu

cpu.stat

cpu.cfs\_period\_us

cpu.cfs\_quota\_us

### **cpu.shares**

cgroup.sane\_behavior

cgroup.clone\_children

cgroup.event\_control

cgroup.procs

## cpuset

cpuset.memory\_pressure\_enabled

cpuset.memory\_spread\_slab

cpuset.memory\_spread\_page

cpuset.memory\_pressure

cpuset.memory\_migrate

cpuset.sched\_relax\_domain\_level

cpuset.sched\_load\_balance

cpuset.mem\_hardwall

cpuset.mem\_exclusive

cpuset.cpu\_exclusive

### **cpuset.mems**

### **cpuset.cpus**

cgroup.sane\_behavior

cgroup.clone\_children

cgroup.event\_control

cgroup.procs

### Limit a process to a specific CPU core

```
# Create a group
```

```
$ cd /sys/fs/cgroup
```

```
$ mkdir -p cpuset/group1
```

```
# Limit 'group1' to core 0 and enroll the current shell
```

```
$ echo 0 > cpuset/group1/cpuset.cpus
```

```
$ echo $$ > cpuset/group1/tasks
```

## Limit a process to a specific CPU core

# Before

```
$ cat /proc/$$/status | grep '_allowed'
```

```
Cpus_allowed:    3
```

```
Cpus_allowed_list: 0-1
```

```
Mems_allowed:   000000000,000000001
```

```
Mems_allowed_list:  0
```

# After

```
$ cat /proc/$$/status | grep '_allowed'
```

```
Cpus_allowed:    1
```

```
Cpus_allowed_list: 0
```

```
Mems_allowed:   000000000,000000001
```

```
Mems_allowed_list:  0
```

# Allocate “CPU Shares” across two groups

```
# Create two groups
```

```
$ cd /sys/fs/cgroup
```

```
$ mkdir -p cpu/group1 cpu/group2
```

```
# Allocate CPU shares
```

```
$ echo 250 > cpu/group1/cpu.shares
```

```
$ echo 750 > cpu/group2/cpu.shares
```

```
# Fire off the workload
```

```
$ burnP6 --group1 & echo $! > cpu/group1/tasks
```

```
$ burnP6 --group2 & echo $! > cpu/group2/tasks
```

2.1 CPU Resources

```

2. vagrant@vagrant-ubuntu-trusty-64: ~ (ssh)

CPU[|||||||||||||||||||||||||||||||||100.0%]   Tasks: 54, 17 thr; 3 running
Mem[||||| ||||| ||||| ||||| ||||| |||||]      174/2001MB   Load average: 0.05 0.07 0.10
Swp[|||||]                                     0/0MB       Uptime: 01:40:35

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
 2792 root    20   0   148    4     0  R  80.7  0.0   0:03.75 burnP6 --group1
 2793 root    20   0   148    4     0  R  20.0  0.0   0:00.93 burnP6 --group2
 2789 vagrant 20   0 24456 2004 1464  R   0.7  0.1   0:00.07 htop
 1716 vagrant 20   0 24516 1376  724  S   0.0  0.1   0:18.55 htop
 2243 vagrant 20   0 24516 2132 1472  S   0.0  0.1   0:08.42 htop
 2364 root     20   0 21860 4472 1888  S   0.0  0.2   0:00.24 -su
 1663 vagrant 20   0 25692 2592  820  S   0.0  0.1   0:06.24 tmux
 1201 root     20   0 227M   376   44  S   0.0  0.0   0:02.12 /usr/sbin/VBoxSer
 2089 vagrant 20   0 105M  2384 1376  S   0.0  0.1   0:01.33 sshd: vagrant@pts
 1208 root     20   0 227M   376   44  S   0.0  0.0   0:00.90 /usr/sbin/VBoxSer
    1 root     20   0 33648 1676  188  S   0.0  0.1   0:00.90 /sbin/init
  282 root     20   0 23660  808   568  S   0.0  0.0   0:00.03 /sbin/cgmanager -
  446 root     20   0 19472  300    4  S   0.0  0.0   0:00.06 upstart-udev-brid
  451 root     20   0 51440  704    4  S   0.0  0.0   0:00.03 /lib/systemd/syst
  529 root     20   0 10220 2292    0  S   0.0  0.1   0:00.00 dhclient -1 -v -p
  611 root     20   0 23416  312    4  S   0.0  0.0   0:00.02 rpcbind
  683 statd    20   0 21540  464    4  S   0.0  0.0   0:00.00 rpc.statd -L
  688 root     20   0 15388  380    0  S   0.0  0.0   0:00.03 upstart-socket-br

F1Help  F2Setup  F3Search  F4Filter  F5Tree  F6SortBy  F7Nice -  F8Nice +  F9Kill  F10Quit

```

'cpu.shares' in action

## 2.2 Memory Resources

## Memory

memory.kmem.tcp.max\_usage\_in\_bytes    memory.force\_empty  
memory.kmem.tcp.failcnt    memory.stat  
memory.kmem.tcp.usage\_in\_bytes    memory.failcnt  
memory.kmem.tcp.limit\_in\_bytes    memory.soft\_limit\_in\_bytes  
memory.kmem.slabinfo    **memory.limit\_in\_bytes**  
memory.kmem.max\_usage\_in\_bytes    memory.max\_usage\_in\_bytes  
memory.kmem.failcnt    memory.usage\_in\_bytes  
memory.kmem.usage\_in\_bytes    cgroup.sane\_behavior  
memory.kmem.limit\_in\_bytes    cgroup.clone\_children  
memory.numa\_stat    cgroup.event\_control  
memory.pressure\_level    cgroup.procs  
**memory.oom\_control**    memory.move\_charge\_at\_immigrate  
memory.swappiness    memory.use\_hierarchy

# Setting up memory policies

```
# Create a group
```

```
$ cd /sys/fs/cgroup
```

```
$ mkdir -p memory/group1
```

```
# Set a memory limit of 150M
```

```
$ echo 150M > memory/group1/memory.limit_in_bytes
```

```
# Add shell to group
```

```
$ echo $$ > memory/group1/tasks
```

```
# Fire off a memory eating task
```

```
$ ./memhog
```



# 2.1 Memory Resources

```
2. root@vagrant-ubuntu-trusty-64: /sys/fs/cgroup (ssh)

CPU[|||| 3.3%] Tasks: 59, 17 thr; 1 running
Mem[||||| 169/2001MB] Load average: 0.26 0.13 0.08
Swp[ 0/0MB] Uptime: 02:54:25
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
3177	root	20	0	24468	1148	592	R	0.7	0.1	0:00.23	htop
2243	vagrant	20	0	24516	1268	608	S	0.7	0.1	0:24.78	htop
770	messagebu	20	0	39224	572	172	S	0.7	0.0	0:00.50	dbus-daemon --system --fork
1207	root	20	0	227M	332	0	S	0.7	0.0	0:00.34	/usr/sbin/VBoxService
1716	vagrant	20	0	24516	1264	612	S	0.0	0.1	0:34.98	htop
1663	vagrant	20	0	25692	2228	456	S	0.0	0.1	0:11.49	tmux
2089	vagrant	20	0	105M	1500	488	S	0.0	0.1	0:02.47	sshd: vagrant@pts/0
909	syslog	20	0	251M	744	0	S	0.0	0.0	0:00.01	rsyslogd
893	syslog	20	0	251M	744	0	S	0.0	0.0	0:00.03	rsyslogd
3162	root	20	0	21836	3108	552	S	0.0	0.2	0:00.03	-bash

```
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice - F8Nice + F9Kill F10Quit

root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# mkdir -p memory/group1
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# echo 150M > memory/group1/memory.limit_in_bytes
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# echo $$ > blkio/group1/tasks
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# /vagrant/bin/memhog
Killed
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup#
```

[0] 0:~\* "vagrant-ubuntu-trusty-" 12:58 04-Apr-15

*'memory.limit\_in\_bytes' in action*

## 2.3 Block I/O Resources

## Block IO

blkio.io\_queued\_recursive  
blkio.io\_merged\_recursive  
blkio.io\_wait\_time\_recursive  
blkio.io\_service\_time\_recursive  
blkio.io\_serviced\_recursive  
blkio.io\_service\_bytes\_recursive  
blkio.sectors\_recursive  
blkio.time\_recursive  
blkio.io\_queued  
blkio.io\_merged  
blkio.io\_wait\_time  
blkio.io\_service\_time  
blkio.io\_serviced  
blkio.io\_service\_bytes  
blkio.sectors

blkio.time  
blkio.leaf\_weight  
blkio.leaf\_weight\_device  
**blkio.weight**  
blkio.weight\_device  
blkio.throttle.io\_serviced  
blkio.throttle.io\_service\_bytes  
**blkio.throttle.write\_iops\_device**  
**blkio.throttle.read\_iops\_device**  
**blkio.throttle.write\_bps\_device**  
**blkio.throttle.read\_bps\_device**  
blkio.reset\_stats  
cgroup.sane\_behavior  
cgroup.clone\_children  
cgroup.event\_control

# Setting up I/O policies

```
# Find the device
```

```
$ lsblk
```

```
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda   8:0    0 40G  0 disk
└─sda1 8:1    0 40G  0 part /
```

```
# Create the groups
```

```
$ cd /sys/fs/cgroup
```

```
$ mkdir blkio/group1 blkio/group2
```

## Setting up I/O policies

```
# Group 1 and shell 1
```

```
$ echo "8:0 10485760" > blkio/group1/blkio.throttle.write_bps_device
```

```
$ echo $$ > blkio/group1/tasks
```

```
$ dd if=/dev/zero of=/tmp/writetest bs=64k count=3200 conv=fdatasync && \  
    rm /tmp/writetest
```

```
# Group 2 and shell 2
```

```
$ echo "8:0 20971520" > blkio/group1/blkio.throttle.write_bps_device
```

```
$ echo $$ > blkio/group2/tasks
```

```
$ dd if=/dev/zero of=/tmp/writetest bs=64k count=3200 conv=fdatasync && \  
    rm /tmp/writetest
```

## 2.3 Block I/O Resources

```
2. root@vagrant-ubuntu-trusty-64: /sys/fs/cgroup (ssh)
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# echo $$ > blkio/group1/tasks
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# echo "8:0 10485760" > blkio/group1/blkio.throttle.write_bps_device
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# dd if=/dev/zero of=/tmp/writetest bs=64k count=3200 conv=fdatasync && rm /tmp/writetest
3200+0 records in
3200+0 records out
209715200 bytes (210 MB) copied, 14.8784 s, 14.1 MB/s
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup#

root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# echo $$ > blkio/group2/tasks
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# echo "8:0 20971520" > blkio/group2/blkio.throttle.write_bps_device
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# dd if=/dev/zero of=/tmp/writetest bs=64k count=3200 conv=fdatasync && rm /tmp/writetest
3200+0 records in
3200+0 records out
209715200 bytes (210 MB) copied, 7.44203 s, 28.2 MB/s
root@vagrant-ubuntu-trusty-64:/sys/fs/cgroup# █

[0] 0:bash* "vagrant-ubuntu-trusty-" 12:53 04-Apr-15
```

*'blkio.throttle.write\_bps\_device' in action*

# 3. Cgroup tools

## Overview

- Filesystem
- libcggroup
- cgroupspy
- systemd
- Docker



## Using the filesystem

```
$ cd /sys/fs/cgroup
```

```
# Create a CPU group
```

```
$ mkdir -p cpu/group1
```

```
# Set a CPU Share
```

```
$ echo 250 > cpu/group1/cpu.shares
```

```
# Enroll [PID] in 'group1'
```

```
$ echo [PID] > cpu/group1/tasks
```

## Using libcgroup

# On Debian and Ubuntu

```
$ apt-get install -y cgroup-bin
```

# Create a group

```
$ cgcreate -g cpu:foobar
```

# Set values

```
$ cgset -r cpu.shares=6 foobar
```

# Run a command

```
$ cgexec -g cpu:foobar bash
```

```
^D
```

# Delete group

```
$ cgdelete cpu:foobar
```

## Cgroupspy

- Python wrapper for cgroups
- Integration with libvirt for interacting with VMs
- Developed by and used at CloudSigma

# Getting started with cgroupspy

```
$ pip install cgroupspy
```

```
$ python
```

```
>>> from cgroupspy import trees
```

```
>>> t = trees.Tree()
```

```
>>> cset = t.get_node_by_path('/cpuset/')
```

```
>>> cset.controller.cpus
```

```
set([0, 1])
```

```
>>> test = cset.create_cgroup('test')
```

```
>>> test.controller.cpus
```

```
set([0, 1])
```

```
>>> test.controller.cpus = [1]
```

```
>>> test.controller.cpus
```

```
set([1])
```

```
>>> cset.delete_cgroup('test')
```

## Controlling VMs with cgroupspy

```
>>> from cgroupspy.trees import VMTree
>>> vmt = VMTree()
>>> print vmt.vms
{u'1ce10f47-fb4e-4b6a-8ee6-ba34940cdda7.libvirt-qemu': <NodeVM 1ce10f47-
fb4e-4b6a-8ee6-ba34940cdda7.libvirt-qemu>,
 u'3d5013b9-93ed-4ef1-b518-a2cea43f69ad.libvirt-qemu': <NodeVM
3d5013b9-93ed-4ef1-b518-a2cea43f69ad.libvirt-qemu>,
}
>>> vm = vmt.get_vm_node("1ce10f47-fb4e-4b6a-8ee6-ba34940cdda7")

>>> print vm.cpuset.cpus
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

>>> print vm.memory.limit_in_bytes
25603080192
```

## Controlling VMs with cgroupspy (cont'd)

```
>>> print vm.children  
[<NodeControlGroup vcpu1>,  
 <NodeControlGroup vcpu0>,  
 <NodeControlGroup emulator>]
```

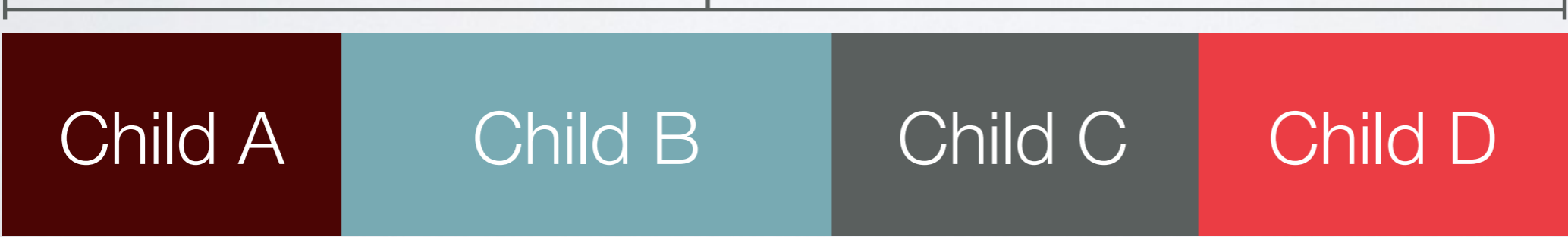
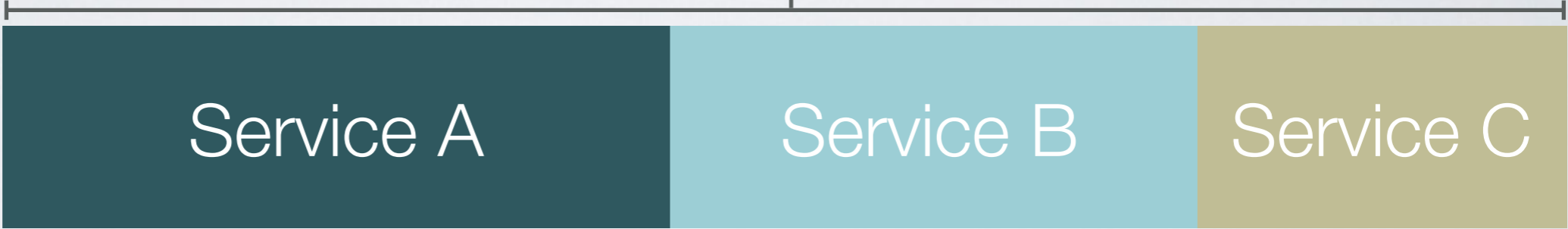
```
>>> print vm.path  
/machine/grey/1ce10f47-fb4e-4b6a-8ee6-ba34940cdda7.libvirt-qemu
```

```
>>> vcpu1 = vm.children[0]  
>>> print vcpu1.cpuset.cpus  
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
```

```
>>> vcpu1.cpuset.cpus = {1,2,3}  
>>> print vcpu1.cpuset.cpus  
{1, 2, 3}
```

## Systemd and cgroups

- Resource control in unit files
- Pre-configured slices
  - system
  - machine
  - user



3.4 Systemd



## Slices on CoreOS

```
$ cat cpu/system.slice/system-apache.slice/tasks  
730  
733  
734  
735  
736  
737
```

# Unit file for locksmithd on CoreOS

[Unit]

Description=Cluster reboot manager

Requires=update-engine.service

After=update-engine.service

ConditionVirtualization=!container

ConditionPathExists=!/usr/.nouupdate

[Service]

**CPUShares=16**

**MemoryLimit=32M**

PrivateDevices=true

EnvironmentFile=-/usr/share/coreos/update.conf

EnvironmentFile=-/etc/coreos/update.conf

ExecStart=/usr/lib/locksmith/locksmithd

Restart=always

RestartSec=10s

[Install]

WantedBy=multi-user.target

## Docker and cgroups

- Based on LXC
- Built-in support for cgroups via LXC
  - LXC driver must be activated

## Notes for Ubuntu 14.04

```
$ apt-get install -y lxc
```

```
$ echo 'DOCKER_OPTS="--exec-driver=lxc" \  
  >> /etc/default/docker
```

```
$ service docker restart
```

## Using cgroups in Docker

```
$ docker run -d --name='low_prio' \  
  --lxc-conf="lxc.cgroup.cpu.shares=250" \  
  --lxc-conf="lxc.cgroup.cpuset.cpus=0" \  
  busybox md5sum /dev/urandom
```

```
$ docker run -d --name='high_prio' \  
  --lxc-conf="lxc.cgroup.cpu.shares=750" \  
  --lxc-conf="lxc.cgroup.cpuset.cpus=0" \  
  busybox md5sum /dev/urandom
```

3.5 Docker

```

2. vagrant@vagrant-ubuntu-trusty-64: ~ (ssh)

1 [|||||||||||||||||||||||||||100.0%] Tasks: 48, 25 thr; 3 running
2 [|                               1.3%] Load average: 1.20 0.42 0.25
Mem[|||||                         193/2001MB] Uptime: 05:51:59
Swp[                               0/0MB]

 PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
11682 root        20   0  3156    72    36  R  75.3  0.0   0:38.38 md5sum /dev/urand
11601 root        20   0  3156    72    36  R  24.0  0.0   0:21.55 md5sum /dev/urand
 7894 vagrant    20   0 24580  2212 1472  S   1.3  0.1   0:03.08 htop
  813 messagebu  20   0 39212  1516 1056  S   0.7  0.1   0:02.18 dbus-daemon --sys
1220 root        20   0  227M  1092   764  S   0.7  0.1   0:10.25 /usr/sbin/VBoxSer
6868 vagrant    20   0  105M  2368 1356  S   0.0  0.1   0:02.75 sshd: vagrant@pts
11900 vagrant    20   0 24488  1996 1456  R   0.0  0.1   0:00.02 htop
   1 root        20   0 33640  3000 1504  S   0.0  0.1   0:02.06 /sbin/init
 300 root        20   0 23660  1472 1216  S   0.0  0.1   0:00.59 /sbin/cgmanager -
 458 root        20   0 19472   656  468  S   0.0  0.0   0:00.37 upstart-udev-brid
 462 root        20   0 51212  1544 1008  S   0.0  0.1   0:00.14 /lib/systemd/syst
 593 root        20   0 10220  2404  116  S   0.0  0.1   0:00.00 dhclient -1 -v -p
 649 root        20   0 23416  1104  800  S   0.0  0.1   0:00.09 rpcbind
 691 statd      20   0 21540  1376  920  S   0.0  0.1   0:00.00 rpc.statd -L
 705 root        20   0 15388   808  416  S   0.0  0.0   0:00.26 upstart-socket-br
 887 root        20   0 43448  1852 1488  S   0.0  0.1   0:00.46 /lib/systemd/syst
 890 root        20   0 25540   428  212  S   0.0  0.0   0:00.00 rpc.idmapd
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice - F8Nice + F9Kill F10Quit

```

*cgroups with Docker*

## Contact info

- Email: [viktor@cloudsigma.com](mailto:viktor@cloudsigma.com)
- WWW: <http://vpetersson.com>
- Twitter: [@vpetersson](https://twitter.com/vpetersson)
- LinkedIn: <http://www.linkedin.com/in/vpetersson>

## Resources

- This deck - <http://goo.gl/rKFT4C>
- Red Hat's Resource Management Guide  
- <http://goo.gl/tqh6l1>
- Cgroup in kernel docs - <http://goo.gl/MOX0xH>
- SUS15: LXC, Cgroups and Advanced Linux  
Container Technology Lecture - <http://goo.gl/6jb71g>
- Systemd's Resource Control - <http://goo.gl/dwUotd>
- Docker Run reference for LXC - <http://goo.gl/dmBIMK>
- Cgroupspy - <http://goo.gl/ahKvgs>