Managing Linux Resources with cgroups

Thursday, August 13, 2015: 01:45 PM - 02:45 PM,
Dolphin, Americas Seminar

Richard Young
Executive I.T. Specialist
IBM Systems Lab Services

Agenda

- Control groups overview
- Control groups what is new?
- Control groups configuration
- Assignment and display of cgroup
- CPU resource examples
- Memory resource examples
- Namespaces and containers

Linux Control Groups

- **What are they**?
  - Finer grain means to control resources among different processes in a Linux instance. Besides limiting access to an amount of resource, it can also prioritize, isolate, and account for resource usage.
  - Control groups are also known as "**cgroups**"
  - Allow for control of resources such as
    - CPU
    - Memory
    - Network
    - IO
    - Others…
  - **libcgroup** package can be used to more easily manage cgroups. Set of userspace tools. It contains man pages, commands, configuration files, and services (daemons)
    - Without libcgroup all configuration is lost at reboot
    - Without libcgroup tasks don't get automatically assigned to the proper cgroup
  - Two configuration files in /etc
    - **`cgconfig.conf`** and **`cgrules.conf`**

Where might they be of value?

- Linux servers hosting multiple applications, workloads, or middleware component instances
- Resource control or isolation of misbehaving applications
  - Memory leaks or spikes
  - CPU loops
  - Actively polling application code
- Need to limit an application or middleware to a subset of resources
  - For example 2 of 10 IFLs
  - 2 of 40 GB of memory or memory and swap (includes limiting filesystem cache)
  - Assign a relative priority to one workload over another
  - Throttle CPU to a fraction of available CPU.
    - Making more resource available to other workloads in the same server
    - Making more resource available to other server in the same or other z/VMs or LPARs.

Where might they be of value?

- Security Isolation
  - Help prevent or limit scope of a denial of service attack
  - Resource usage by a given process(s) made finite
  - Name space container isolation also limits what other processes can see in terms of pids, network (unique IPs and unique loopback), UTS (hostname domain), filesystem mount (remount root readonly), IPC and user information

Linux Control Groups – Subsystems

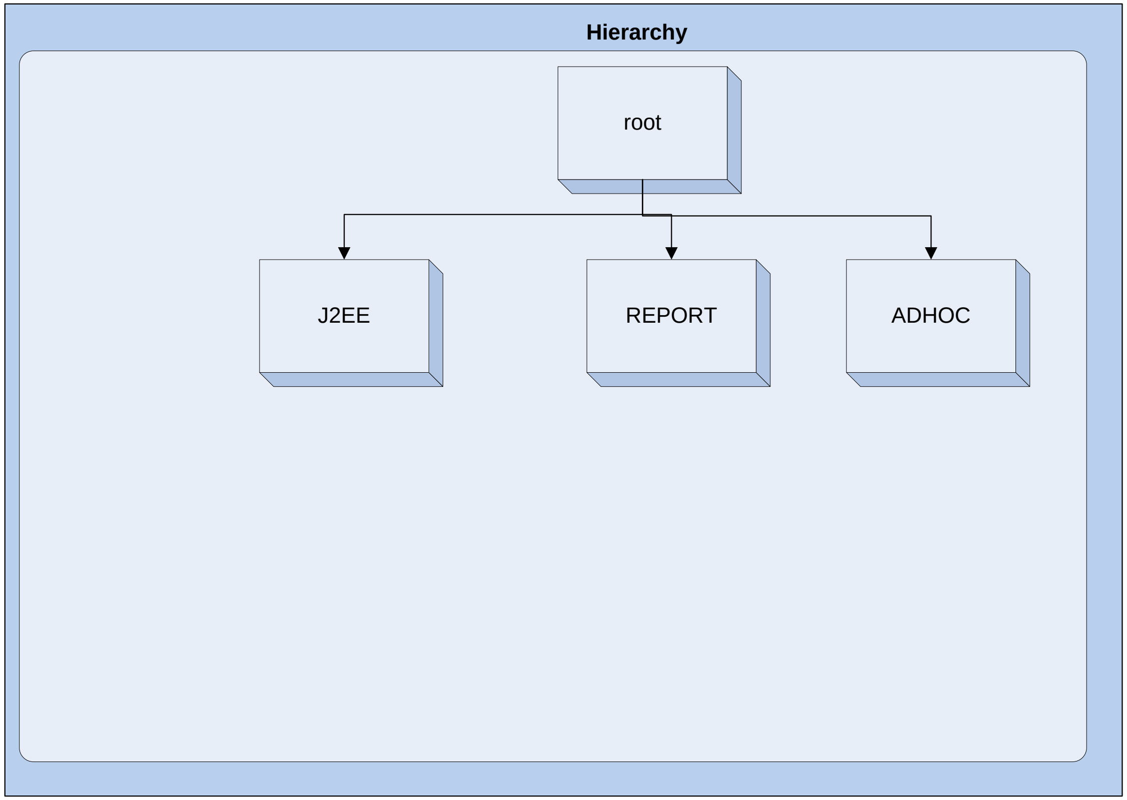
- **Subsystems – aka Resource Controllers**
  - blkio – control/limit IO from block devices
  - cpu - uses the kernels scheduler to control access to cpu resource
  - cpuacct – reporting of cpu usage
  - cpuset – assignment of cpu and memory nodes
    - cpusets.cpus (mandatory)
    - cpusets.mems (mandatory)
    - others optional
  - devices – allow or deny device access
  - freezer – suspend or resume tasks
  - hugetlb – controls and reporting on hugepages
  - memory – limit and report on memory usage by tasks
  - net_cls – tags network packets with class id
  - net_prio – by network interface set priority of network traffic
  - ns – namespace subsystem
- `lssubsys` - list hierarchies containing subsystem

Hierarchy Concepts

- cgroup are groups and they have sub groups much like a directory structure with subdirectories.
- The hierarchy is mounted as a virtual file system you can view and work with directly
- Initially everything belongs to the root level control group hierarchy and resources at that level would typically be unlimited.
- Children of processes inherit cgroup assignments from their parent. After the child has been created, it is managed independently of the parent.
- A control group hierarchy can have one or more subsystem associated with it
- A subsystem (resource) can only attach to one cgroup hierarchy unless the additional cgroup hierarchies contain only the same subsystem.
- If a task is added to a 2nd cgroup in a hierarchy, it is removed from the first cgroup

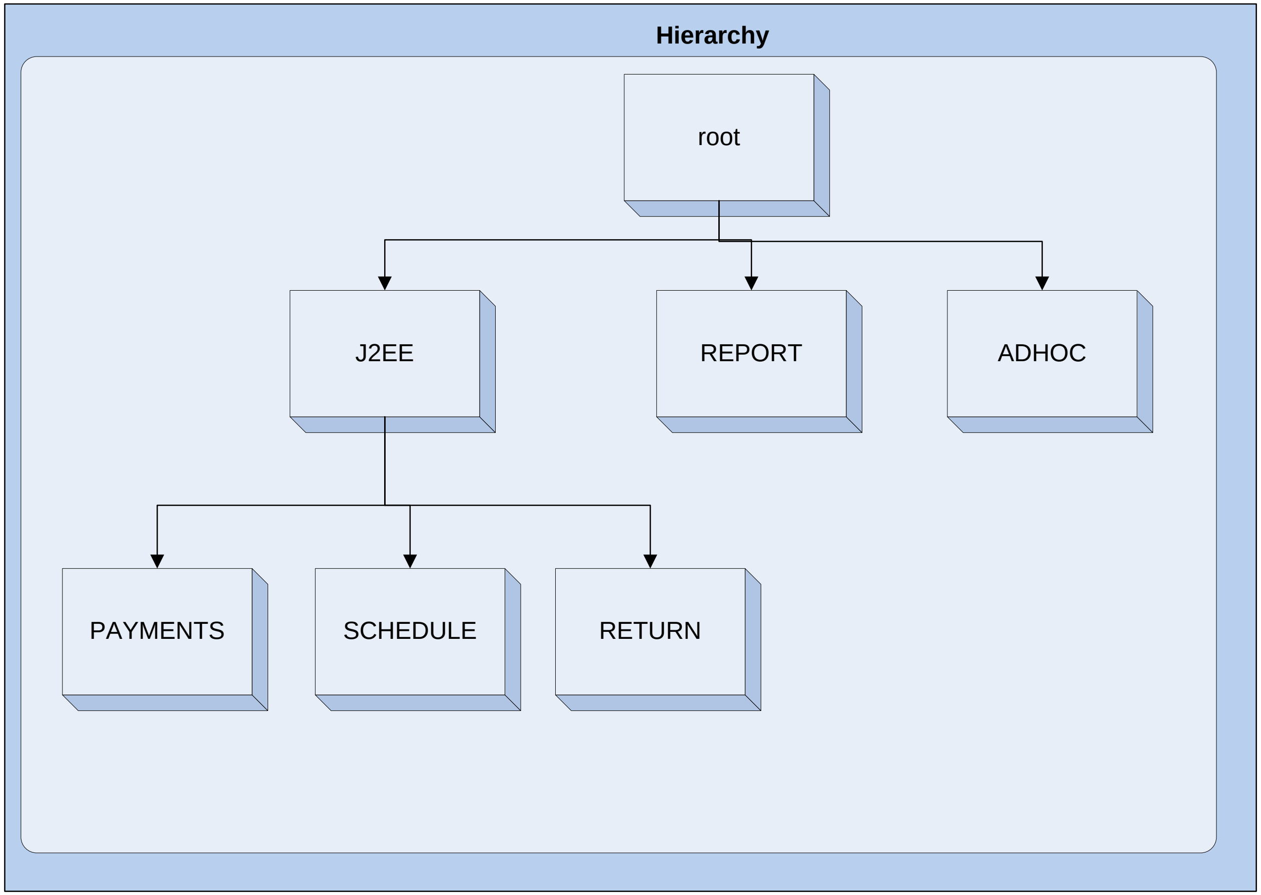
cgroups Hierarchy

- An example of workloads broken in to three distinct categories



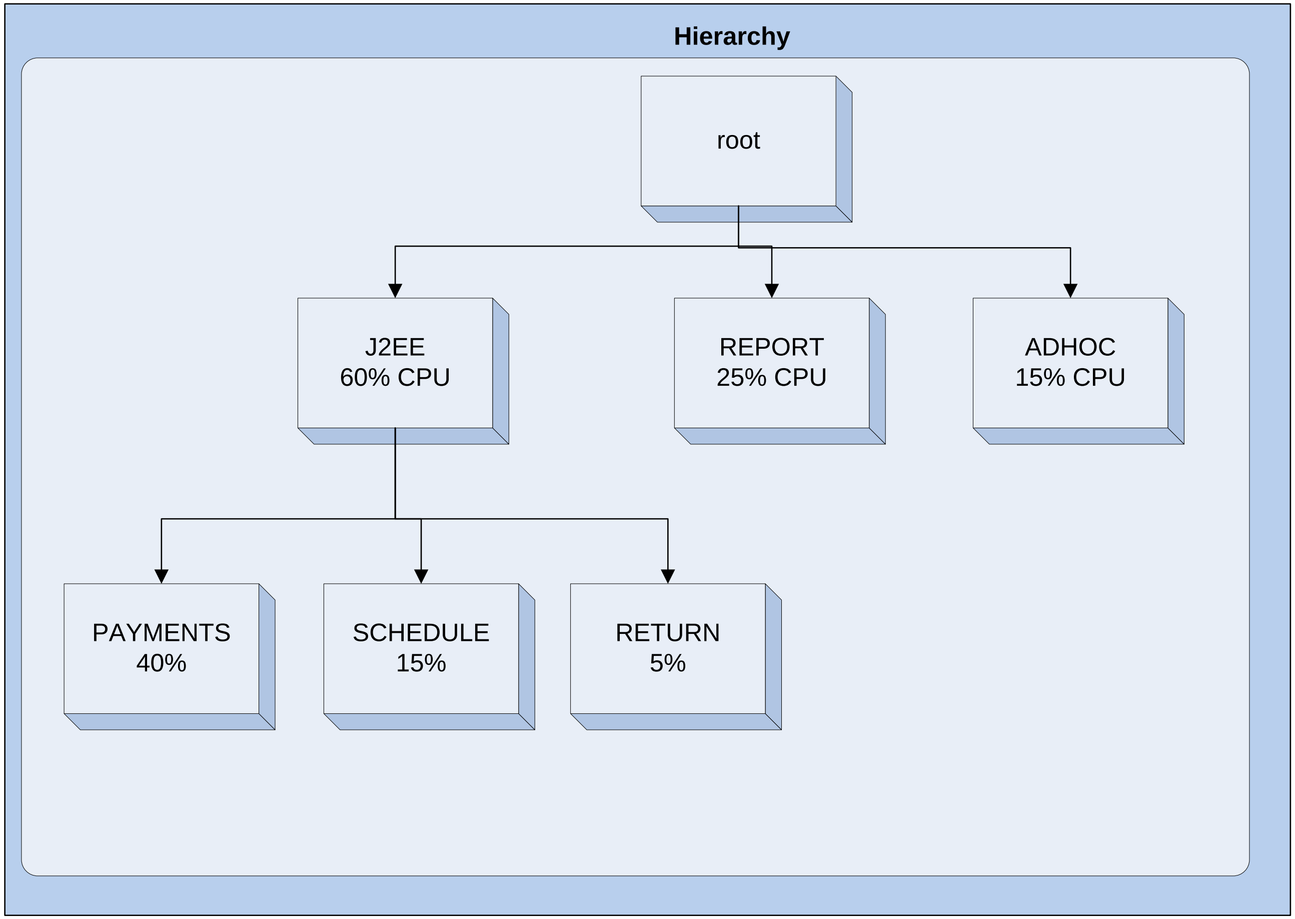
cgroups Hierarchy

- The J2EE hierarchy extended in to three distinct subgroups



cgroups Hierarchy

- Resource allocations applied based upon business need of workloads



Agenda

- Control groups overview
- ➢ **Control groups what is new?**
- Control groups configuration
- Assignment and display of cgroup
- CPU resource examples
- Memory resource examples
- Namespaces and containers

Systemd and control groups

- Starting with RHEL 7/ SLES 12, systemd is primarily responsible for application bindings. You can use systemctl commands or modify systemd unit files directly
- https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Resource_Management_Guide/chap-Introduction_to_Control_Groups.html
  - "This package is now deprecated and it is not recommended to use it since it can easily create conflicts with the default cgroup hierarchy. However, libcgroup is still available to cover for certain specific cases, where system is not yet applicable, most notably for using the net-prio subsystem"

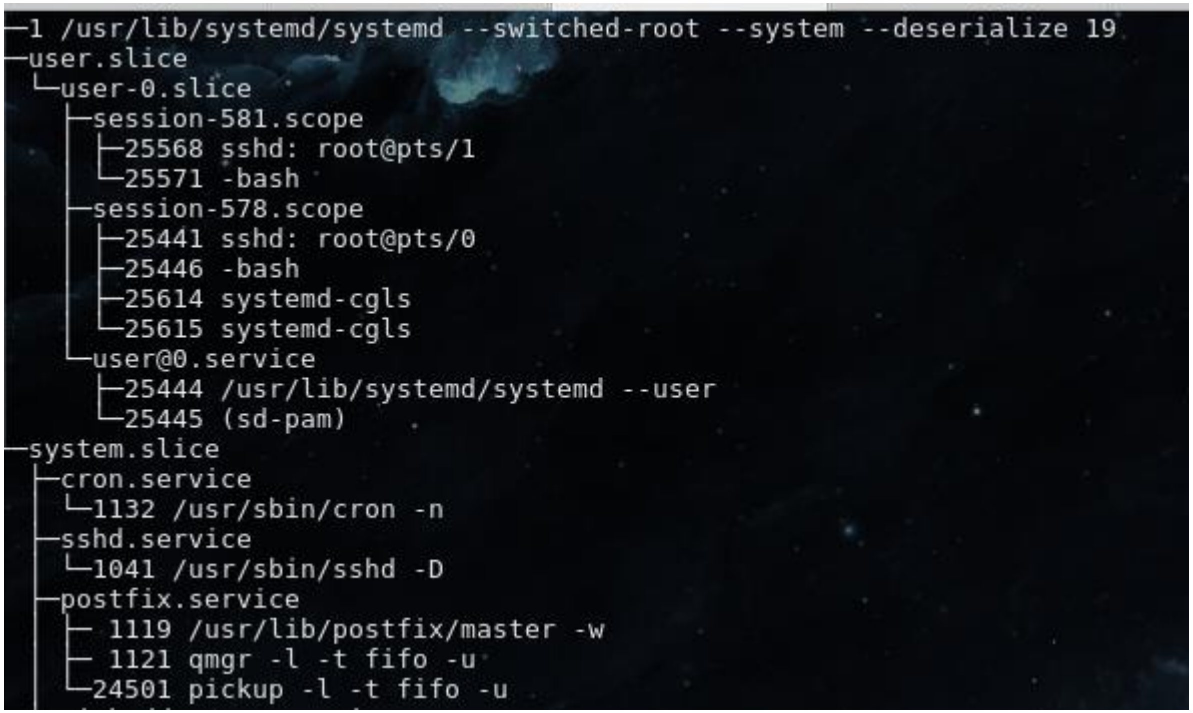
- All processes on the system are children of systemd init
- Command `systemd-cgls shows cgroup hierarchy in a tree`
- Documentation in kernel doc `/usr/share/doc/kernel-doc-<version>/Documentation/cgroups`

Systemd and cgroups overview

- Systemd provides three main unit types:
  - **Services** contain one or more processes that are started and stopped by systemd based on configuration.
  - **Scopes** contain one ore more processes that are started by arbitrary processes via fork()
  - **Slices** are used to group services and scopes together
- Service, scope and slice units map to objects in the cgroup filesystem.

Systemd control group list and slices

- Command `systemd-cgls`
  - User and System slice shown.
  - Machine slice (for virtual services and containers) not shown
  - Processes automatically placed slices



```
─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 19
─user.slice
 └─user-0.slice
   ├─session-581.scope
   │ ├─25568 sshd: root@pts/1
   │ └─25571 -bash
   ├─session-578.scope
   │ ├─25441 sshd: root@pts/0
   │ ├─25446 -bash
   │ ├─25614 systemd-cgls
   │ └─25615 systemd-cgls
   └─user@0.service
     ├─25444 /usr/lib/systemd/systemd --user
     └─25445 (sd-pam)
─system.slice
 ├─cron.service
 │ └─1132 /usr/sbin/cron -n
 ├─sshd.service
 │ └─1041 /usr/sbin/sshd -D
 ├─postfix.service
 │ ├─ 1119 /usr/lib/postfix/master -w
 │ ├─ 1121 qmgr -l -t fifo -u
 │ └─24501 pickup -l -t fifo -u
```

Systemd and control groups

- Two types of cgroups
  - Transient
    - Via `systemd-run` or API calls to systemd
    - Removed when service is stopped
  - Persistent
    - Created by editing unit configuration files
    - `systemctl enable`
- `systemctl set-property`
  - Persistently change resource controls during application runtime
  - `systemctl set-property was.service CPUShares=800 MemoryLimit=1500M`

Systemd and control groups

- Edit service file in /usr/lib/systemd/system/ <xxx>.service
- Make/Change resource control entry in file:

```
 [Service]
CPUShare=2000
MemoryLimit=1500M
ControlGroupAttribute=memory.swappiness  70
```

- `systemctl daemon-reload`
- `systemctl restart <xxx>.service`

- `systemd-cgtop` – top control groups by resource usage

Systemd and control groups

- The cpu controller is enabled by default in the kernel
  - Every system service receives the same amount of CPU no matter how many processes

Systemd’s support cgroup attributes is evolving

- Systemd unit file can set the following cgroup related parameters
  - CPUAccounting=
  - CPUShares=weight
  - MemoryAccounting=
  - MemoryLimit=bytes, MemorySoftLimit=bytes
  - BlockIOAccounting=
  - BlockIOWeight=weight
  - BlockIODeviceWeight=device weight
  - BlockIOReadBandwidth=device bytes, BlockIOWriteBandwidth=device bytes
  - DeviceAllow=
  - DevicePolicy=auto|closed|strict
  - Slice=

Systemd and cgroup attributes

- Where no high level cgroup attribute is available in systemd, there is:
- ControlGroupAttribute=<<attribute>> <<value>>
- A service file example, might look like:

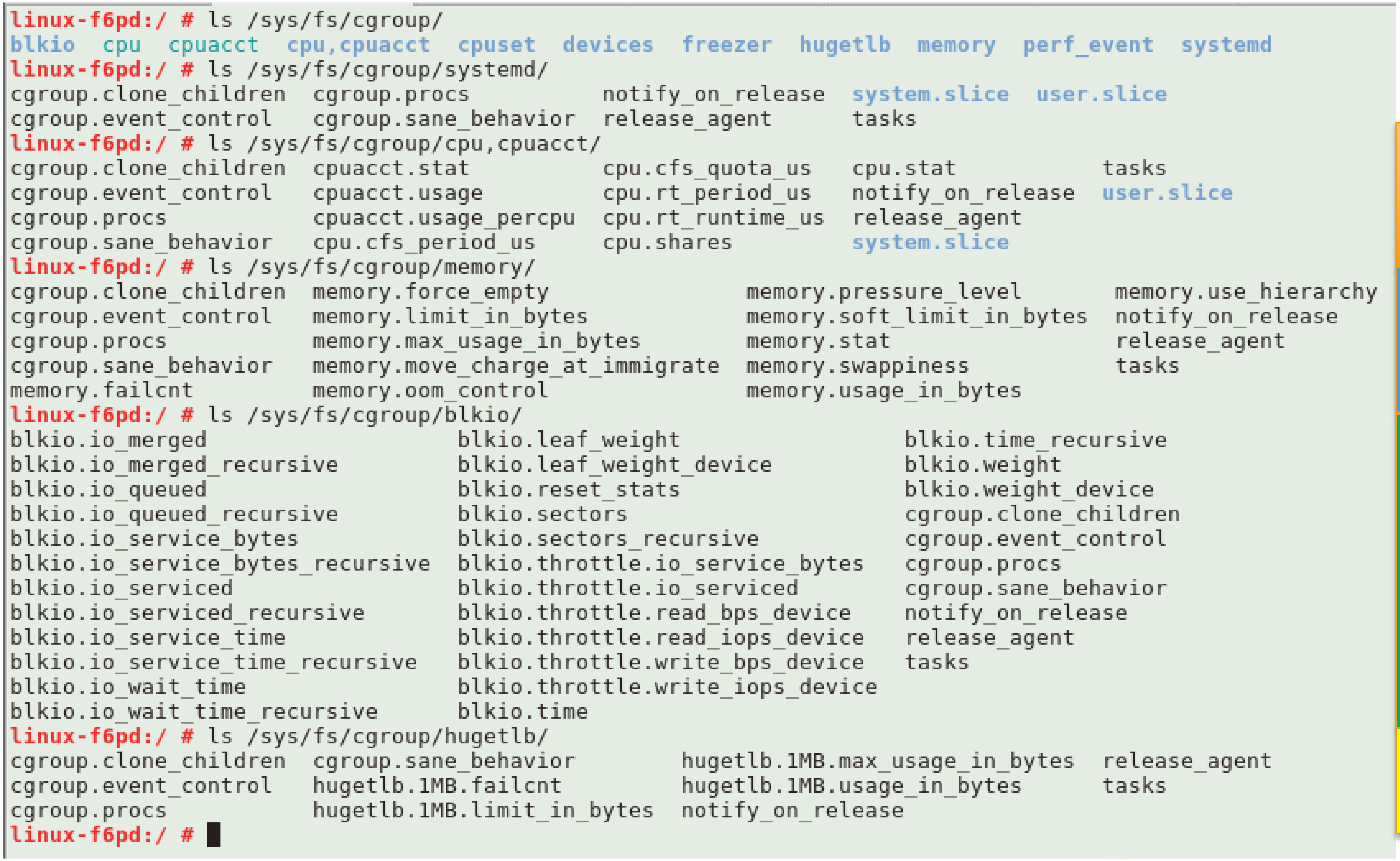
```
 [Service]
ControlGroupAttribute=memory.swappiness 70
```

The only constant in the universe is change

- https://bugzilla.redhat.com/show_bug.cgi?id=1172890

- Lukáš Nykrýn 2014-12-11 01:35:42 EST This option was removed in 208 From 208 release notes:* As discussed earlier, the low-level cgroup configuration options ControlGroup=, ControlGroupModify=, ControlGroupPersistent=, ControlGroupAttribute= have been removed. Please use high-level attribute settings instead as well as slice units. And some other info http://lwn.net/Articles/555923/

A look at the cgroup sys fs



```
linux-f6pd:/ # ls /sys/fs/cgroup/
blkio  cpu  cpuacct  cpu,cpuacct  cpuset  devices  freezer  hugetlb  memory  perf_event  systemd
linux-f6pd:/ # ls /sys/fs/cgroup/systemd/
cgroup.clone_children  cgroup.procs          notify_on_release  system.slice  user.slice
cgroup.event_control   cgroup.sane_behavior  release_agent      tasks
linux-f6pd:/ # ls /sys/fs/cgroup/cpu,cpuacct/
cgroup.clone_children  cpuacct.stat          cpu.cfs_quota_us   cpu.stat           tasks
cgroup.event_control   cpuacct.usage         cpu.rt_period_us   notify_on_release  user.slice
cgroup.procs           cpuacct.usage_percpu  cpu.rt_runtime_us  release_agent
cgroup.sane_behavior   cpu.cfs_period_us     cpu.shares         system.slice
linux-f6pd:/ # ls /sys/fs/cgroup/memory/
cgroup.clone_children  memory.force_empty               memory.pressure_level       memory.use_hierarchy
cgroup.event_control   memory.limit_in_bytes            memory.soft_limit_in_bytes  notify_on_release
cgroup.procs           memory.max_usage_in_bytes        memory.stat                 release_agent
cgroup.sane_behavior   memory.move_charge_at_immigrate  memory.swappiness           tasks
memory.failcnt         memory.oom_control               memory.usage_in_bytes
linux-f6pd:/ # ls /sys/fs/cgroup/blkio/
blkio.io_merged                   blkio.leaf_weight                 blkio.time_recursive
blkio.io_merged_recursive         blkio.leaf_weight_device          blkio.weight
blkio.io_queued                   blkio.reset_stats                 blkio.weight_device
blkio.io_queued_recursive         blkio.sectors                     cgroup.clone_children
blkio.io_service_bytes            blkio.sectors_recursive           cgroup.event_control
blkio.io_service_bytes_recursive  blkio.throttle.io_service_bytes   cgroup.procs
blkio.io_serviced                 blkio.throttle.io_serviced        cgroup.sane_behavior
blkio.io_serviced_recursive       blkio.throttle.read_bps_device    notify_on_release
blkio.io_service_time             blkio.throttle.read_iops_device   release_agent
blkio.io_service_time_recursive   blkio.throttle.write_bps_device   tasks
blkio.io_wait_time                blkio.throttle.write_iops_device
blkio.io_wait_time_recursive      blkio.time
linux-f6pd:/ # ls /sys/fs/cgroup/hugetlb/
cgroup.clone_children  cgroup.sane_behavior        hugetlb.1MB.max_usage_in_bytes  release_agent
cgroup.event_control   hugetlb.1MB.failcnt         hugetlb.1MB.usage_in_bytes      tasks
cgroup.procs           hugetlb.1MB.limit_in_bytes  notify_on_release
linux-f6pd:/ #
```

CPU

Memory

Blkio

hugetlb

Agenda

- Control groups overview
- Control groups what is new?
- ➢ **Control groups configuration**
- Assignment and display of cgroup
- CPU resource examples
- Memory resource examples
- Namespaces and containers

Control Group Configuration

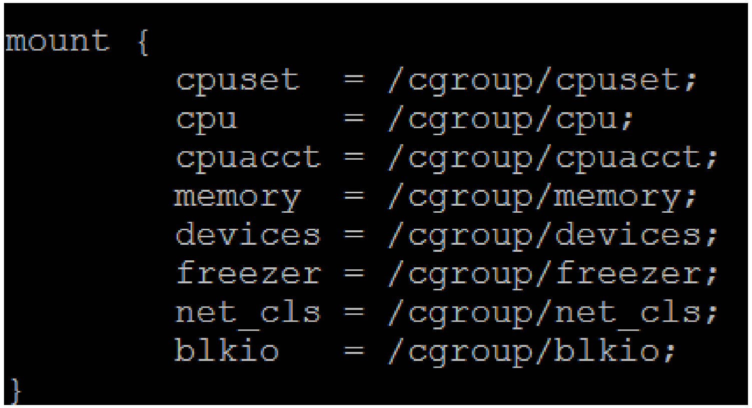
- Cgroup can be configured three main ways (without the user writing code)
  - Manually via the /sys/fs/cgroup
  - Via libcgroup commands
  - Via systemd commands
- Remember you don't generally mix systemd and libcgroup implementations

libcgroup summary

- Commands
  - `cgclassify` – Assign processes to specific control groups
  - `cgcreate/cgdelete` - create/delete a control group
  - `cgexec` – start a process with a cgroup assignment
  - `cgget/cgset` – Set or retrieve parameter values on a control group
  - `cgsnapshot` – Capture the current control group to a file
  - `lscgroup` – List the control group
  - `lssubsys` – List the hierarchies containing the subsystems
- System V init services
  - cgconfig
  - cgred

Control group configuration - definition

- `cgcreate` command to create new cgroups
  - Basic format `cgreate –g subsytem: path`
  - Does NOT add the new cgroup to the cgconfig.conf
- `cgdelete` command to delete cgroups
  - `cgdelete subsystem: path`
  - -r option to recursively delete subgroups
- cgconfig service mounts hierarchy based on /etc/cgconfig.conf
  - Changes to cgconfig.conf requires a restart of the service to become effective
  - Hierarchy could be mounted manually but it is recommend to use the configuration file and supplied service



```
mount {
        cpuset  = /cgroup/cpuset;
        cpu     = /cgroup/cpu;
        cpuacct = /cgroup/cpuacct;
        memory  = /cgroup/memory;
        devices = /cgroup/devices;
        freezer = /cgroup/freezer;
        net_cls = /cgroup/net_cls;
        blkio   = /cgroup/blkio;
}
```

Control Group Configuration

- A more complex cgconfig.conf

```
mount { cpuset = /cgroup/cpu_and_mem;
        cpu = /cgroup/cpu_and_mem;
        cpuacct = /cgroup/cpu_and_mem;
        memory = /cgroup/cpu_and_mem;
}
group blue-subgroup {
       cpu {  cpu.cfs_period_us="100000";
                cpu.cfs_quota_us=“1000";   }
        memory {   memory.swappiness="0";           }
}
group red-subgroup {
    cpu {  cpu.cfs_period_us="100000";
                cpu.cfs_quota_us="-1";     }
     memory {  memory.swappiness="60";          }
}
```

Setting cgroup parameters

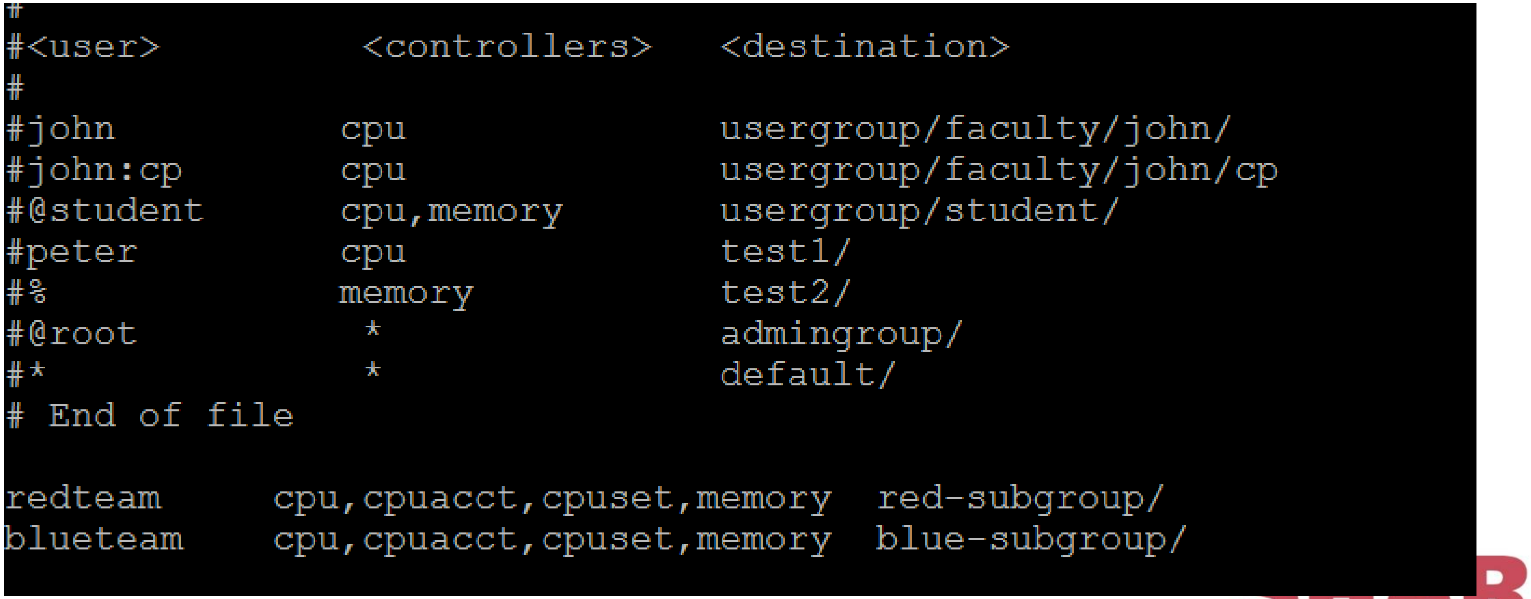
- The `cgset` command allows you to the parameter values or limits for a given control group
  - `cgset -r parm=value <cgroup path>`
  - `cgset -r memory.limit_in_bytes=2m red-subgroup`
  - Does not update cgconfig.conf
- Be aware that some resources controllers have mandatory parameters that must be set

Agenda

- Control groups overview
- Control groups what is new?
- Control groups configuration
- ➢ **Assignment and display of cgroup**
- CPU resource examples
- Memory resource examples
- Namespaces and containers

Control group configuration - task assignment

- The `cgred` service assigns processes to cgroups based on `/etc/cgrules.conf`
  - `File has two formats`
    - `User susbystems cgroup`
    - `User:command susbystems cgroup`
  - Can identify a user or group
  - Supports wild cards



```
#
#<user>          <controllers>   <destination>
#
#john            cpu              usergroup/faculty/john/
#john:cp         cpu              usergroup/faculty/john/cp
#@student        cpu,memory       usergroup/student/
#peter           cpu              test1/
#%               memory           test2/
#@root            *               admingroup/
#*                *               default/
# End of file

redteam     cpu,cpuacct,cpuset,memory  red-subgroup/
blueteam    cpu,cpuacct,cpuset,memory  blue-subgroup/
```

Control group configuration - task assignment

- Processes can be directly assigned to a control group at invocation via `cgexec.`

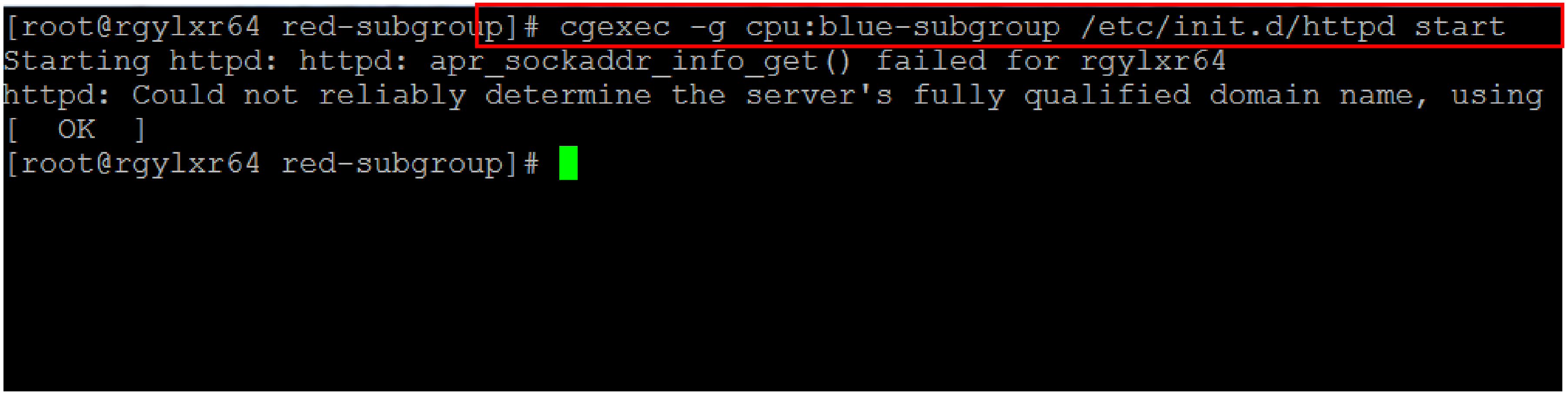
```
cgexec –g subsystem:cgroup  command
```

- Running processes can be moved dynamically to a control group

```
cgclassify –g subsystem:cgroup pid
```

Control group configuration - assignment

- `cgexec` used to start apache and assign it to the “blue-subgroup” for the cpu resource manager

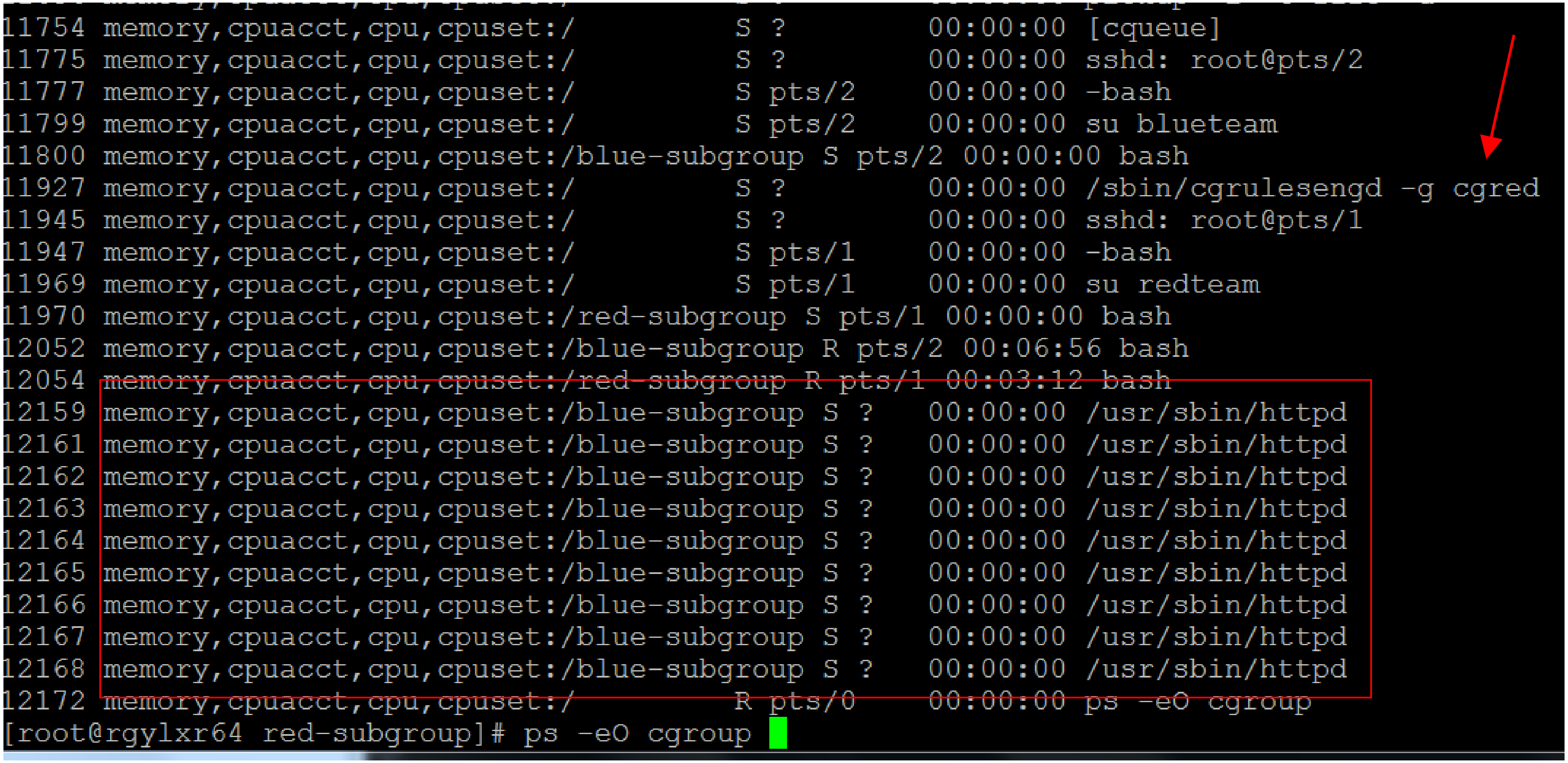


```
[root@rgylxr64 red-subgroup]# cgexec -g cpu:blue-subgroup /etc/init.d/httpd start
Starting httpd: httpd: apr_sockaddr_info_get() failed for rgylxr64
httpd: Could not reliably determine the server's fully qualified domain name, using
[  OK  ]
[root@rgylxr64 red-subgroup]#
```

- The results can be confirmed with ps –eO cgroup

Display process cgroup assignment

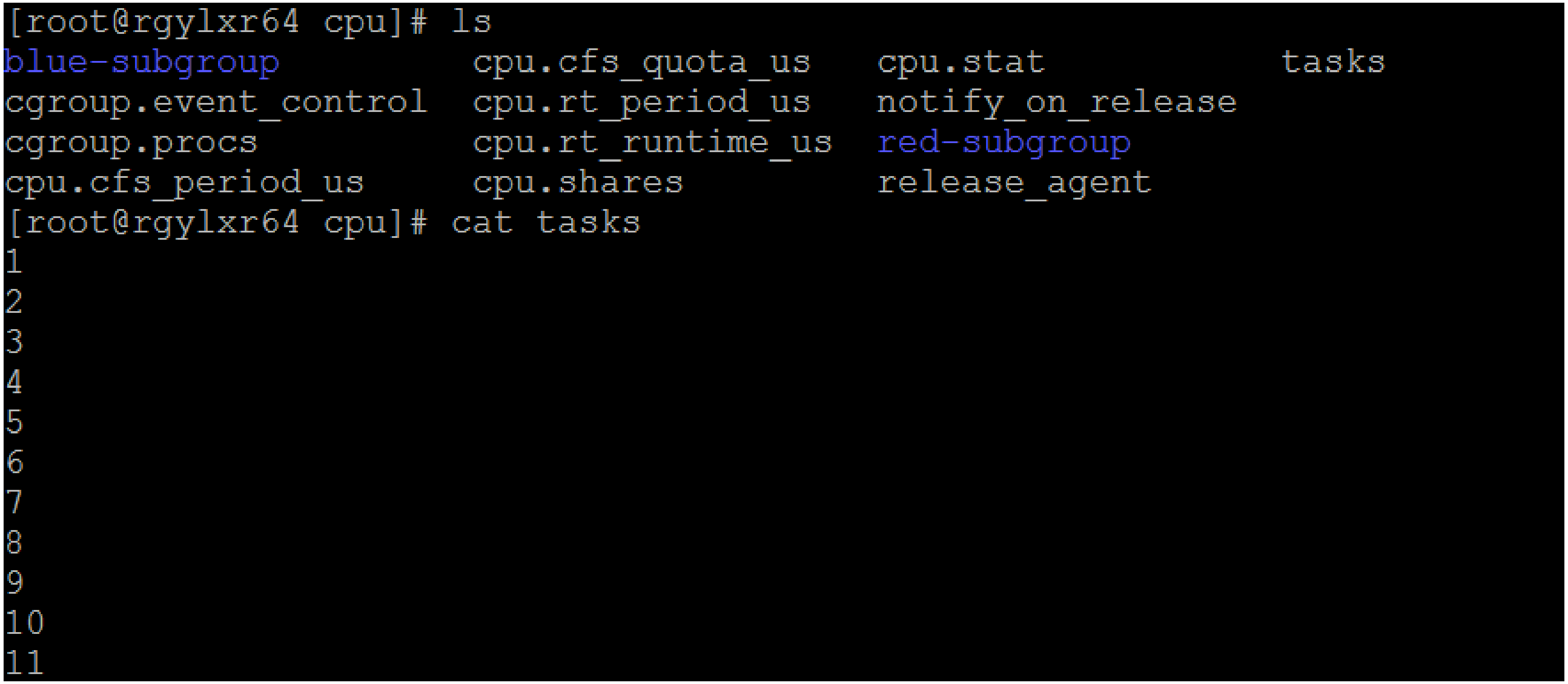
ps –eO cgroup



```
11754 memory,cpuacct,cpu,cpuset:/          S ?       00:00:00 [cqueue]
11775 memory,cpuacct,cpu,cpuset:/          S ?       00:00:00 sshd: root@pts/2
11777 memory,cpuacct,cpu,cpuset:/          S pts/2   00:00:00 -bash
11799 memory,cpuacct,cpu,cpuset:/          S pts/2   00:00:00 su blueteam
11800 memory,cpuacct,cpu,cpuset:/blue-subgroup S pts/2 00:00:00 bash
11927 memory,cpuacct,cpu,cpuset:/          S ?       00:00:00 /sbin/cgrulesengd -g cgred
11945 memory,cpuacct,cpu,cpuset:/          S ?       00:00:00 sshd: root@pts/1
11947 memory,cpuacct,cpu,cpuset:/          S pts/1   00:00:00 -bash
11969 memory,cpuacct,cpu,cpuset:/          S pts/1   00:00:00 su redteam
11970 memory,cpuacct,cpu,cpuset:/red-subgroup S pts/1 00:00:00 bash
12052 memory,cpuacct,cpu,cpuset:/blue-subgroup R pts/2 00:06:56 bash
12054 memory,cpuacct,cpu,cpuset:/red-subgroup R pts/1 00:03:12 bash
12159 memory,cpuacct,cpu,cpuset:/blue-subgroup S ?   00:00:00 /usr/sbin/httpd
12161 memory,cpuacct,cpu,cpuset:/blue-subgroup S ?   00:00:00 /usr/sbin/httpd
12162 memory,cpuacct,cpu,cpuset:/blue-subgroup S ?   00:00:00 /usr/sbin/httpd
12163 memory,cpuacct,cpu,cpuset:/blue-subgroup S ?   00:00:00 /usr/sbin/httpd
12164 memory,cpuacct,cpu,cpuset:/blue-subgroup S ?   00:00:00 /usr/sbin/httpd
12165 memory,cpuacct,cpu,cpuset:/blue-subgroup S ?   00:00:00 /usr/sbin/httpd
12166 memory,cpuacct,cpu,cpuset:/blue-subgroup S ?   00:00:00 /usr/sbin/httpd
12167 memory,cpuacct,cpu,cpuset:/blue-subgroup S ?   00:00:00 /usr/sbin/httpd
12168 memory,cpuacct,cpu,cpuset:/blue-subgroup S ?   00:00:00 /usr/sbin/httpd
12172 memory,cpuacct,cpu,cpuset:/          R pts/0   00:00:00 ps -eO cgroup
[root@rgylxr64 red-subgroup]# ps -eO cgroup
```

Display process cgroup assignment

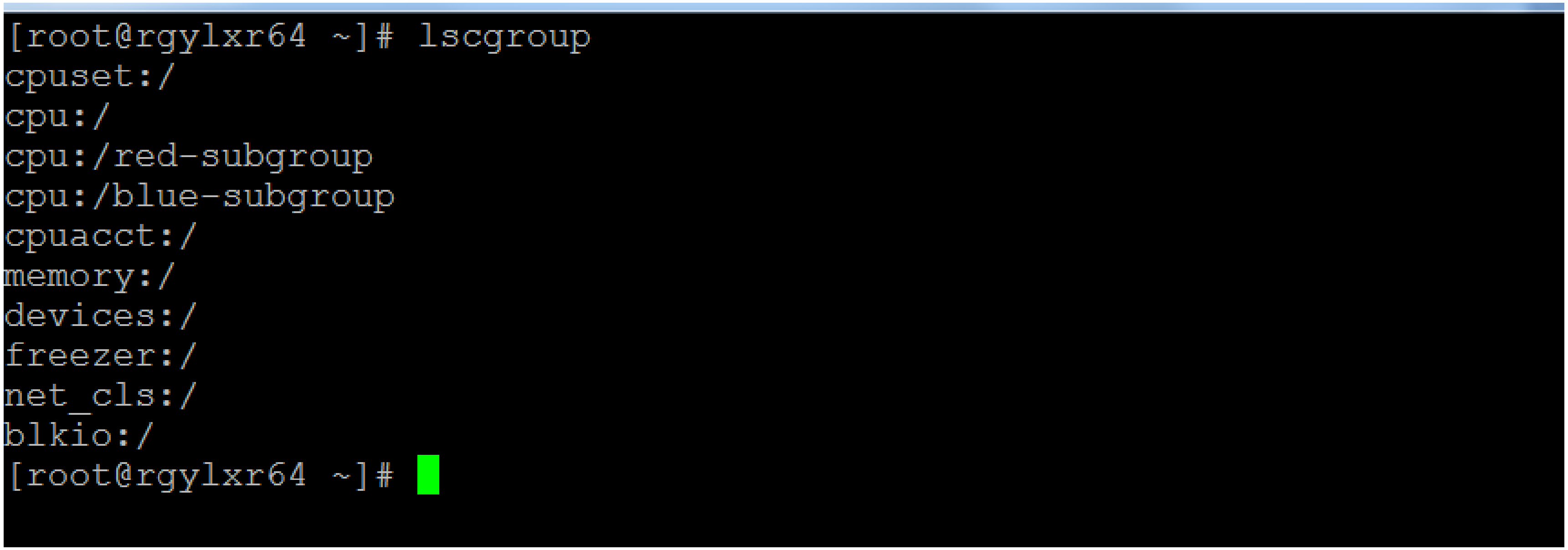
- Another method to display processes in a cgroup is to `cat` the "tasks" file in the given part of the hierarchy. The root level cgroup for CPU is shown.



```
[root@rgylxr64 cpu]# ls
blue-subgroup          cpu.cfs_quota_us   cpu.stat           tasks
cgroup.event_control   cpu.rt_period_us   notify_on_release
cgroup.procs           cpu.rt_runtime_us  red-subgroup
cpu.cfs_period_us      cpu.shares         release_agent
[root@rgylxr64 cpu]# cat tasks
1
2
3
4
5
6
7
8
9
10
11
```

Display cgroups example

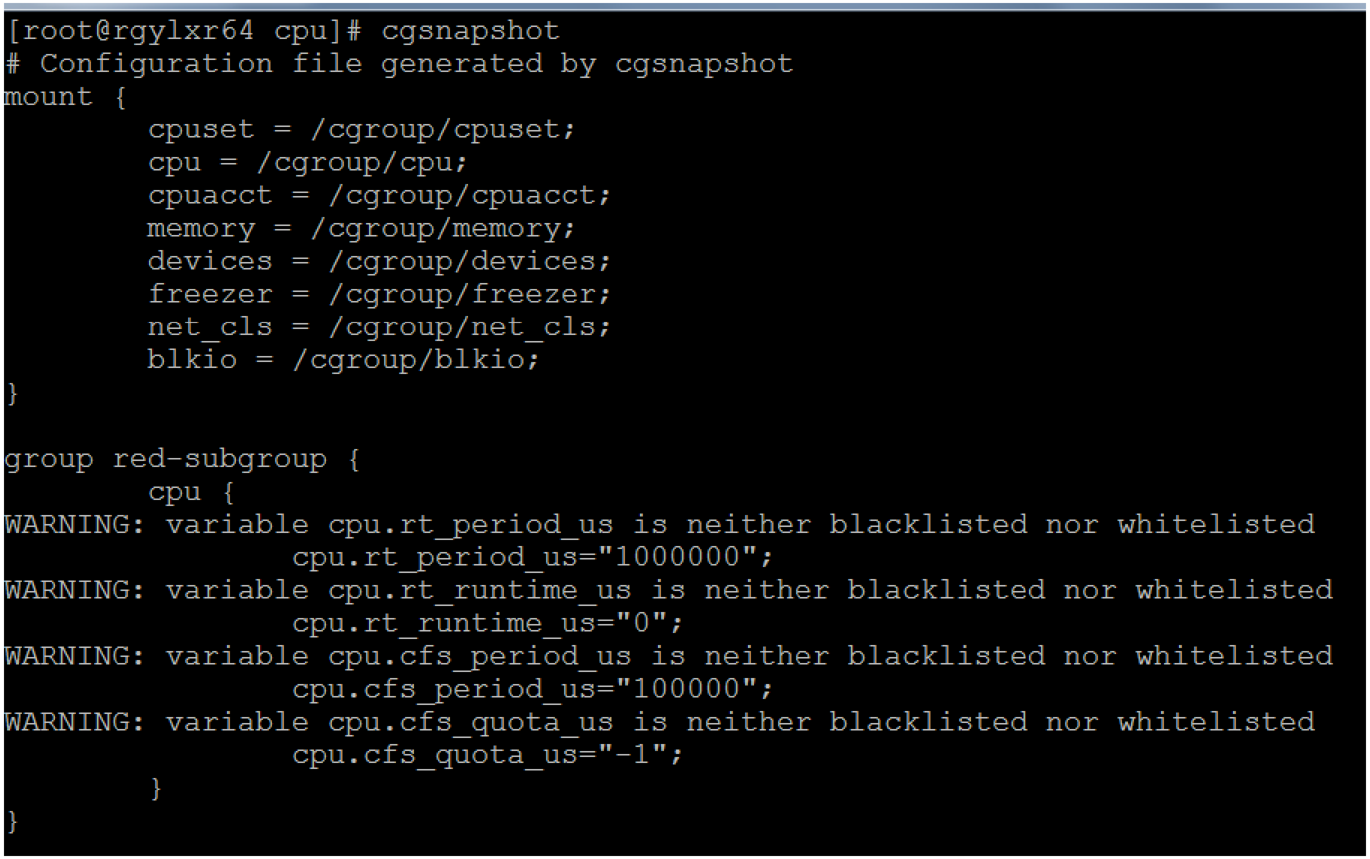
- `lscgroup` lists all defined control groups
- red-subgroup and blue-subgroup defined under root level cpu controller



```
[root@rgylxr64 ~]# lscgroup
cpuset:/
cpu:/
cpu:/red-subgroup
cpu:/blue-subgroup
cpuacct:/
memory:/
devices:/
freezer:/
net_cls:/
blkio:/
[root@rgylxr64 ~]#
```

Control group configuration

- `cgsnapshot` – generate new cgroups configuration file (cgconfig.conf) based on current runtime environment. Variables displayed can be blacklisted or whitelisted.



```
[root@rgylxr64 cpu]# cgsnapshot
# Configuration file generated by cgsnapshot
mount {
        cpuset = /cgroup/cpuset;
        cpu = /cgroup/cpu;
        cpuacct = /cgroup/cpuacct;
        memory = /cgroup/memory;
        devices = /cgroup/devices;
        freezer = /cgroup/freezer;
        net_cls = /cgroup/net_cls;
        blkio = /cgroup/blkio;
}

group red-subgroup {
        cpu {
WARNING: variable cpu.rt_period_us is neither blacklisted nor whitelisted
                cpu.rt_period_us="1000000";
WARNING: variable cpu.rt_runtime_us is neither blacklisted nor whitelisted
                cpu.rt_runtime_us="0";
WARNING: variable cpu.cfs_period_us is neither blacklisted nor whitelisted
                cpu.cfs_period_us="100000";
WARNING: variable cpu.cfs_quota_us is neither blacklisted nor whitelisted
                cpu.cfs_quota_us="-1";
        }
}
```

Agenda

- Control groups overview
- Control groups what is new?
- Control groups configuration
- Assignment and display of cgroup

➢ **CPU resource examples**
- Memory resource examples
- Namespaces and containers

CPU resource examples

- **cpusets and cpu.shares**
  - A cpuset assigns a processor or memory node
  - cpusets typically only have one memory node, except for NUMA architectures
  - cpu share assigns a relative portion of the CPU
- **cpu controlgroups support both realtime and non-realtime scheduled processes**
  - CFS vs RT schedulers
  - cpu.cfs_period_us and cpu.cfs_quota_us vs. cpu.rt_period_us and cpu.rt_runtime_us
  - CPU control group “Share” settings are not scheduler specific

Our business application simulator

After 1 minute of R&D we have a new agile program to simulate all new applications ☺

```
linux-f6pd:~ # cat cpuhog.sh
#!/usr/bin/env bash
        dd if=/dev/zero of=/dev/null
linux-f6pd:~ #
```

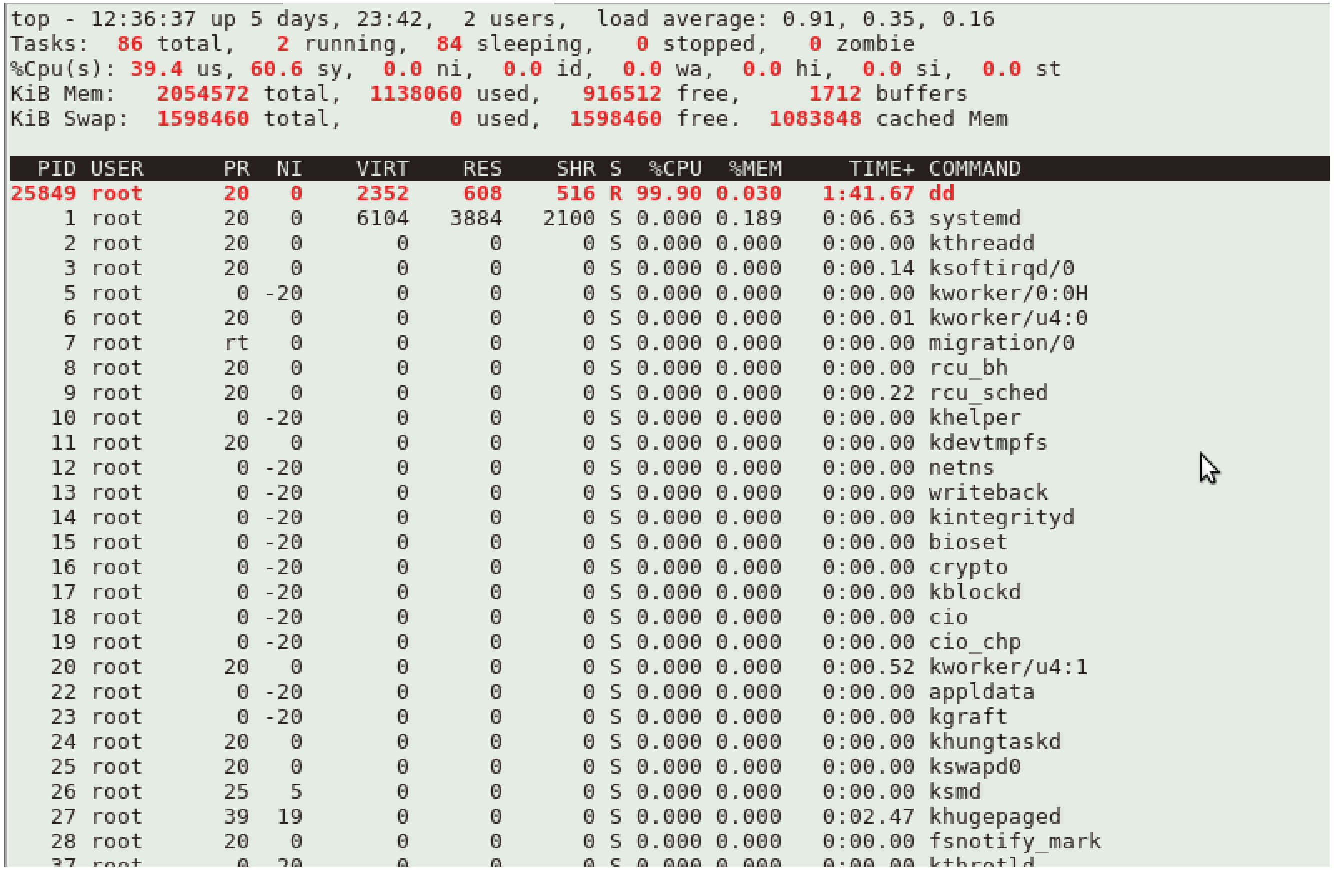
systemd-run

- systemd-run provides a means to run our program in a transient systemd unit in which is assigned to a default cgroup configuration
- The transient unit exists until the program or service completes

```
linux-f6pd:~ # vi cpuhog.sh
linux-f6pd:~ # systemd-run /root/cpuhog.sh
 Running as unit run-25847.service.
linux-f6pd:~ #
```

Simulator program consumption baseline

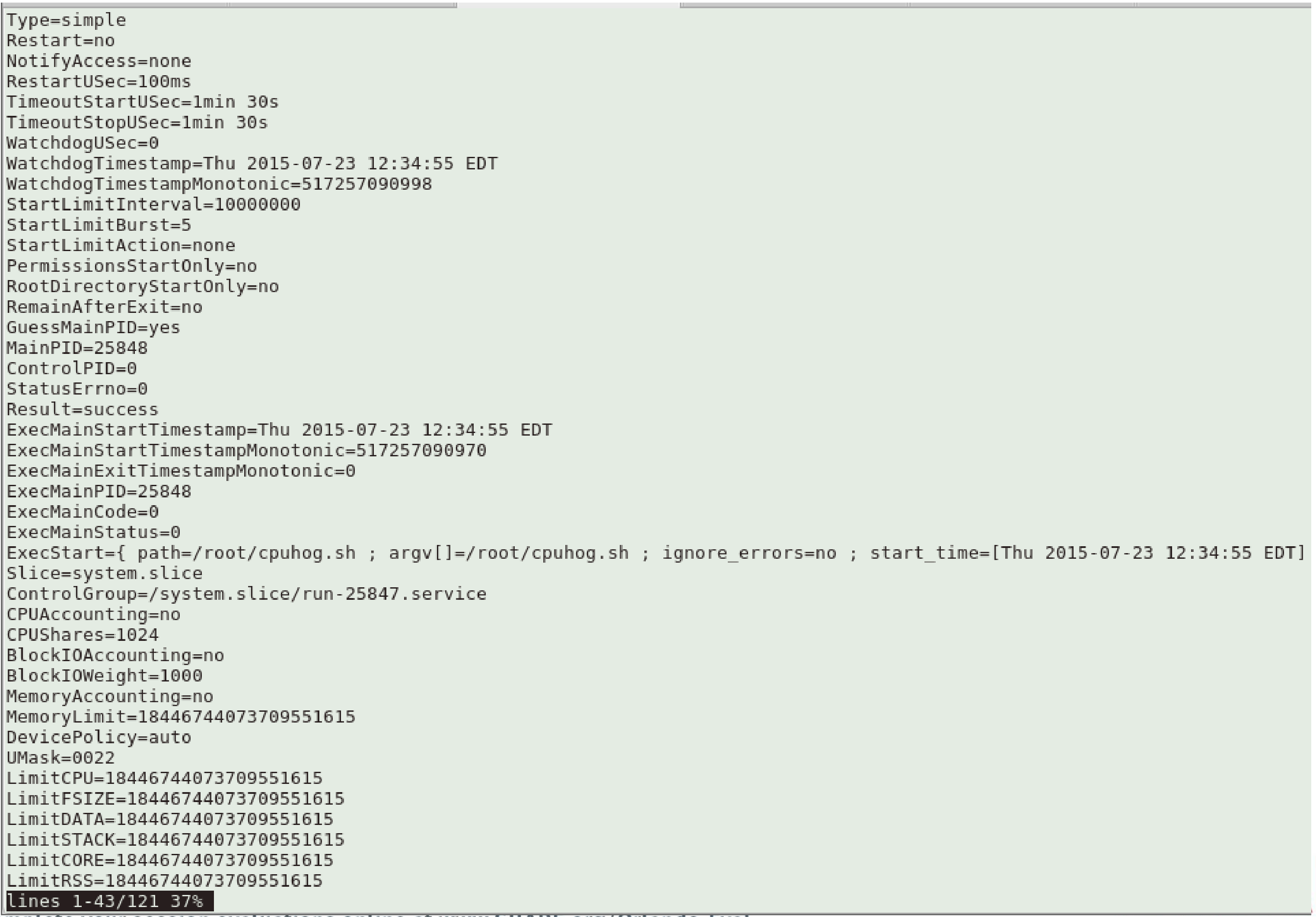
- A view from top



```
top - 12:36:37 up 5 days, 23:42,  2 users,  load average: 0.91, 0.35, 0.16
Tasks:  86 total,   2 running,  84 sleeping,   0 stopped,   0 zombie
%Cpu(s): 39.4 us, 60.6 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2054572 total,  1138060 used,   916512 free,     1712 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1083848 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
25849 root      20   0    2352    608    516 R 99.90 0.030   1:41.67 dd
    1 root      20   0    6104   3884   2100 S 0.000 0.189   0:06.63 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.14 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.22 rcu_sched
   10 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 khelper
   11 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 netns
   13 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 writeback
   14 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kintegrityd
   15 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 bioset
   16 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 crypto
   17 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kblockd
   18 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 cio
   19 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 cio_chp
   20 root      20   0       0      0      0 S 0.000 0.000   0:00.52 kworker/u4:1
   22 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 appldata
   23 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kgraft
   24 root      20   0       0      0      0 S 0.000 0.000   0:00.00 khungtaskd
   25 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kswapd0
   26 root      25   5       0      0      0 S 0.000 0.000   0:00.00 ksmd
   27 root      39  19       0      0      0 S 0.000 0.000   0:02.47 khugepaged
   28 root      20   0       0      0      0 S 0.000 0.000   0:00.00 fsnotify_mark
   37 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kthrotld
```

Systemctl show

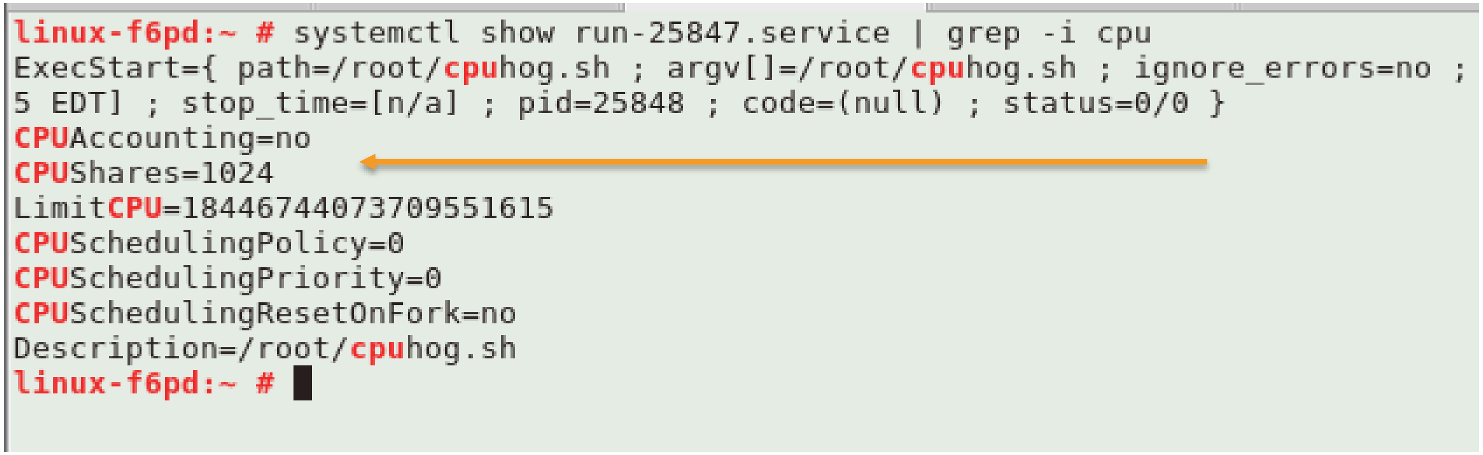
`systemctl show run-25847.service` provides detailed information about the service



```
Type=simple
Restart=no
NotifyAccess=none
RestartUSec=100ms
TimeoutStartUSec=1min 30s
TimeoutStopUSec=1min 30s
WatchdogUSec=0
WatchdogTimestamp=Thu 2015-07-23 12:34:55 EDT
WatchdogTimestampMonotonic=517257090998
StartLimitInterval=10000000
StartLimitBurst=5
StartLimitAction=none
PermissionsStartOnly=no
RootDirectoryStartOnly=no
RemainAfterExit=no
GuessMainPID=yes
MainPID=25848
ControlPID=0
StatusErrno=0
Result=success
ExecMainStartTimestamp=Thu 2015-07-23 12:34:55 EDT
ExecMainStartTimestampMonotonic=517257090970
ExecMainExitTimestampMonotonic=0
ExecMainPID=25848
ExecMainCode=0
ExecMainStatus=0
ExecStart={ path=/root/cpuhog.sh ; argv[]=/root/cpuhog.sh ; ignore_errors=no ; start_time=[Thu 2015-07-23 12:34:55 EDT]
Slice=system.slice
ControlGroup=/system.slice/run-25847.service
CPUAccounting=no
CPUShares=1024
BlockIOAccounting=no
BlockIOWeight=1000
MemoryAccounting=no
MemoryLimit=18446744073709551615
DevicePolicy=auto
UMask=0022
LimitCPU=18446744073709551615
LimitFSIZE=18446744073709551615
LimitDATA=18446744073709551615
LimitSTACK=18446744073709551615
LimitCORE=18446744073709551615
LimitRSS=18446744073709551615
lines 1-43/121 37%
```

CPU attributes of a system services

- Default CPUShares of 1024 assigned
- Notice there is no cpu period or quota
- LimitCPU like ulimit –t, or z/OS TIME=



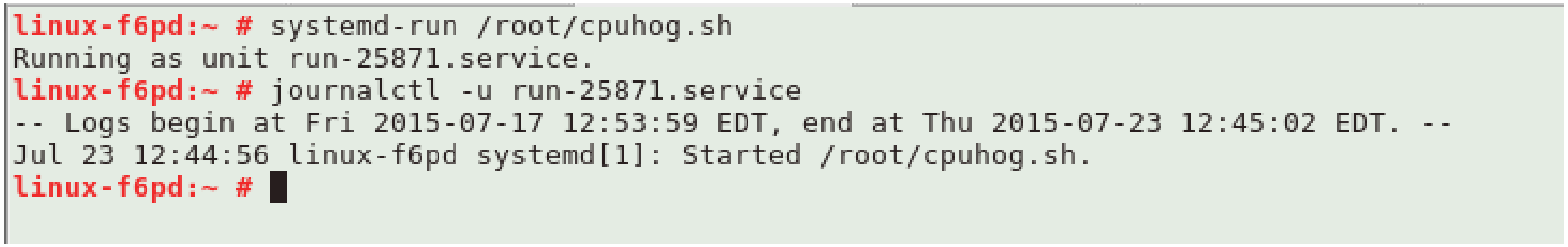
```
linux-f6pd:~ # systemctl show run-25847.service | grep -i cpu
ExecStart={ path=/root/cpuhog.sh ; argv[]=/root/cpuhog.sh ; ignore_errors=no ;
5 EDT] ; stop_time=[n/a] ; pid=25848 ; code=(null) ; status=0/0 }
CPUAccounting=no
CPUShares=1024
LimitCPU=18446744073709551615
CPUSchedulingPolicy=0
CPUSchedulingPriority=0
CPUSchedulingResetOnFork=no
Description=/root/cpuhog.sh
linux-f6pd:~ #
```

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

SHARE in Orlando 2015

Systemd control groups

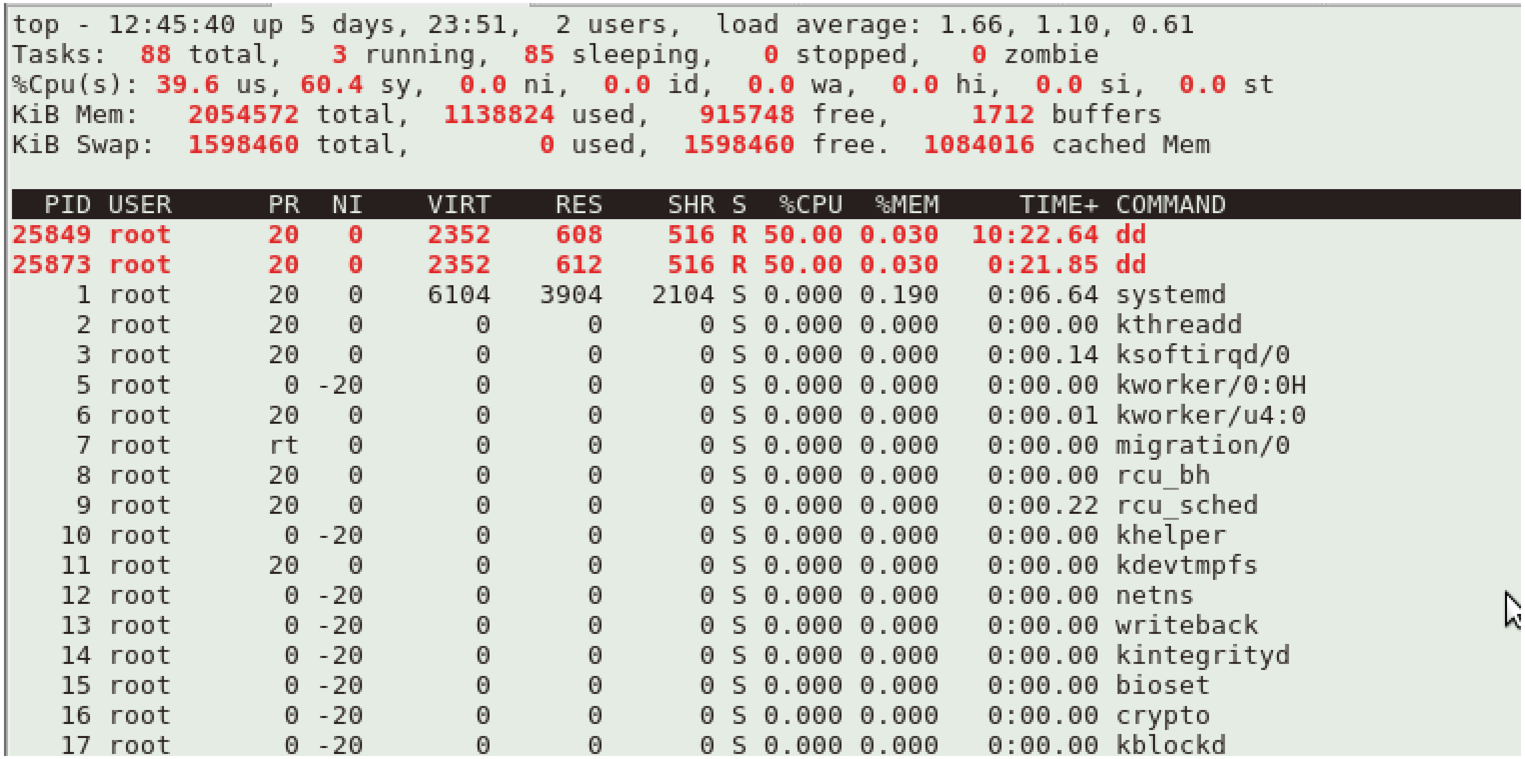
- Start a second simulated application
- And take a look at the resource assignment



```
linux-f6pd:~ # systemd-run /root/cpuhog.sh
Running as unit run-25871.service.
linux-f6pd:~ # journalctl -u run-25871.service
-- Logs begin at Fri 2015-07-17 12:53:59 EDT, end at Thu 2015-07-23 12:45:02 EDT. --
Jul 23 12:44:56 linux-f6pd systemd[1]: Started /root/cpuhog.sh.
linux-f6pd:~ #
```

Systemd and control groups

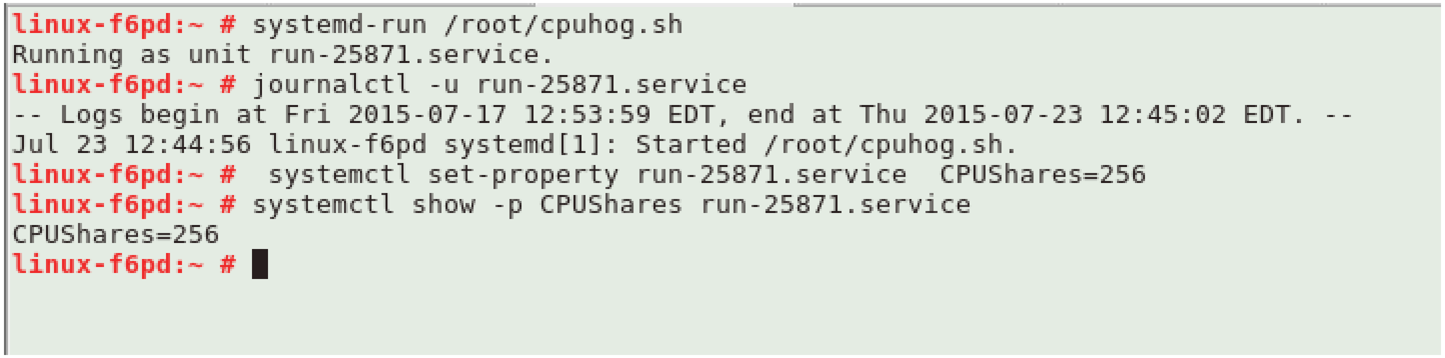
- Both program running with the defaults CPUShare value
- Both programs using the same amount of CPU



```
top - 12:45:40 up 5 days, 23:51,  2 users,  load average: 1.66, 1.10, 0.61
Tasks:  88 total,   3 running,  85 sleeping,   0 stopped,   0 zombie
%Cpu(s): 39.6 us, 60.4 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2054572 total,  1138824 used,   915748 free,     1712 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1084016 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
25849 root      20   0    2352    608    516 R 50.00 0.030  10:22.64 dd
25873 root      20   0    2352    612    516 R 50.00 0.030   0:21.85 dd
    1 root      20   0    6104   3904   2104 S 0.000 0.190   0:06.64 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.14 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.22 rcu_sched
   10 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 khelper
   11 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 netns
   13 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 writeback
   14 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kintegrityd
   15 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 bioset
   16 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 crypto
   17 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kblockd
```

Systemctl set-property and show

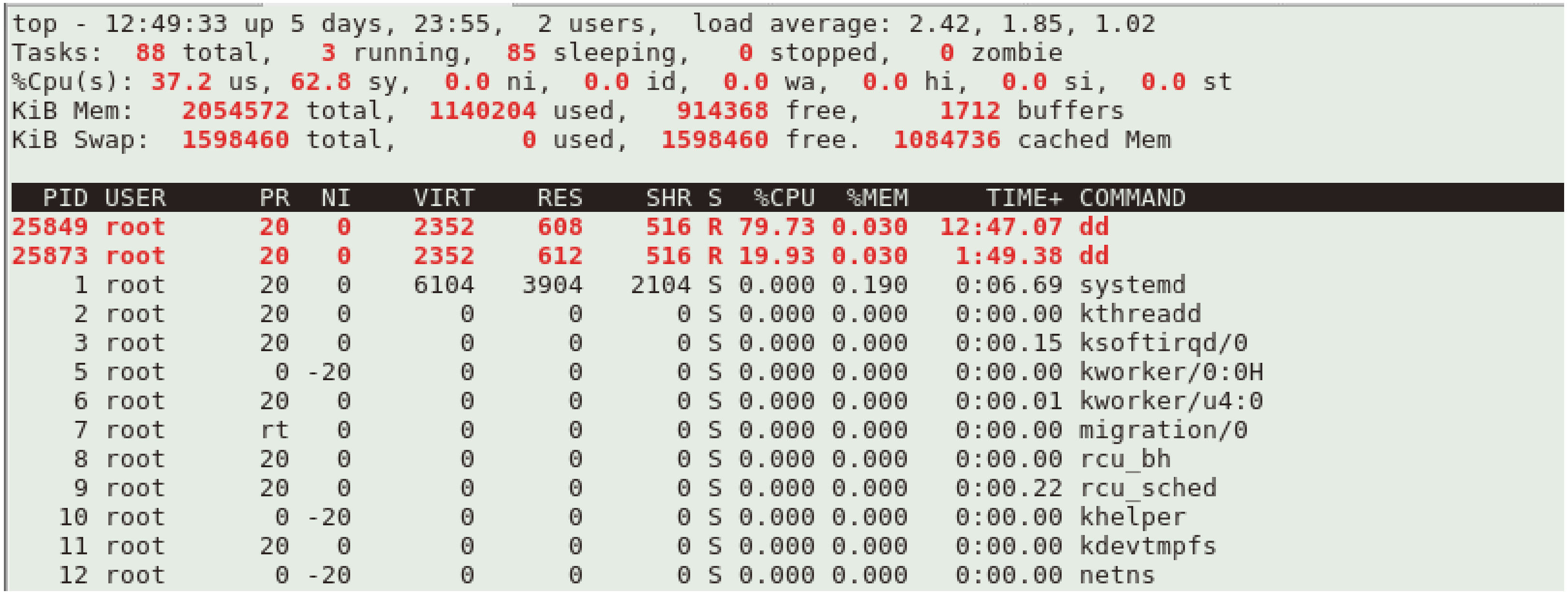
- Setting a lower CPUShare



```
linux-f6pd:~ # systemd-run /root/cpuhog.sh
Running as unit run-25871.service.
linux-f6pd:~ # journalctl -u run-25871.service
-- Logs begin at Fri 2015-07-17 12:53:59 EDT, end at Thu 2015-07-23 12:45:02 EDT. --
Jul 23 12:44:56 linux-f6pd systemd[1]: Started /root/cpuhog.sh.
linux-f6pd:~ #  systemctl set-property run-25871.service  CPUShares=256
linux-f6pd:~ # systemctl show -p CPUShares run-25871.service
CPUShares=256
linux-f6pd:~ #
```

Systemd and CPUShares

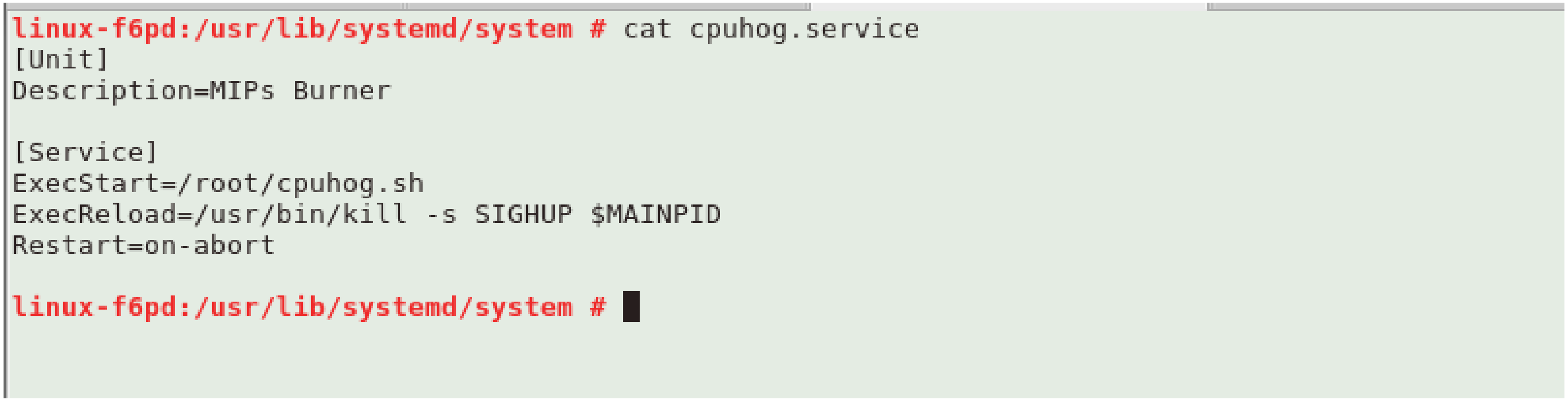
- Shares of 256 and default of 1024



```
top - 12:49:33 up 5 days, 23:55,  2 users,  load average: 2.42, 1.85, 1.02
Tasks:  88 total,   3 running,  85 sleeping,   0 stopped,   0 zombie
%Cpu(s): 37.2 us, 62.8 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2054572 total,  1140204 used,   914368 free,     1712 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1084736 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
25849 root      20   0    2352    608    516 R 79.73 0.030  12:47.07 dd
25873 root      20   0    2352    612    516 R 19.93 0.030   1:49.38 dd
    1 root      20   0    6104   3904   2104 S 0.000 0.190   0:06.69 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.15 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.22 rcu_sched
   10 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 khelper
   11 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 netns
```

Defining a persistent system services

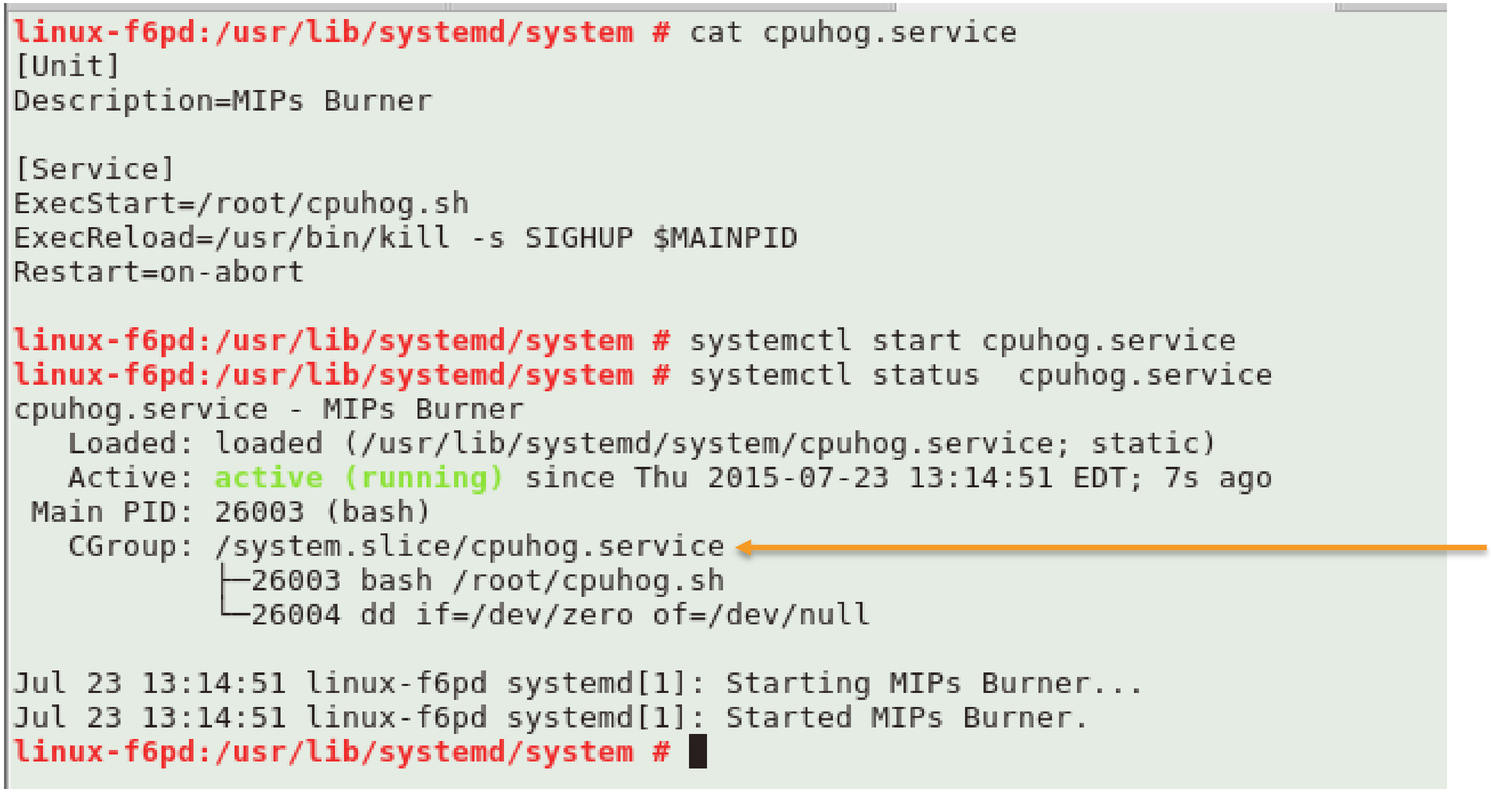
- To be persistent, the control group are placed in systemd unit files
- Our base application simulator service, with no control group attributes yet.



```
linux-f6pd:/usr/lib/systemd/system # cat cpuhog.service
[Unit]
Description=MIPs Burner

[Service]
ExecStart=/root/cpuhog.sh
ExecReload=/usr/bin/kill -s SIGHUP $MAINPID
Restart=on-abort

linux-f6pd:/usr/lib/systemd/system # 
```

Our running systemd service

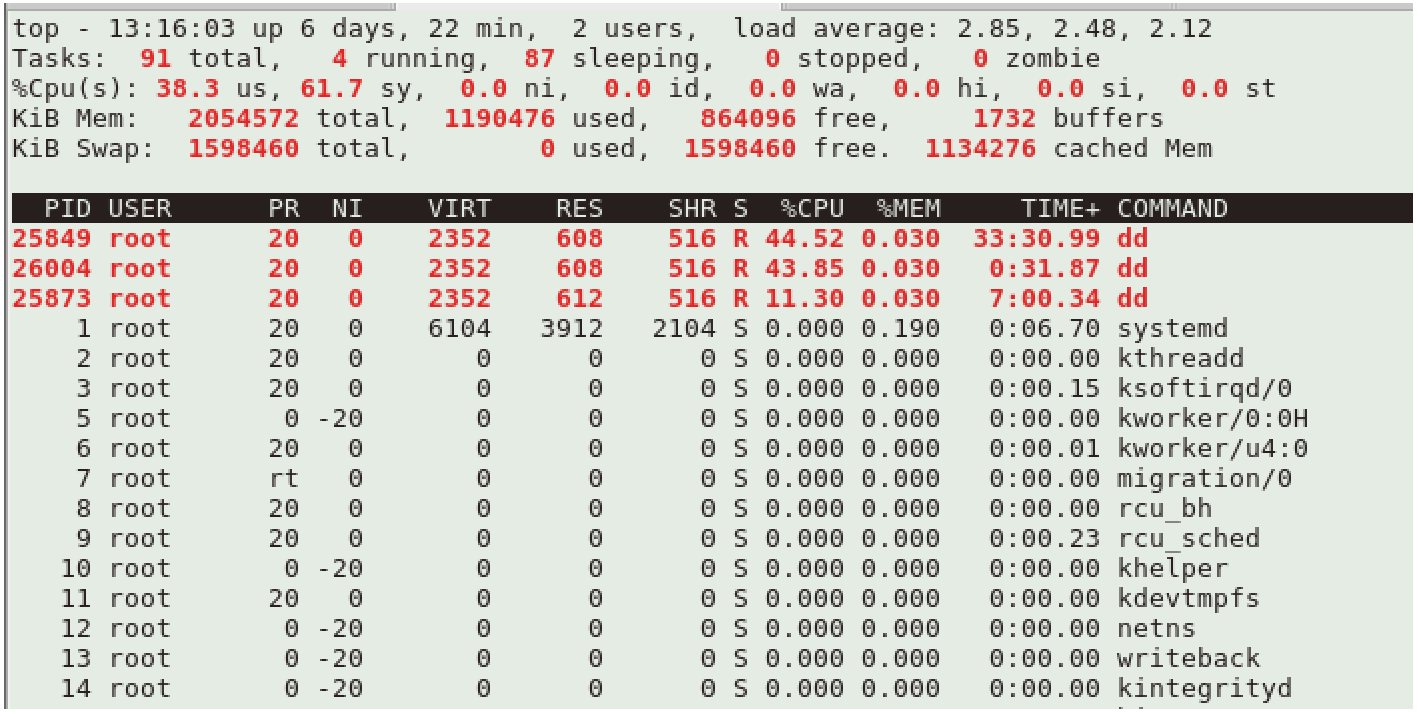
- Default system slice assignment shown



```
linux-f6pd:/usr/lib/systemd/system # cat cpuhog.service
[Unit]
Description=MIPs Burner

[Service]
ExecStart=/root/cpuhog.sh
ExecReload=/usr/bin/kill -s SIGHUP $MAINPID
Restart=on-abort

linux-f6pd:/usr/lib/systemd/system # systemctl start cpuhog.service
linux-f6pd:/usr/lib/systemd/system # systemctl status  cpuhog.service
cpuhog.service - MIPs Burner
   Loaded: loaded (/usr/lib/systemd/system/cpuhog.service; static)
   Active: active (running) since Thu 2015-07-23 13:14:51 EDT; 7s ago
 Main PID: 26003 (bash)
   CGroup: /system.slice/cpuhog.service
           ├─26003 bash /root/cpuhog.sh
           └─26004 dd if=/dev/zero of=/dev/null

Jul 23 13:14:51 linux-f6pd systemd[1]: Starting MIPs Burner...
Jul 23 13:14:51 linux-f6pd systemd[1]: Started MIPs Burner.
linux-f6pd:/usr/lib/systemd/system #
```

Three system services

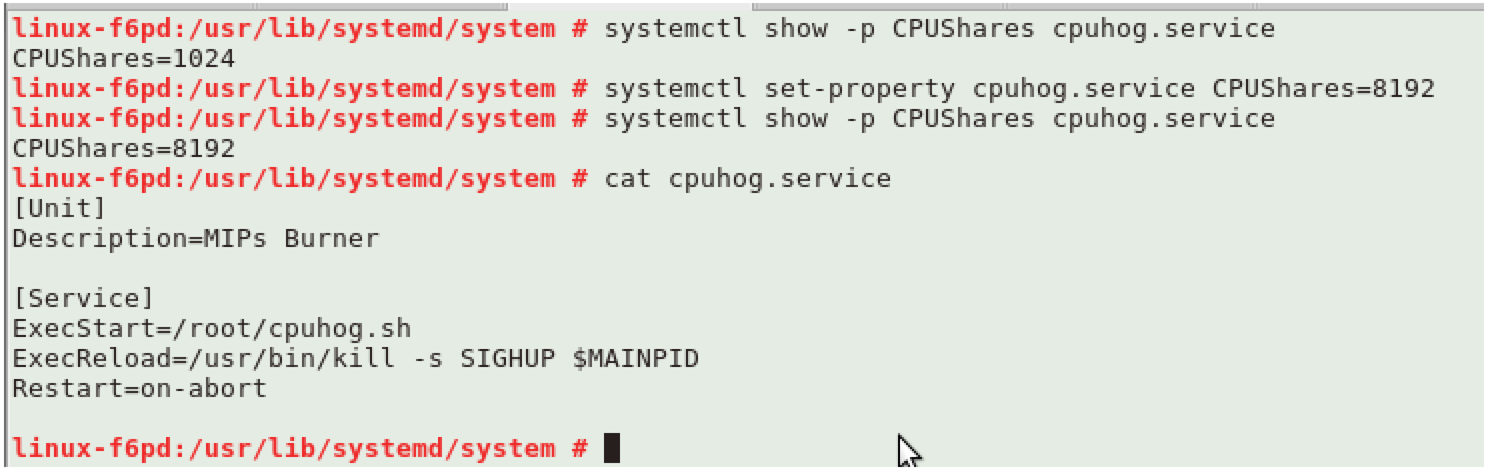
- Shares of 1024,1024, and 256



```
top - 13:16:03 up 6 days, 22 min,  2 users,  load average: 2.85, 2.48, 2.12
Tasks:  91 total,   4 running,  87 sleeping,   0 stopped,   0 zombie
%Cpu(s): 38.3 us, 61.7 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2054572 total,  1190476 used,   864096 free,     1732 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1134276 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
25849 root      20   0    2352    608    516 R 44.52 0.030  33:30.99 dd
26004 root      20   0    2352    608    516 R 43.85 0.030   0:31.87 dd
25873 root      20   0    2352    612    516 R 11.30 0.030   7:00.34 dd
    1 root      20   0    6104   3912   2104 S 0.000 0.190   0:06.70 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.15 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.23 rcu_sched
   10 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 khelper
   11 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 netns
   13 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 writeback
   14 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kintegrityd
```

Systemctl share adjustment

- Adjust from default of 1024 to 8192 via set-property

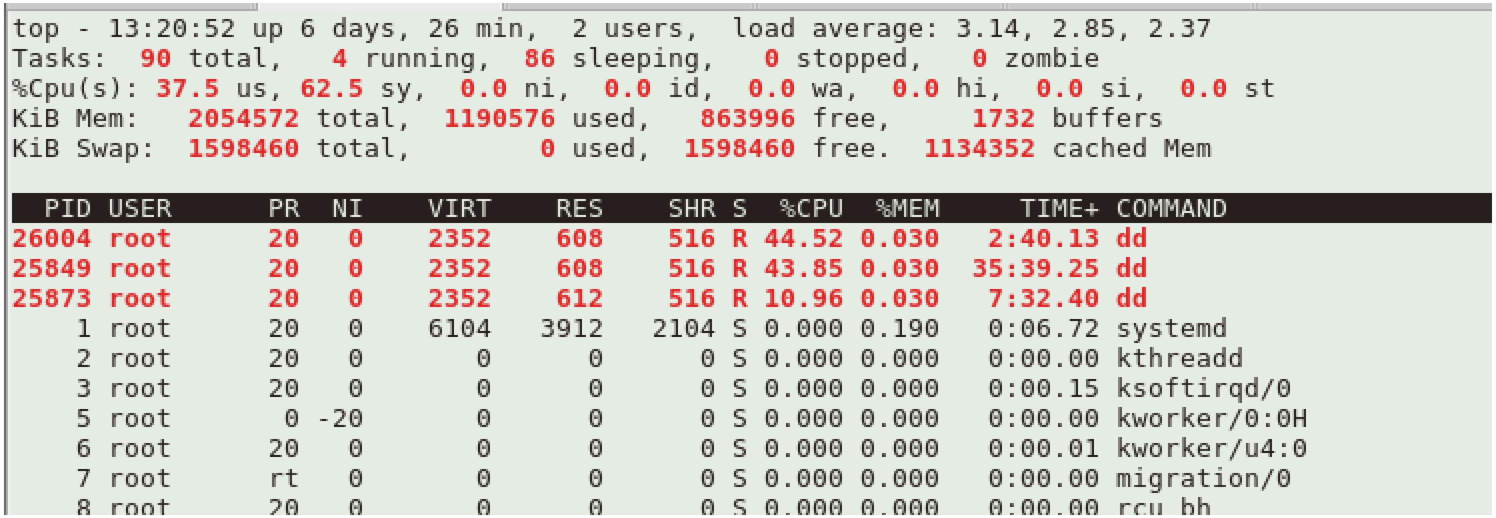


```
linux-f6pd:/usr/lib/systemd/system # systemctl show -p CPUShares cpuhog.service
CPUShares=1024
linux-f6pd:/usr/lib/systemd/system # systemctl set-property cpuhog.service CPUShares=8192
linux-f6pd:/usr/lib/systemd/system # systemctl show -p CPUShares cpuhog.service
CPUShares=8192
linux-f6pd:/usr/lib/systemd/system # cat cpuhog.service
[Unit]
Description=MIPs Burner

[Service]
ExecStart=/root/cpuhog.sh
ExecReload=/usr/bin/kill -s SIGHUP $MAINPID
Restart=on-abort

linux-f6pd:/usr/lib/systemd/system #
```

- The property was set for CPUShares=8192, where does that reside?

CPUShares

- 8192, 1024, 256 ?
- The change did NOT dynamically take effect



```
top - 13:20:52 up 6 days, 26 min,  2 users,  load average: 3.14, 2.85, 2.37
Tasks:  90 total,   4 running,  86 sleeping,   0 stopped,   0 zombie
%Cpu(s): 37.5 us, 62.5 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2054572 total,  1190576 used,   863996 free,     1732 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1134352 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
26004 root      20   0    2352    608    516 R 44.52 0.030   2:40.13 dd
25849 root      20   0    2352    608    516 R 43.85 0.030  35:39.25 dd
25873 root      20   0    2352    612    516 R 10.96 0.030   7:32.40 dd
    1 root      20   0    6104   3912   2104 S 0.000 0.190   0:06.72 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.15 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
```

Cgroups pseudo filesystem

- Four different locations on the system exist related to our “cpuhog” application service

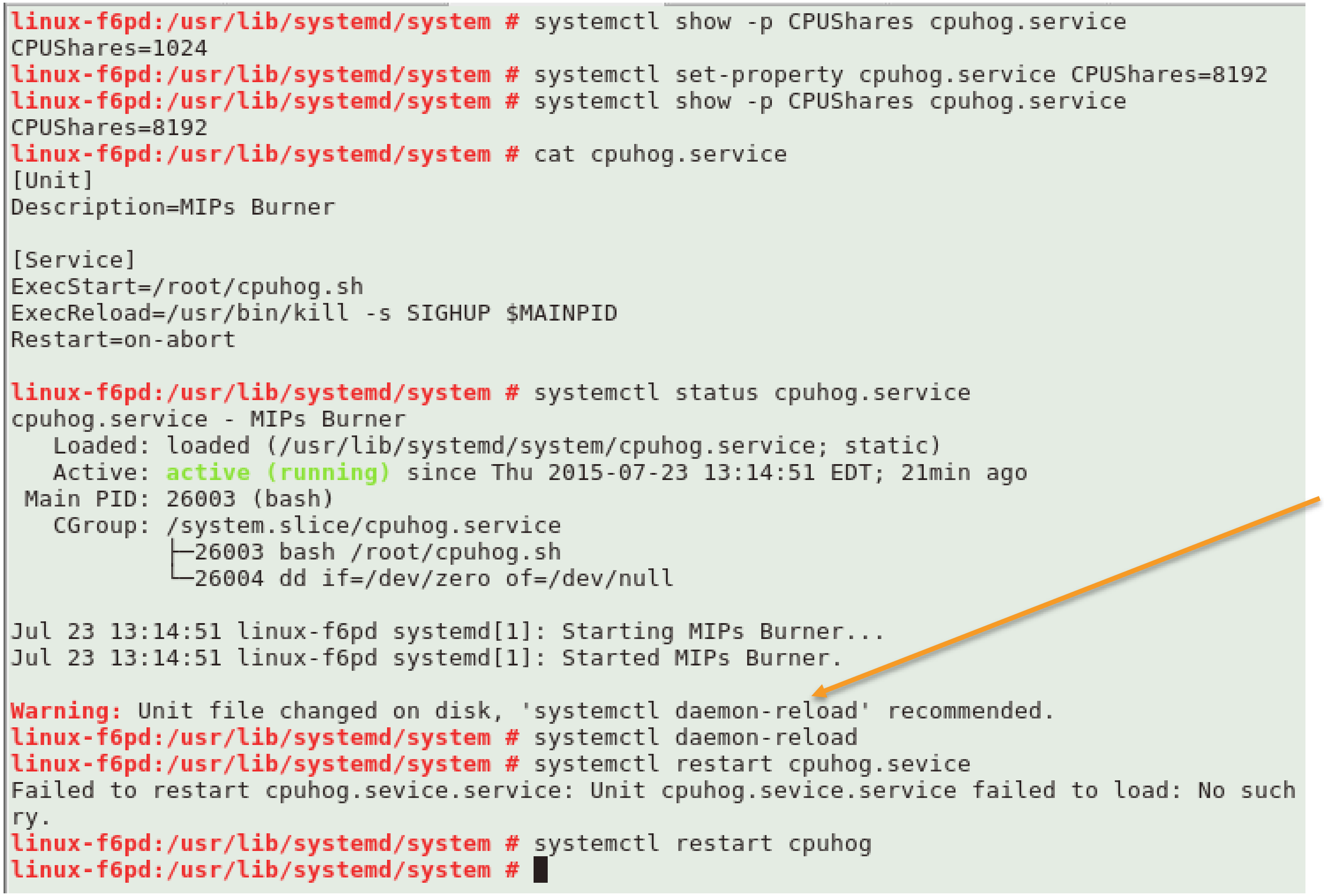
```
linux-f6pd:~ # find / -name "*cpuhog*‟
/etc/systemd/system/cpuhog.service.d
/sys/fs/cgroup/cpu,cpuacct/system.slice/cpuhog.service
/sys/fs/cgroup/systemd/system.slice/cpuhog.service
/usr/lib/systemd/system/cpuhog.service/root/cpuhog.sh
linux-f6pd:~ #
```

CPUShares.conf systemd file

- Our systemctl set-property for CPUShares resulted in a specific configuration file being created under /etc/system/system/<<service>>.service.d

```
linux-f6pd:/etc/systemd/system/cpuhog.service.d # ls
90-CPUShares.conf

linux-f6pd:/etc/systemd/system/cpuhog.service.d # cat 90-CPUShares.conf
[Service]
CPUShares=8192
```

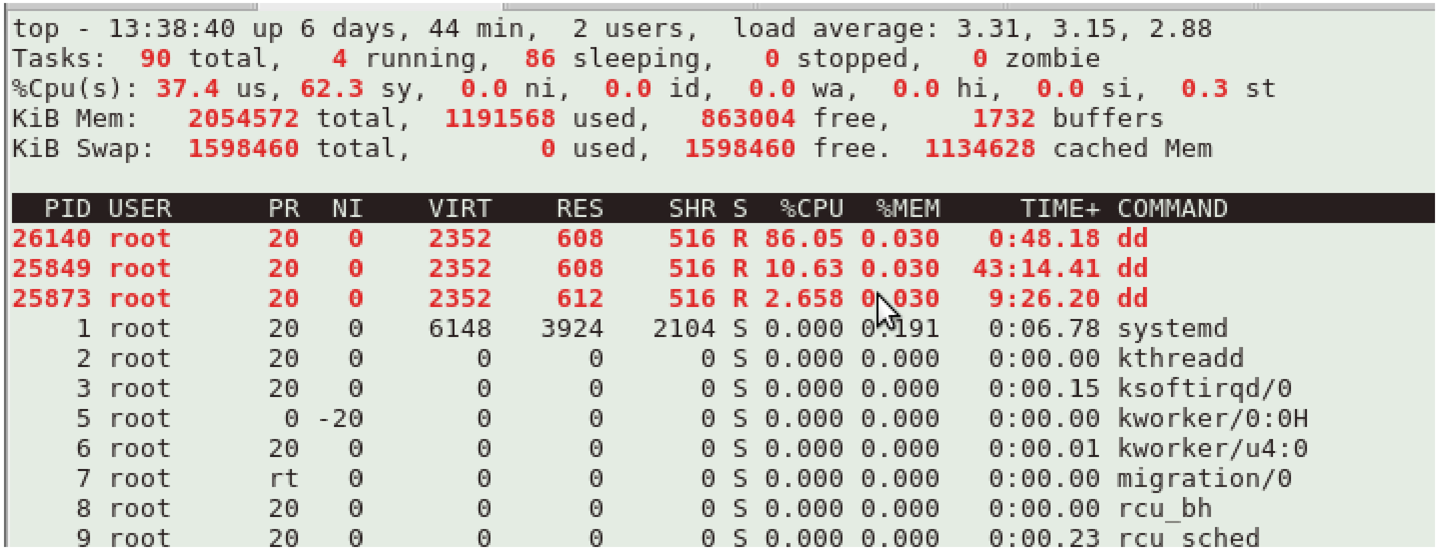
Systemctl and CPUShares



```
linux-f6pd:/usr/lib/systemd/system # systemctl show -p CPUShares cpuhog.service
CPUShares=1024
linux-f6pd:/usr/lib/systemd/system # systemctl set-property cpuhog.service CPUShares=8192
linux-f6pd:/usr/lib/systemd/system # systemctl show -p CPUShares cpuhog.service
CPUShares=8192
linux-f6pd:/usr/lib/systemd/system # cat cpuhog.service
[Unit]
Description=MIPs Burner

[Service]
ExecStart=/root/cpuhog.sh
ExecReload=/usr/bin/kill -s SIGHUP $MAINPID
Restart=on-abort

linux-f6pd:/usr/lib/systemd/system # systemctl status cpuhog.service
cpuhog.service - MIPs Burner
   Loaded: loaded (/usr/lib/systemd/system/cpuhog.service; static)
   Active: active (running) since Thu 2015-07-23 13:14:51 EDT; 21min ago
 Main PID: 26003 (bash)
   CGroup: /system.slice/cpuhog.service
           ├─26003 bash /root/cpuhog.sh
           └─26004 dd if=/dev/zero of=/dev/null

Jul 23 13:14:51 linux-f6pd systemd[1]: Starting MIPs Burner...
Jul 23 13:14:51 linux-f6pd systemd[1]: Started MIPs Burner.

Warning: Unit file changed on disk, 'systemctl daemon-reload' recommended.
linux-f6pd:/usr/lib/systemd/system # systemctl daemon-reload
linux-f6pd:/usr/lib/systemd/system # systemctl restart cpuhog.sevice
Failed to restart cpuhog.sevice.service: Unit cpuhog.sevice.service failed to load: No such
ry.
linux-f6pd:/usr/lib/systemd/system # systemctl restart cpuhog
linux-f6pd:/usr/lib/systemd/system # 
```

CPUShares

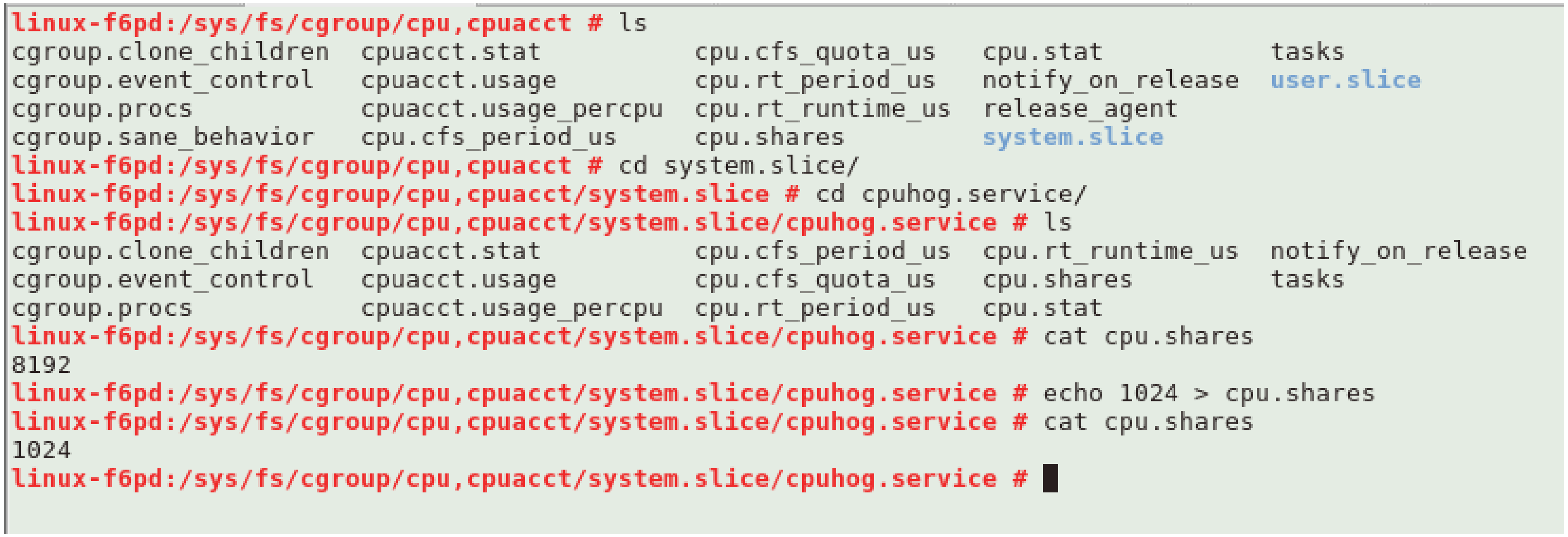
- Shares of 8192, 1024, and 256



```
top - 13:38:40 up 6 days, 44 min,  2 users,  load average: 3.31, 3.15, 2.88
Tasks:  90 total,   4 running,  86 sleeping,   0 stopped,   0 zombie
%Cpu(s): 37.4 us, 62.3 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.3 st
KiB Mem:   2054572 total,  1191568 used,   863004 free,     1732 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1134628 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
26140 root      20   0    2352    608    516 R 86.05 0.030   0:48.18 dd
25849 root      20   0    2352    608    516 R 10.63 0.030  43:14.41 dd
25873 root      20   0    2352    612    516 R 2.658 0.030   9:26.20 dd
    1 root      20   0    6148   3924   2104 S 0.000 0.191   0:06.78 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.15 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.23 rcu_sched
```

Control group queries and manipulation

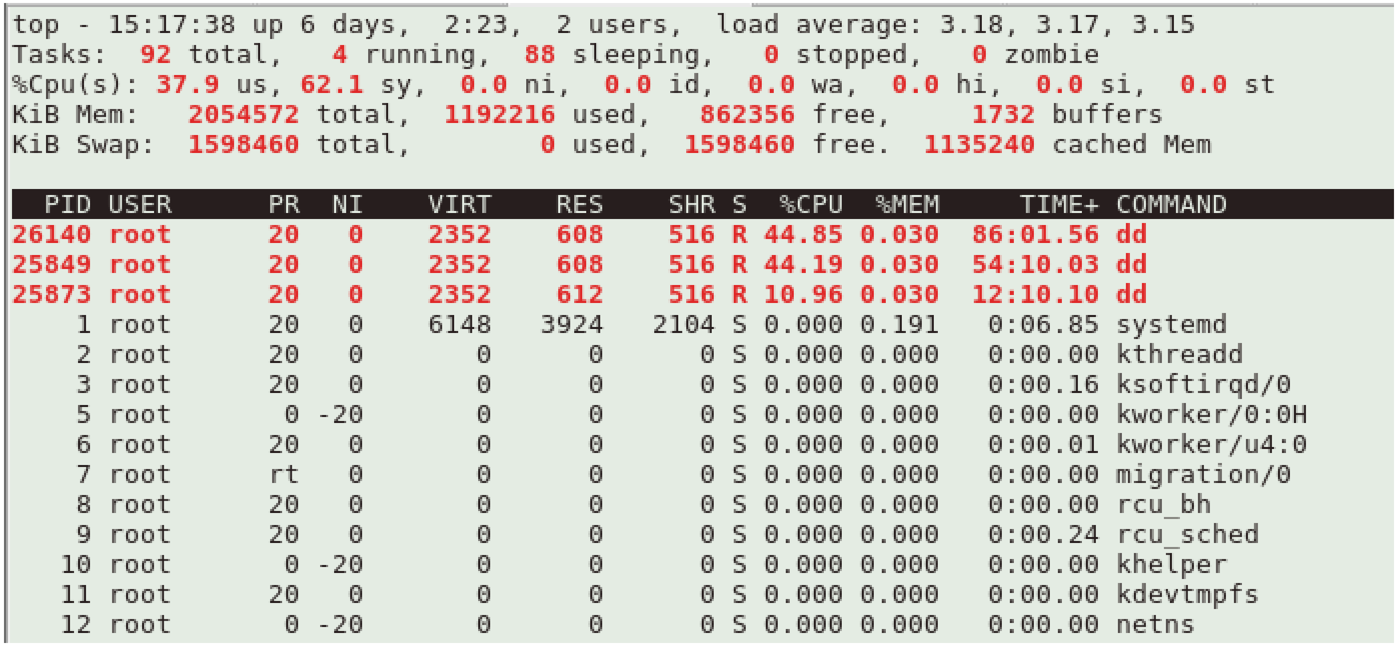
- Via systemctl unit files and commands
- Via libcgroup commands
- Manually via sys/fs
  - Reset cpu.shares from 8192 to 1024



```
linux-f6pd:/sys/fs/cgroup/cpu,cpuacct # ls
cgroup.clone_children  cpuacct.stat          cpu.cfs_quota_us   cpu.stat           tasks
cgroup.event_control   cpuacct.usage         cpu.rt_period_us   notify_on_release  user.slice
cgroup.procs           cpuacct.usage_percpu  cpu.rt_runtime_us  release_agent
cgroup.sane_behavior   cpu.cfs_period_us     cpu.shares         system.slice
linux-f6pd:/sys/fs/cgroup/cpu,cpuacct # cd system.slice/
linux-f6pd:/sys/fs/cgroup/cpu,cpuacct/system.slice # cd cpuhog.service/
linux-f6pd:/sys/fs/cgroup/cpu,cpuacct/system.slice/cpuhog.service # ls
cgroup.clone_children  cpuacct.stat          cpu.cfs_period_us  cpu.rt_runtime_us  notify_on_release
cgroup.event_control   cpuacct.usage         cpu.cfs_quota_us   cpu.shares         tasks
cgroup.procs           cpuacct.usage_percpu  cpu.rt_period_us   cpu.stat
linux-f6pd:/sys/fs/cgroup/cpu,cpuacct/system.slice/cpuhog.service # cat cpu.shares
8192
linux-f6pd:/sys/fs/cgroup/cpu,cpuacct/system.slice/cpuhog.service # echo 1024 > cpu.shares
linux-f6pd:/sys/fs/cgroup/cpu,cpuacct/system.slice/cpuhog.service # cat cpu.shares
1024
linux-f6pd:/sys/fs/cgroup/cpu,cpuacct/system.slice/cpuhog.service # █
```

CPUShares

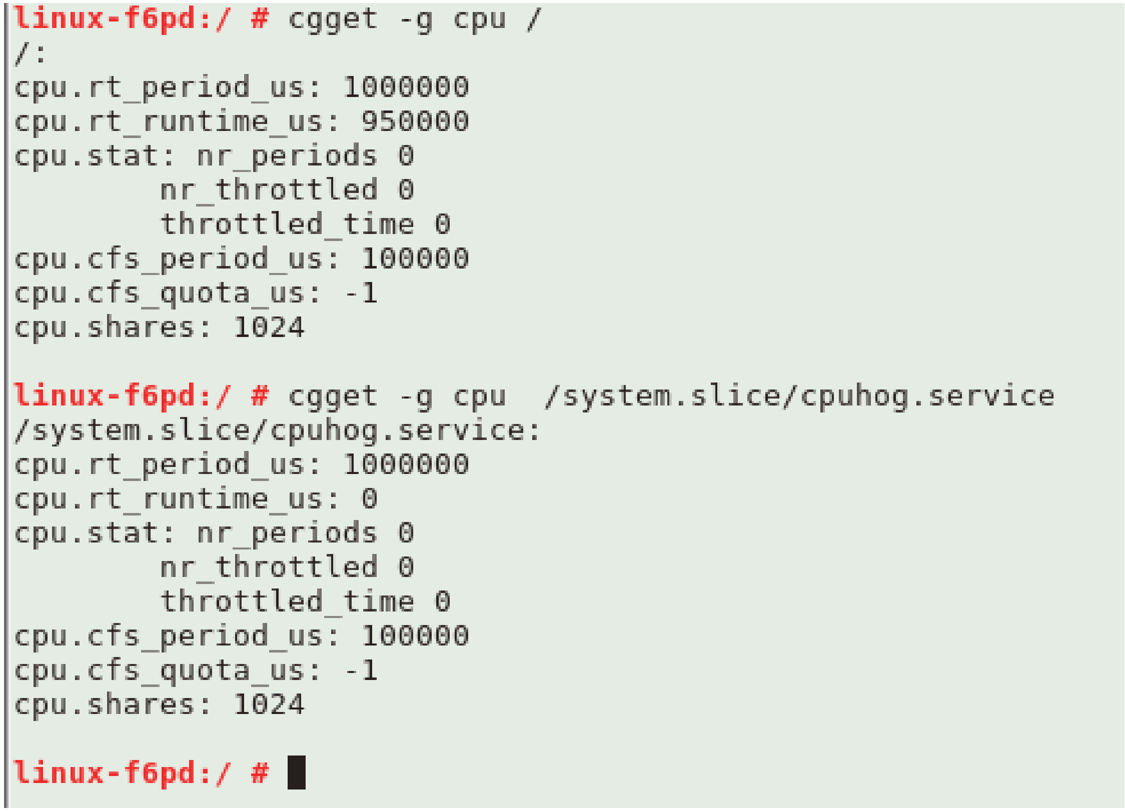
- Share adjusted manually



```
top - 15:17:38 up 6 days,  2:23,  2 users,  load average: 3.18, 3.17, 3.15
Tasks:  92 total,   4 running,  88 sleeping,   0 stopped,   0 zombie
%Cpu(s): 37.9 us, 62.1 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2054572 total,  1192216 used,   862356 free,     1732 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1135240 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
26140 root      20   0    2352    608    516 R 44.85 0.030  86:01.56 dd
25849 root      20   0    2352    608    516 R 44.19 0.030  54:10.03 dd
25873 root      20   0    2352    612    516 R 10.96 0.030  12:10.10 dd
    1 root      20   0    6148   3924   2104 S 0.000 0.191   0:06.85 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.16 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.24 rcu_sched
   10 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 khelper
   11 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 netns
```

CPU and cgget

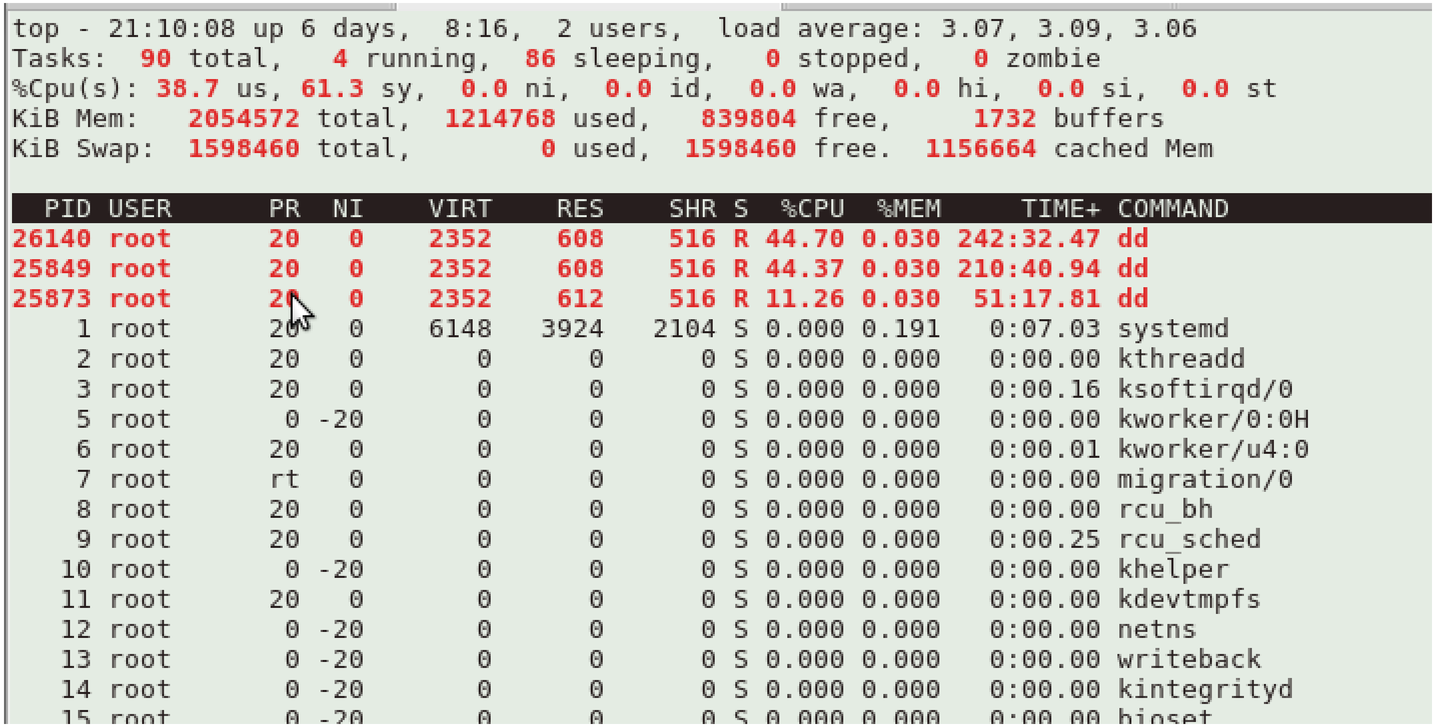
- Default CPUShare of 1024 and no quota set



```
linux-f6pd:/ # cgget -g cpu /
/:
cpu.rt_period_us: 1000000
cpu.rt_runtime_us: 950000
cpu.stat: nr_periods 0
        nr_throttled 0
        throttled_time 0
cpu.cfs_period_us: 100000
cpu.cfs_quota_us: -1
cpu.shares: 1024

linux-f6pd:/ # cgget -g cpu  /system.slice/cpuhog.service
/system.slice/cpuhog.service:
cpu.rt_period_us: 1000000
cpu.rt_runtime_us: 0
cpu.stat: nr_periods 0
        nr_throttled 0
        throttled_time 0
cpu.cfs_period_us: 100000
cpu.cfs_quota_us: -1
cpu.shares: 1024

linux-f6pd:/ # 
```

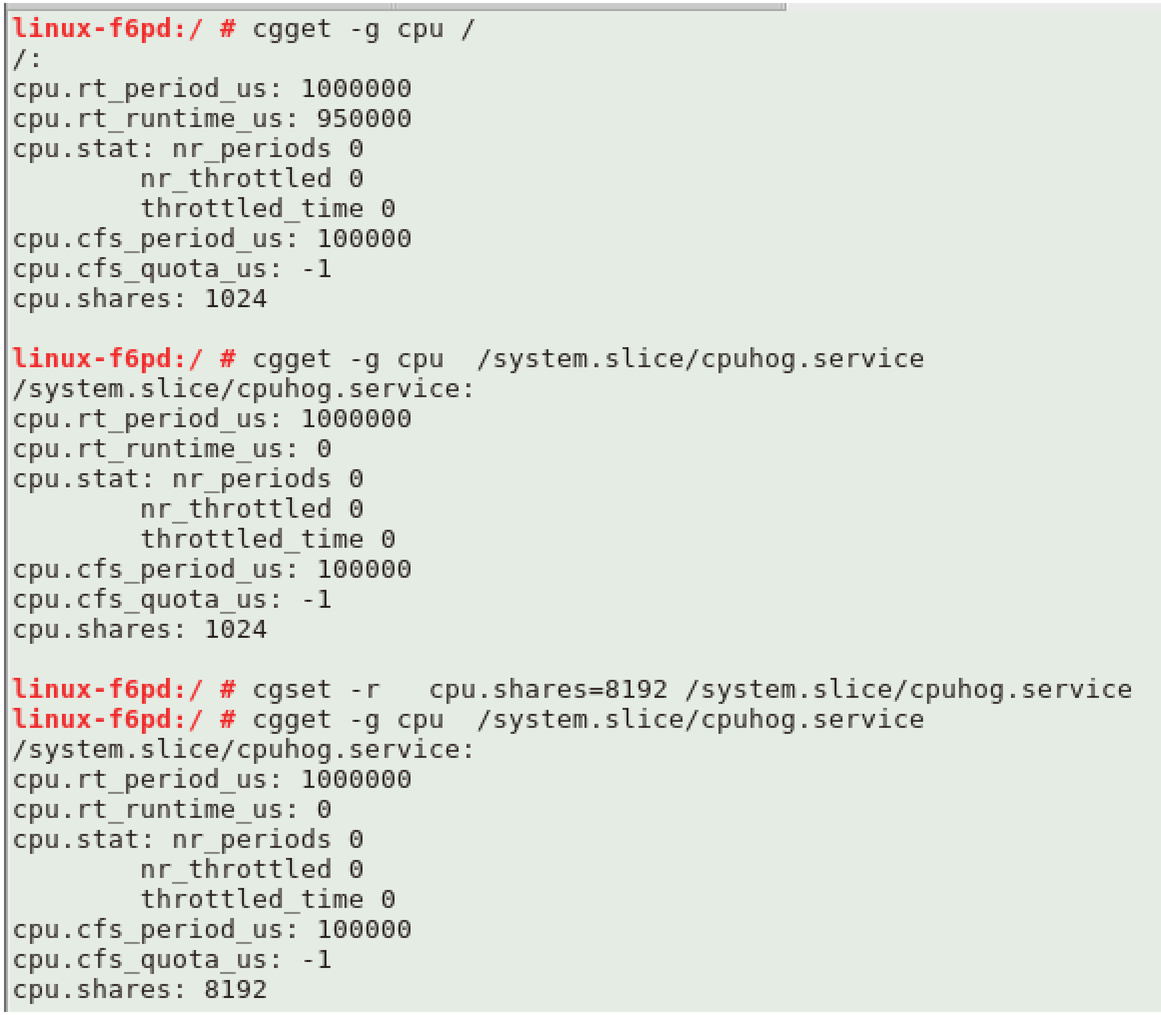
CPUShares

- Baseline - Shares of 1024, 1024, 256



```
top - 21:10:08 up 6 days,  8:16,  2 users,  load average: 3.07, 3.09, 3.06
Tasks:  90 total,   4 running,  86 sleeping,   0 stopped,   0 zombie
%Cpu(s): 38.7 us, 61.3 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2054572 total,  1214768 used,   839804 free,     1732 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1156664 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
26140 root      20   0    2352    608    516 R 44.70 0.030 242:32.47 dd
25849 root      20   0    2352    608    516 R 44.37 0.030 210:40.94 dd
25873 root      20   0    2352    612    516 R 11.26 0.030  51:17.81 dd
    1 root      20   0    6148   3924   2104 S 0.000 0.191   0:07.03 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.16 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.25 rcu_sched
   10 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 khelper
   11 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 netns
   13 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 writeback
   14 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kintegrityd
   15 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 bioset
```

libcgroup - cpuget/cpuset

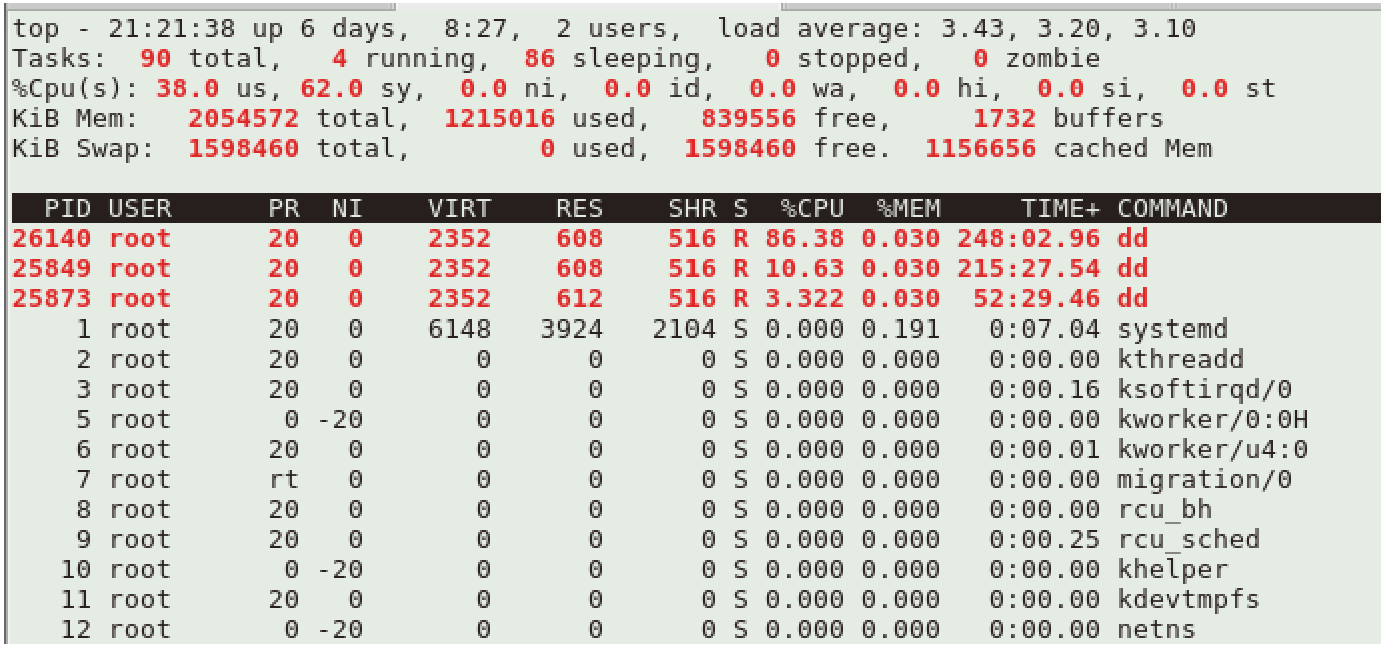


```
linux-f6pd:/ # cgget -g cpu /
/:
cpu.rt_period_us: 1000000
cpu.rt_runtime_us: 950000
cpu.stat: nr_periods 0
        nr_throttled 0
        throttled_time 0
cpu.cfs_period_us: 100000
cpu.cfs_quota_us: -1
cpu.shares: 1024

linux-f6pd:/ # cgget -g cpu  /system.slice/cpuhog.service
/system.slice/cpuhog.service:
cpu.rt_period_us: 1000000
cpu.rt_runtime_us: 0
cpu.stat: nr_periods 0
        nr_throttled 0
        throttled_time 0
cpu.cfs_period_us: 100000
cpu.cfs_quota_us: -1
cpu.shares: 1024

linux-f6pd:/ # cgset -r   cpu.shares=8192 /system.slice/cpuhog.service
linux-f6pd:/ # cgget -g cpu  /system.slice/cpuhog.service
/system.slice/cpuhog.service:
cpu.rt_period_us: 1000000
cpu.rt_runtime_us: 0
cpu.stat: nr_periods 0
        nr_throttled 0
        throttled_time 0
cpu.cfs_period_us: 100000
cpu.cfs_quota_us: -1
cpu.shares: 8192
```

- Alter cpu.share via cgset

CPUShare

- Share adjustment impact immediate, but not persistent



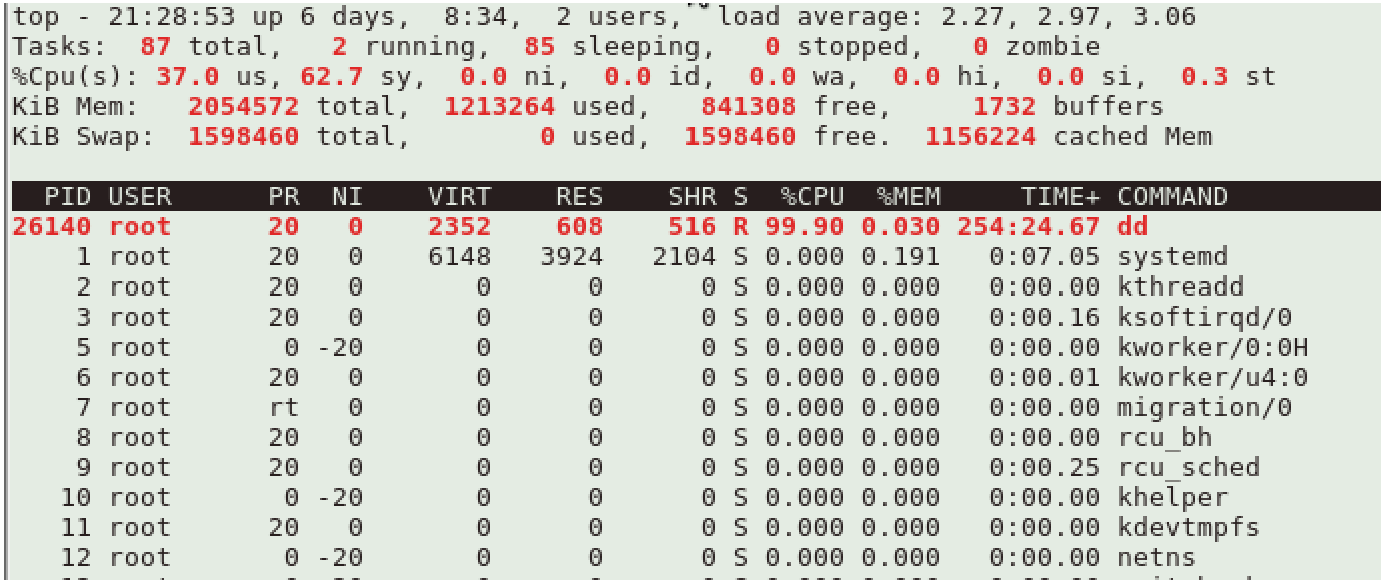
```
top - 21:21:38 up 6 days,  8:27,  2 users,  load average: 3.43, 3.20, 3.10
Tasks:  90 total,   4 running,  86 sleeping,   0 stopped,   0 zombie
%Cpu(s): 38.0 us, 62.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2054572 total,  1215016 used,   839556 free,     1732 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1156656 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
26140 root      20   0    2352    608    516 R 86.38 0.030 248:02.96 dd
25849 root      20   0    2352    608    516 R 10.63 0.030 215:27.54 dd
25873 root      20   0    2352    612    516 R 3.322 0.030  52:29.46 dd
    1 root      20   0    6148   3924   2104 S 0.000 0.191   0:07.04 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.16 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.25 rcu_sched
   10 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 khelper
   11 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 netns
```

CPU - Period & Quota

- **CPU Period**
  - Is the period of time in microseconds that CPU resources are allocated or evaluated
- **CPU Quota**
  - The amount of CPU time in microsecond the control group is allowed to consume before it is throttled
  - A value of -1 means no time restriction
- **cpu.stat**
  - contains information about how many intervals have occurred, how many times throttling of CPU has occurred and the amount of CPU time throttled.

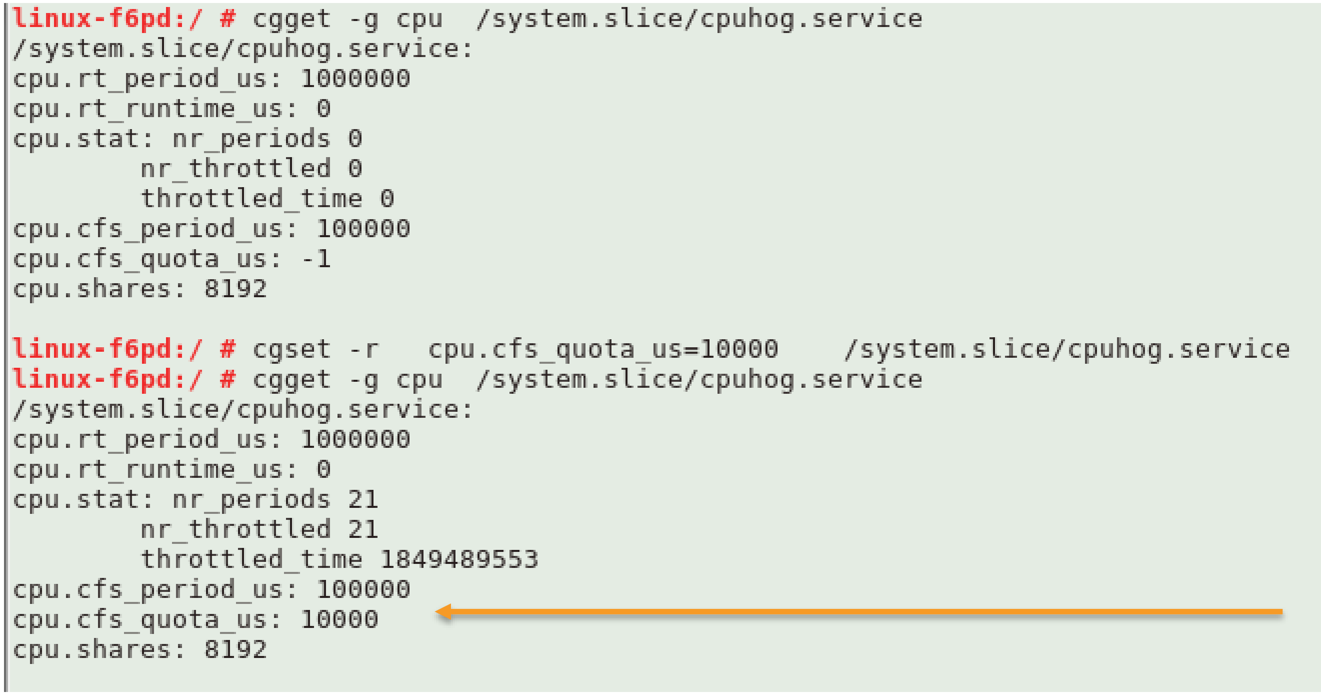
CPU Period and Quota

- Running at ~100%



```
top - 21:28:53 up 6 days,  8:34,  2 users,  load average: 2.27, 2.97, 3.06
Tasks:  87 total,   2 running,  85 sleeping,   0 stopped,   0 zombie
%Cpu(s): 37.0 us, 62.7 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.3 st
KiB Mem:   2054572 total,  1213264 used,   841308 free,     1732 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1156224 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
26140 root      20   0    2352    608    516 R 99.90 0.030 254:24.67 dd
    1 root      20   0    6148   3924   2104 S 0.000 0.191   0:07.05 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.16 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.25 rcu_sched
   10 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 khelper
   11 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 netns
```

CPU Period and Quota

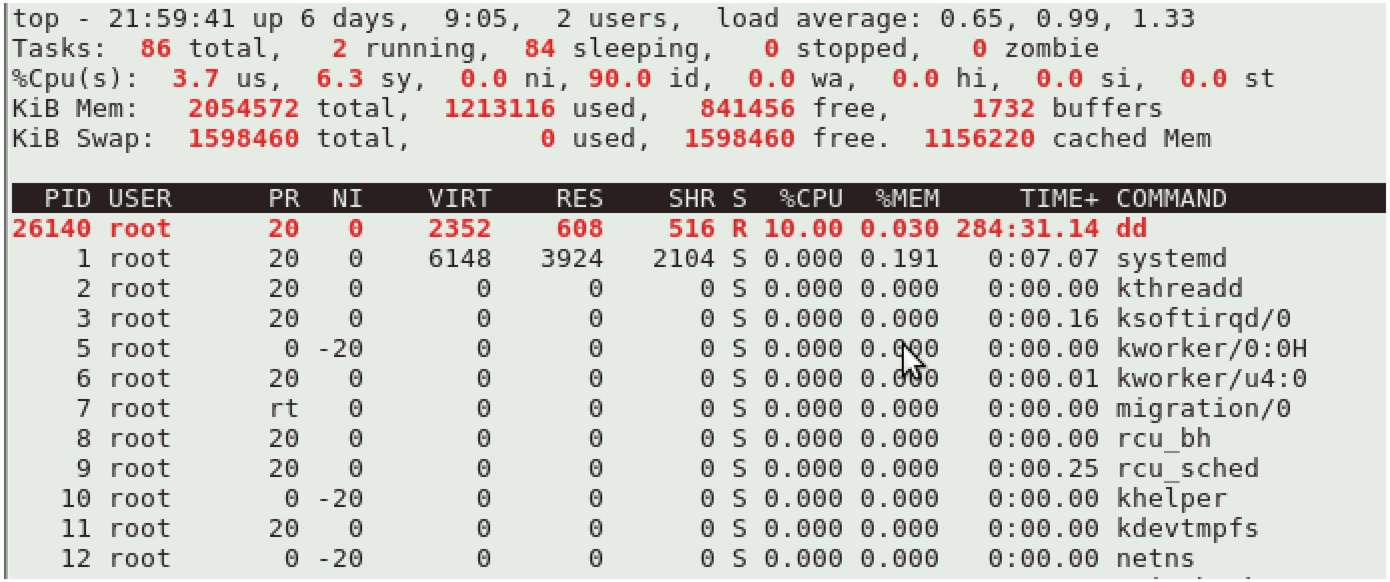


```
linux-f6pd:/ # cgget -g cpu  /system.slice/cpuhog.service
/system.slice/cpuhog.service:
cpu.rt_period_us: 1000000
cpu.rt_runtime_us: 0
cpu.stat: nr_periods 0
        nr_throttled 0
        throttled_time 0
cpu.cfs_period_us: 100000
cpu.cfs_quota_us: -1
cpu.shares: 8192

linux-f6pd:/ # cgset -r   cpu.cfs_quota_us=10000    /system.slice/cpuhog.service
linux-f6pd:/ # cgget -g cpu  /system.slice/cpuhog.service
/system.slice/cpuhog.service:
cpu.rt_period_us: 1000000
cpu.rt_runtime_us: 0
cpu.stat: nr_periods 21
        nr_throttled 21
        throttled_time 1849489553
cpu.cfs_period_us: 100000
cpu.cfs_quota_us: 10000
cpu.shares: 8192
```

- Quota is now 10% of period

CPU Period and Quota

- Using exactly 10%



```
top - 21:59:41 up 6 days,  9:05,  2 users,  load average: 0.65, 0.99, 1.33
Tasks:  86 total,   2 running,  84 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.7 us,  6.3 sy,  0.0 ni, 90.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2054572 total,  1213116 used,   841456 free,     1732 buffers
KiB Swap:  1598460 total,        0 used,  1598460 free.  1156220 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
26140 root      20   0    2352    608    516 R 10.00 0.030 284:31.14 dd
    1 root      20   0    6148   3924   2104 S 0.000 0.191   0:07.07 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.16 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.01 kworker/u4:0
    7 root      rt   0       0      0      0 S 0.000 0.000   0:00.00 migration/0
    8 root      20   0       0      0      0 S 0.000 0.000   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S 0.000 0.000   0:00.25 rcu_sched
   10 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 khelper
   11 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kdevtmpfs
   12 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 netns
```

Agenda

- Control groups overview
- Control groups what is new?
- Control groups configuration
- Assignment and display of cgroup
- CPU resource examples
- ➢ **Memory resource examples**
- Namespaces and containers

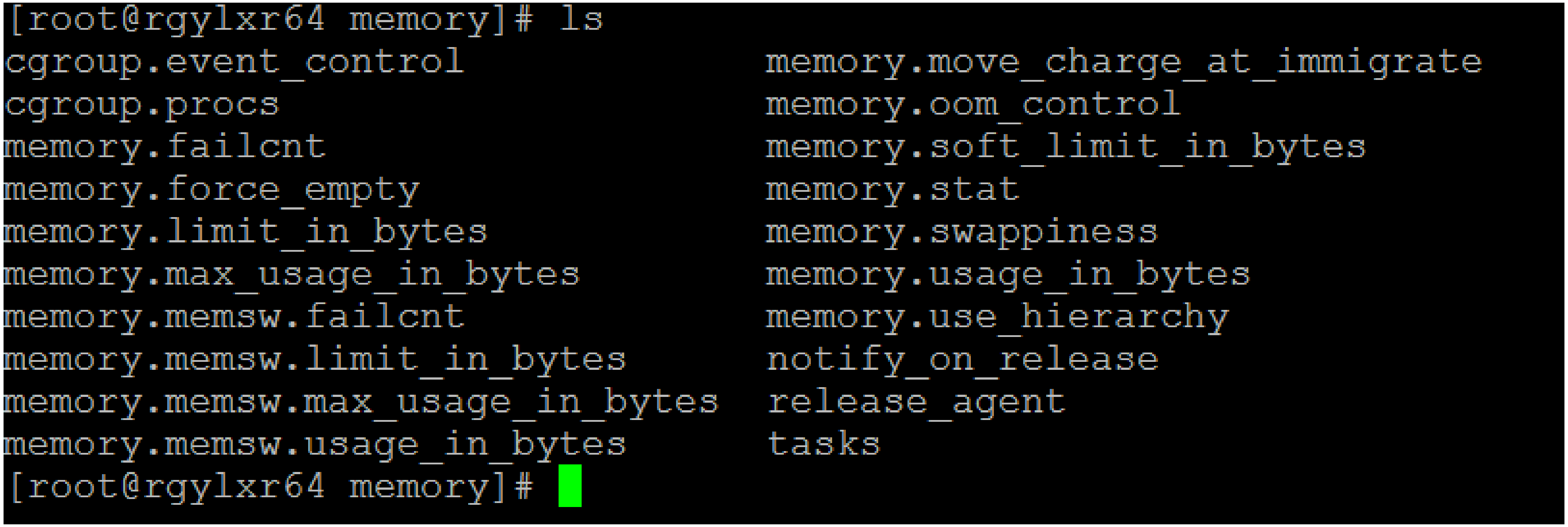
Memory resource

- With memory, each page has a cgroup “owner” assigned at allocation
- Features such as per group swapiness and out of memory management
  - Limit memory of user space process
  - Kernel space memory remains unlimited (well sort of)
  - Memory accounting in addition to controls

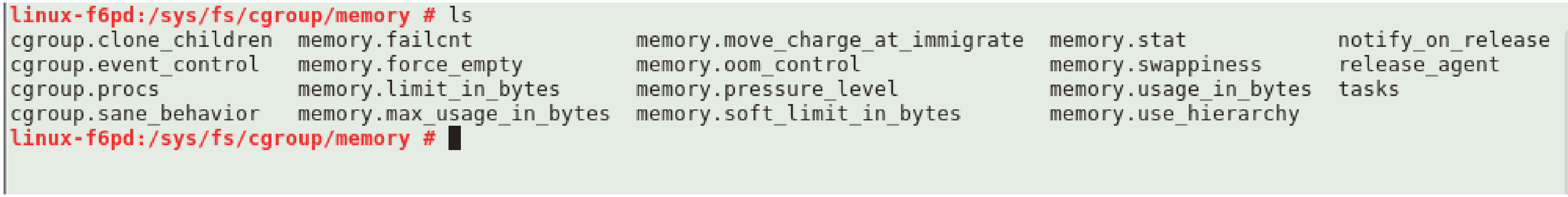
- Example shows initially ~1900MB of a 2GB virtual server consumed for filesystem cache as part of a simple dd command writing to a file
- Use cgroups to limit memory, specifically file system cache usage in this case.
- Can help to avoid system level out of memory condition by also limiting swap space usage.

Memory resource

- Like other resource controllers both accounting and control fields exist



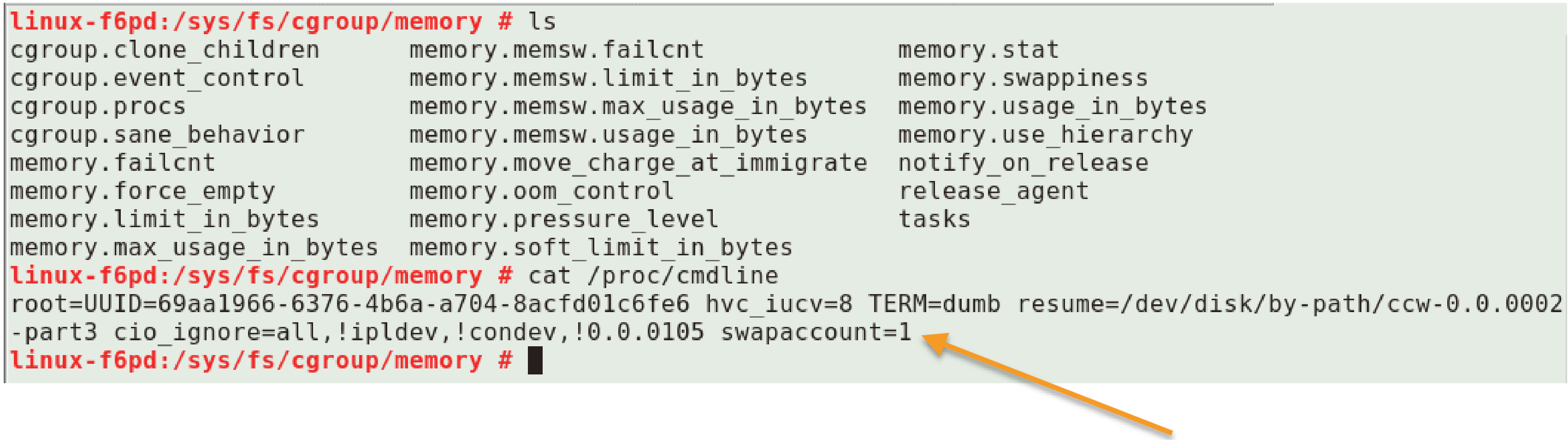
```
[root@rgylxr64 memory]# ls
cgroup.event_control              memory.move_charge_at_immigrate
cgroup.procs                      memory.oom_control
memory.failcnt                    memory.soft_limit_in_bytes
memory.force_empty                memory.stat
memory.limit_in_bytes             memory.swappiness
memory.max_usage_in_bytes         memory.usage_in_bytes
memory.memsw.failcnt              memory.use_hierarchy
memory.memsw.limit_in_bytes       notify_on_release
memory.memsw.max_usage_in_bytes   release_agent
memory.memsw.usage_in_bytes       tasks
[root@rgylxr64 memory]#
```



```
linux-f6pd:/sys/fs/cgroup/memory # ls
cgroup.clone_children  memory.failcnt             memory.move_charge_at_immigrate  memory.stat             notify_on_release
cgroup.event_control   memory.force_empty         memory.oom_control               memory.swappiness       release_agent
cgroup.procs           memory.limit_in_bytes      memory.pressure_level            memory.usage_in_bytes   tasks
cgroup.sane_behavior   memory.max_usage_in_bytes  memory.soft_limit_in_bytes       memory.use_hierarchy
linux-f6pd:/sys/fs/cgroup/memory #
```

Swap memory

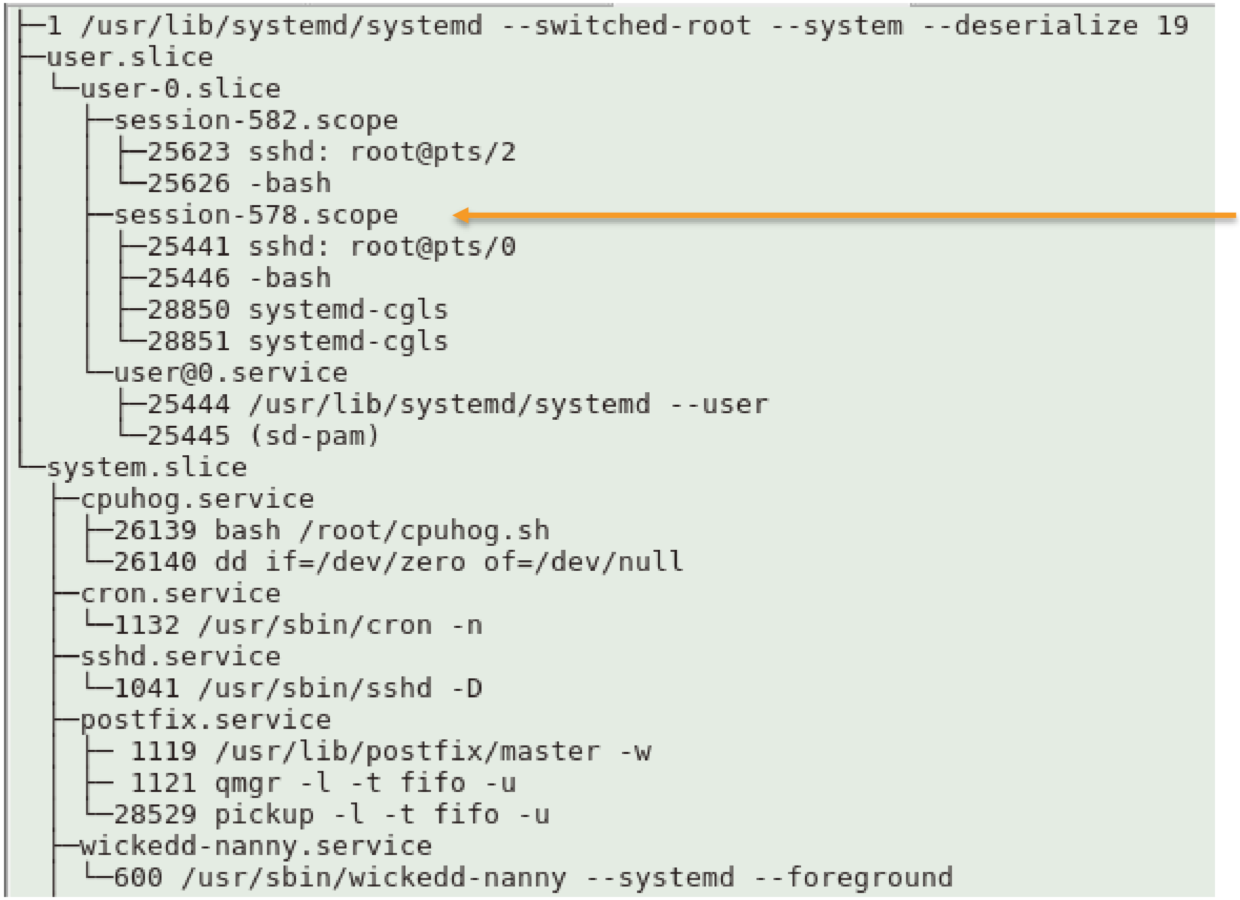
- **swapaccount** kernel parameter can enable/disable the memsw function in the cgroups memory controller



```
linux-f6pd:/sys/fs/cgroup/memory # ls
cgroup.clone_children      memory.memsw.failcnt              memory.stat
cgroup.event_control       memory.memsw.limit_in_bytes       memory.swappiness
cgroup.procs               memory.memsw.max_usage_in_bytes   memory.usage_in_bytes
cgroup.sane_behavior       memory.memsw.usage_in_bytes       memory.use_hierarchy
memory.failcnt             memory.move_charge_at_immigrate   notify_on_release
memory.force_empty         memory.oom_control                release_agent
memory.limit_in_bytes      memory.pressure_level             tasks
memory.max_usage_in_bytes  memory.soft_limit_in_bytes
linux-f6pd:/sys/fs/cgroup/memory # cat /proc/cmdline
root=UUID=69aa1966-6376-4b6a-a704-8acfd01c6fe6 hvc_iucv=8 TERM=dumb resume=/dev/disk/by-path/ccw-0.0.0002
-part3 cio_ignore=all,!ipldev,!condev,!0.0.0105 swapaccount=1
linux-f6pd:/sys/fs/cgroup/memory # 
```

Where is my shell running?

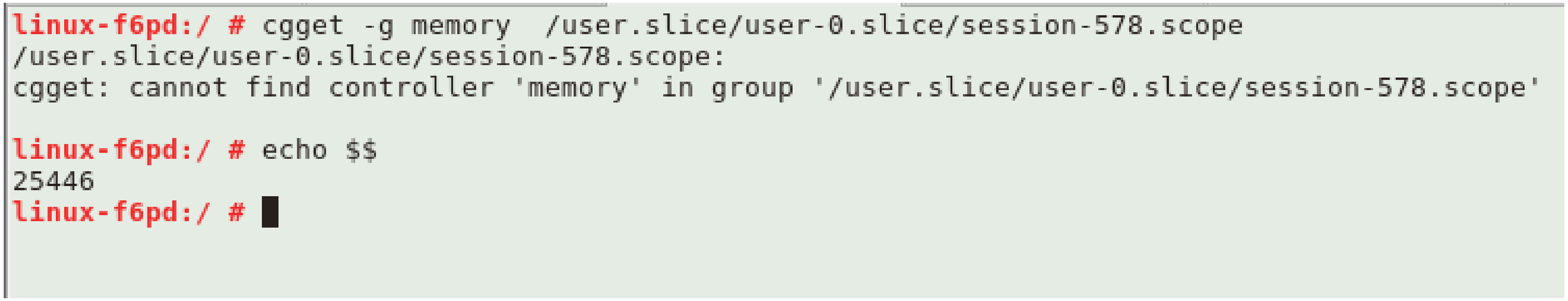
- systemd-cgls



```
├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 19
├─user.slice
│ └─user-0.slice
│   ├─session-582.scope
│   │ ├─25623 sshd: root@pts/2
│   │ └─25626 -bash
│   ├─session-578.scope
│   │ ├─25441 sshd: root@pts/0
│   │ ├─25446 -bash
│   │ ├─28850 systemd-cgls
│   │ └─28851 systemd-cgls
│   └─user@0.service
│     ├─25444 /usr/lib/systemd/systemd --user
│     └─25445 (sd-pam)
└─system.slice
  ├─cpuhog.service
  │ ├─26139 bash /root/cpuhog.sh
  │ └─26140 dd if=/dev/zero of=/dev/null
  ├─cron.service
  │ └─1132 /usr/sbin/cron -n
  ├─sshd.service
  │ └─1041 /usr/sbin/sshd -D
  ├─postfix.service
  │ ├─ 1119 /usr/lib/postfix/master -w
  │ ├─ 1121 qmgr -l -t fifo -u
  │ └─28529 pickup -l -t fifo -u
  ├─wickedd-nanny.service
  │ └─600 /usr/sbin/wickedd-nanny --systemd --foreground
```

Memory subsystem

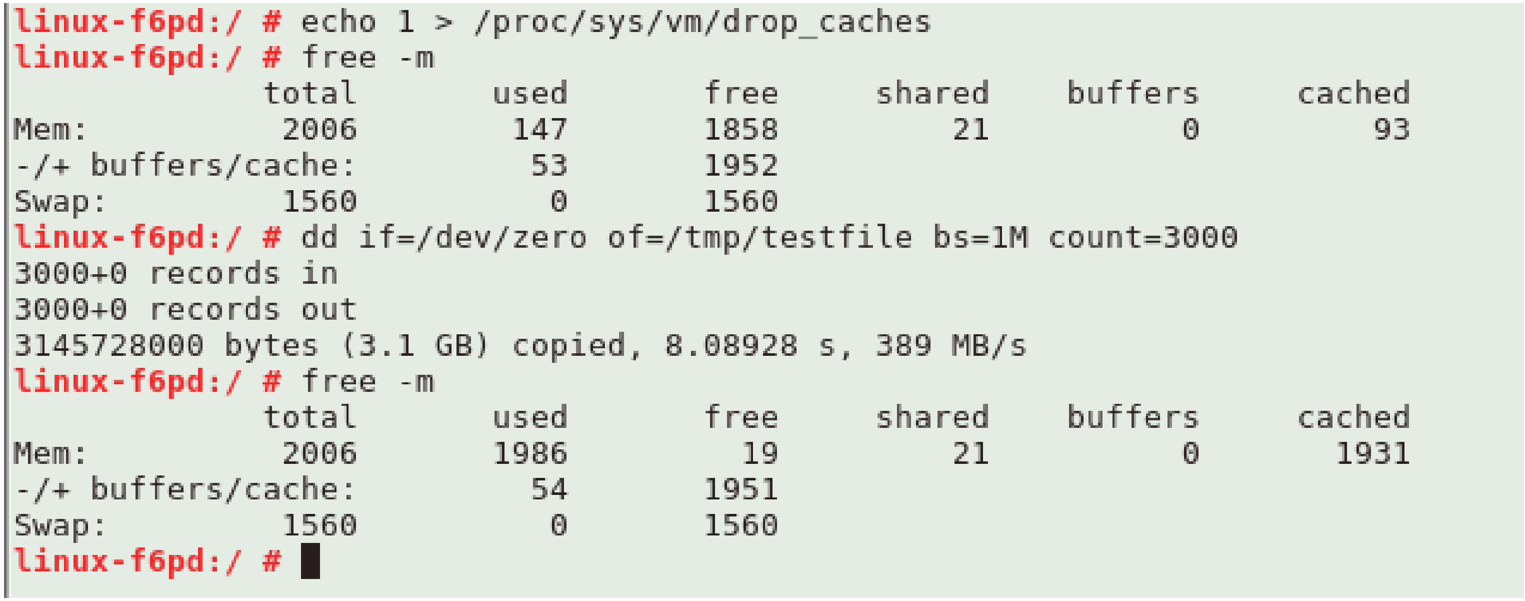
- Since nothing memory related has been set in systemd so far the controller or subsystem is not found yet.



```
linux-f6pd:/ # cgget -g memory  /user.slice/user-0.slice/session-578.scope
/user.slice/user-0.slice/session-578.scope:
cgget: cannot find controller 'memory' in group '/user.slice/user-0.slice/session-578.scope'

linux-f6pd:/ # echo $$
25446
linux-f6pd:/ # 
```

Memory subsystem

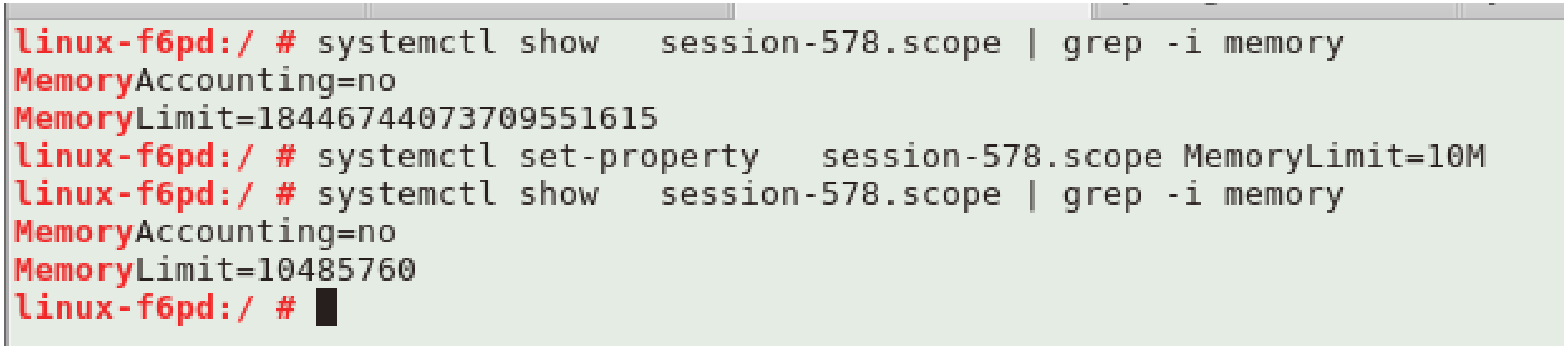
- Baseline view behavior
  - 1.9 GB of cache consumed by file copy



```
linux-f6pd:/ # echo 1 > /proc/sys/vm/drop_caches
linux-f6pd:/ # free -m
             total       used       free     shared    buffers     cached
Mem:          2006        147       1858         21          0         93
-/+ buffers/cache:         53       1952
Swap:         1560          0       1560
linux-f6pd:/ # dd if=/dev/zero of=/tmp/testfile bs=1M count=3000
3000+0 records in
3000+0 records out
3145728000 bytes (3.1 GB) copied, 8.08928 s, 389 MB/s
linux-f6pd:/ # free -m
             total       used       free     shared    buffers     cached
Mem:          2006       1986         19         21          0       1931
-/+ buffers/cache:         54       1951
Swap:         1560          0       1560
linux-f6pd:/ # 
```

Memory subsystem

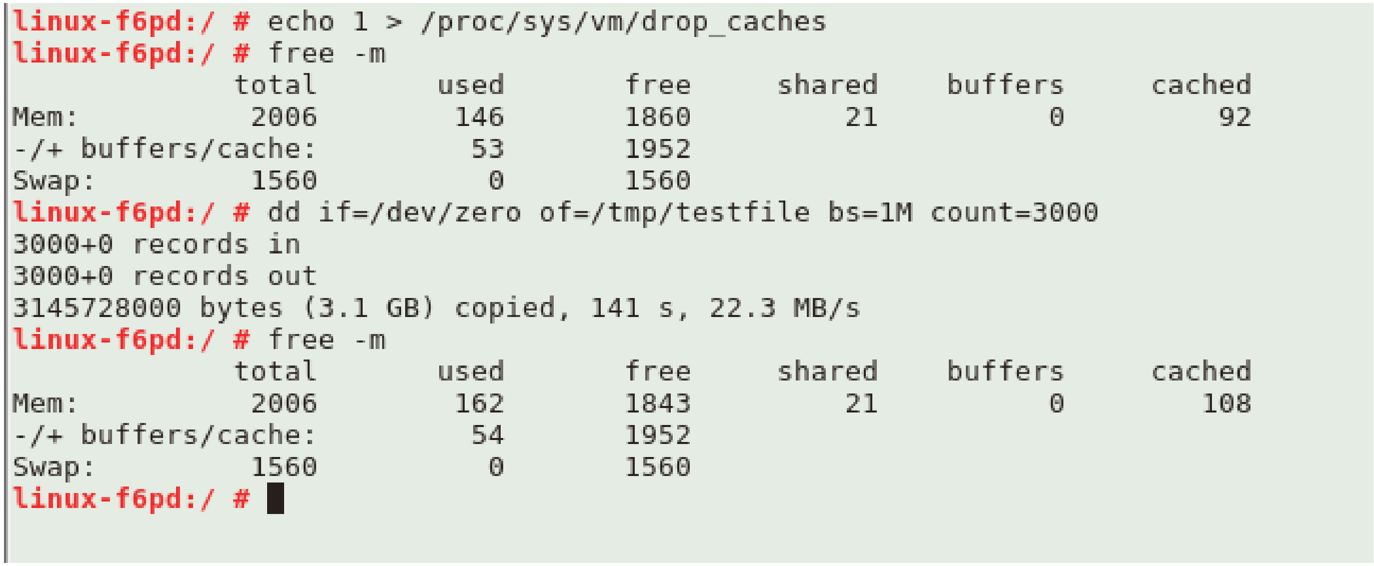
- 10 MB limit imposed on user session-578



```
linux-f6pd:/ # systemctl show   session-578.scope | grep -i memory
MemoryAccounting=no
MemoryLimit=18446744073709551615
linux-f6pd:/ # systemctl set-property   session-578.scope MemoryLimit=10M
linux-f6pd:/ # systemctl show   session-578.scope | grep -i memory
MemoryAccounting=no
MemoryLimit=10485760
linux-f6pd:/ #
```

Memory subsystem

- Memory usage (including filesystem cache) limited by cgroups for session 578



```
linux-f6pd:/ # echo 1 > /proc/sys/vm/drop_caches
linux-f6pd:/ # free -m
             total       used       free     shared    buffers     cached
Mem:          2006        146       1860         21          0         92
-/+ buffers/cache:         53       1952
Swap:         1560          0       1560
linux-f6pd:/ # dd if=/dev/zero of=/tmp/testfile bs=1M count=3000
3000+0 records in
3000+0 records out
3145728000 bytes (3.1 GB) copied, 141 s, 22.3 MB/s
linux-f6pd:/ # free -m
             total       used       free     shared    buffers     cached
Mem:          2006        162       1843         21          0        108
-/+ buffers/cache:         54       1952
Swap:         1560          0       1560
linux-f6pd:/ #
```

Agenda

- Control groups overview
- Control groups what is new?
- Control groups configuration
- Assignment and display of cgroup
- CPU resource examples
- Memory resource examples
- ➢ **Namespaces and containers**

Namespaces

- Lightweight process isolation/virtualization aka Containers
- Each group of processes can have different view of system resources
- No hypervisor layer
- Common namespaces
  - mnt (mountpoints and filesystems)
  - pid (processes)
  - net (networking) independent IP stack, routing, firewall
  - ipc (System V ipc)
  - uts (hostname information)
  - user (uids)
- Namespaces can be created with the unshare command or syscall, or with clone syscall

Examples of technology using cgroups or containers

- Systemd
- Docker http://www.ibm.com/developerworks/linux/linux390/docker.html
- Hadoop yarn – Hadoop cluster resource control
- LXC (LinuX Containters)
- Kubernetes – Container cluster manager from Google
- Lmctfy (Let me contain that for you) (Google)
- Apache Mesos and Mesosphere (Mesosphere Inc)
- Openstack – Starting with Havana
- libvirt-lxc – libvirt driver for Linux containers
- Cloud Foundry's Garden
- Trove - DBaaS
- Google – Joe Beda @ Gluecon 2014 "everything at google runs in a container" and "we start over 2 billion containers per week" see http://www.enterprisetech.com/2014/05/28/google-runs-software-containers/

Namespaces

- Composed mostly of syscalls and a few user space commands such as ip ns
- ls -l /proc/<pid>/ns

  -
    ```
    [root@localhost /]# ls -la /proc/self/ns/
    total 0
    dr-x--x--x 2 root root 0 May 11 11:43 .
    dr-xr-xr-x 9 root root 0 May 11 11:43 ..
    lrwxrwxrwx 1 root root 0 May 11 11:43 ipc -> ipc:[4026531839]
    lrwxrwxrwx 1 root root 0 May 11 11:43 mnt -> mnt:[4026531840]
    lrwxrwxrwx 1 root root 0 May 11 11:43 net -> net:[4026532451]
    lrwxrwxrwx 1 root root 0 May 11 11:43 pid -> pid:[4026531836]
    lrwxrwxrwx 1 root root 0 May 11 11:43 user -> user:[4026531837]
    lrwxrwxrwx 1 root root 0 May 11 11:43 uts -> uts:[4026531838]
    ```
- nsenter – run program with namespace of another process (if no program specified, runs your shell)

Namespaces – network examples

- `unshare` with new shell (could be any program) with unique network namespace

```
[ryoung@localhost /]$ ifconfig
em1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether f0:de:f1:61:bb:02  txqueuelen 1000  (Ethernet)
        RX packets 255301  bytes 134457758 (128.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 203692  bytes 39939401 (38.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0xf2600000-f2620000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 6049694  bytes 6661154950 (6.2 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6049694  bytes 6661154950 (6.2 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
[ryoung@localhost /]$ sudo su
[root@localhost /]# unshare --net /bin/bash
[root@localhost /]# ifconfig
[root@localhost /]#
```

Runs in its own network name space

Namespaces – network example

```
[root@localhost ryoung]# ip netns add mynet
[root@localhost ryoung]# ip netns exec mynet ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
[root@localhost ryoung]# ip netns exec mynet bash
[root@localhost ryoung]# ifconfig
[root@localhost ryoung]# ping ibm.com
ping: unknown host ibm.com
[root@localhost ryoung]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
[root@localhost ryoung]# exit

[root@localhost ryoung]# ip netns exec mynet ifconfig veth1 10.3.2.1/24 up
[root@localhost ryoung]# ifconfig veth0 10.3.2.2/24 up

[root@localhost ryoung]# ip netns exec mynet ping 10.3.2.2
PING 10.3.2.2 (10.3.2.2) 56(84) bytes of data.
64 bytes from 10.3.2.2: icmp_seq=1 ttl=64 time=0.134 ms
64 bytes from 10.3.2.2: icmp_seq=2 ttl=64 time=0.108 ms
^C
--- 10.3.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.108/0.121/0.134/0.013 ms

[root@localhost ryoung]# ip netns exec mynet route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.3.2.0        0.0.0.0         255.255.255.0   U     0      0        0 veth1
[root@localhost ryoung]#
```

New namespace

Running in the new namespace

Connection to new name space

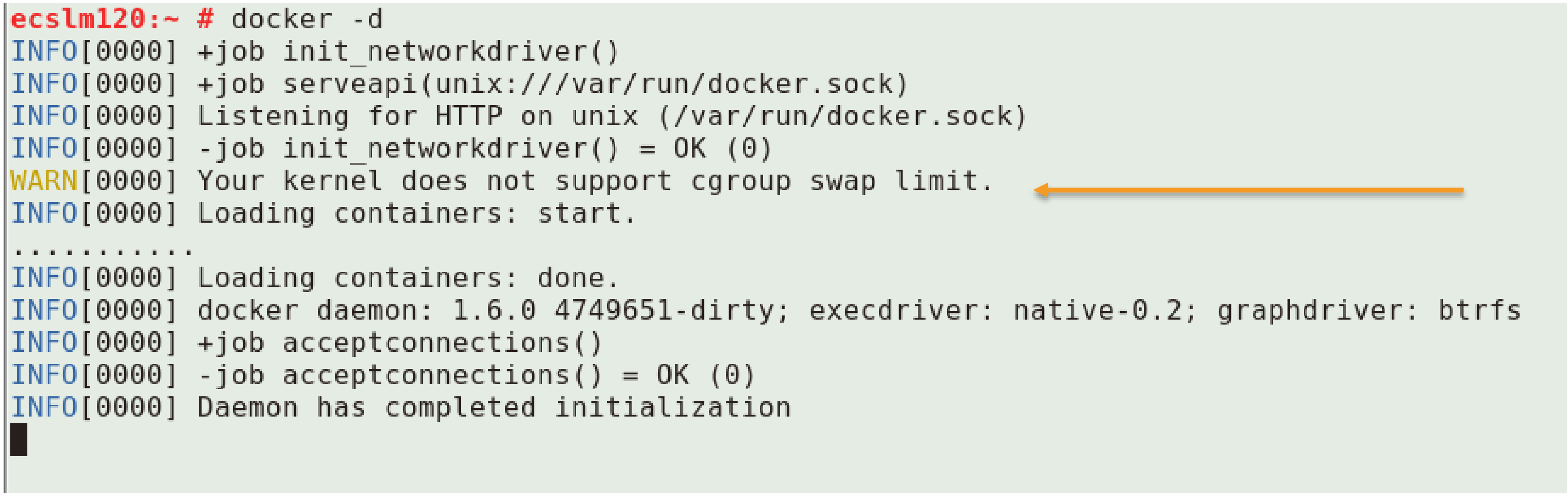
Cgroups and containers

- Docker is a container technology that greatly simplifies management of namespaces, container management, and operation
- Unlike KVM virtualization, the container isolation is via namespaces
- With containers the OS is shared and only the application is started. (very fast to start up and low overhead)
- Cgroups has integration in to container technologies

- Lets take a look…

Starting the Docker daemon

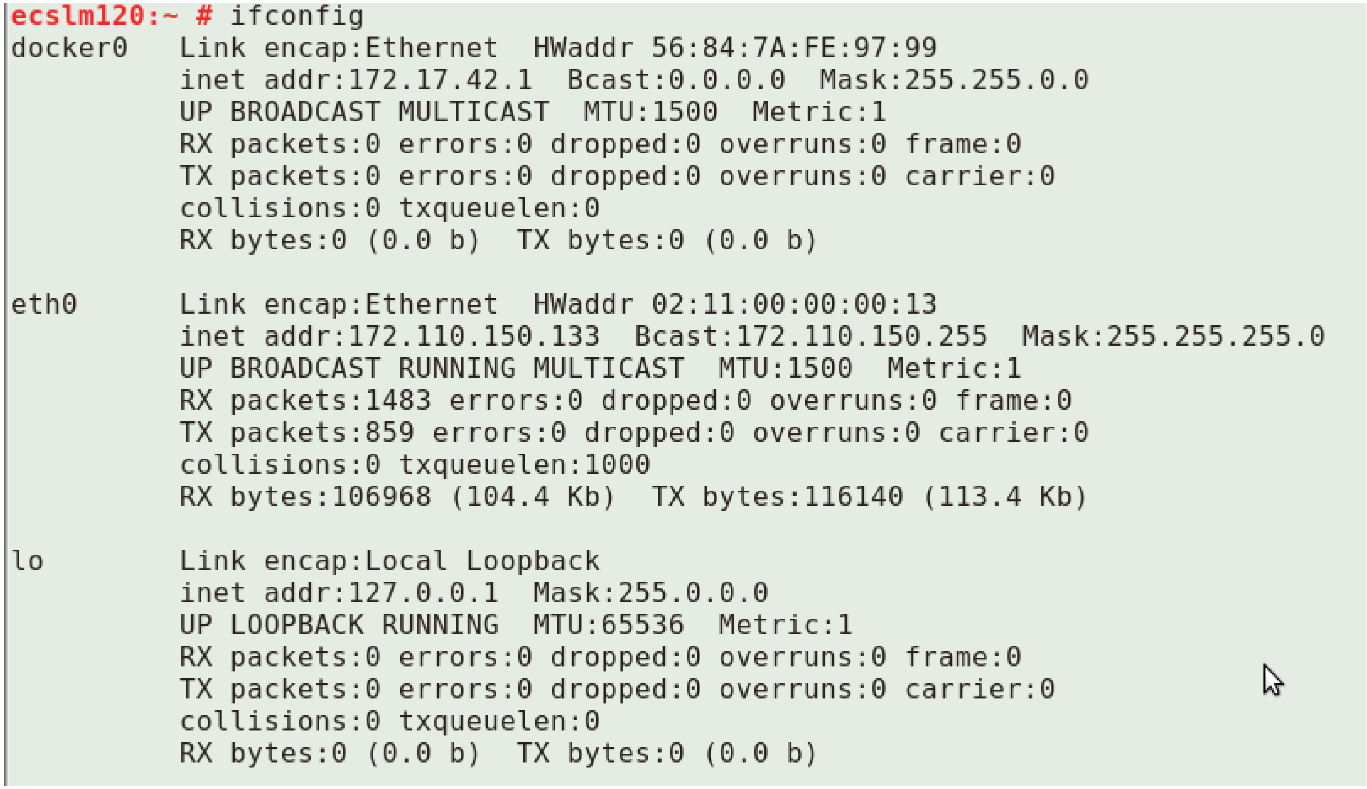
- Notice the cgroup swap limit message
- Cgroup “swapaccount=1” kernel parameter would address



```
ecslm120:~ # docker -d
INFO[0000] +job init_networkdriver()
INFO[0000] +job serveapi(unix:///var/run/docker.sock)
INFO[0000] Listening for HTTP on unix (/var/run/docker.sock)
INFO[0000] -job init_networkdriver() = OK (0)
WARN[0000] Your kernel does not support cgroup swap limit.
INFO[0000] Loading containers: start.
...........
INFO[0000] Loading containers: done.
INFO[0000] docker daemon: 1.6.0 4749651-dirty; execdriver: native-0.2; graphdriver: btrfs
INFO[0000] +job acceptconnections()
INFO[0000] -job acceptconnections() = OK (0)
INFO[0000] Daemon has completed initialization
```

Docker Host IP configuration

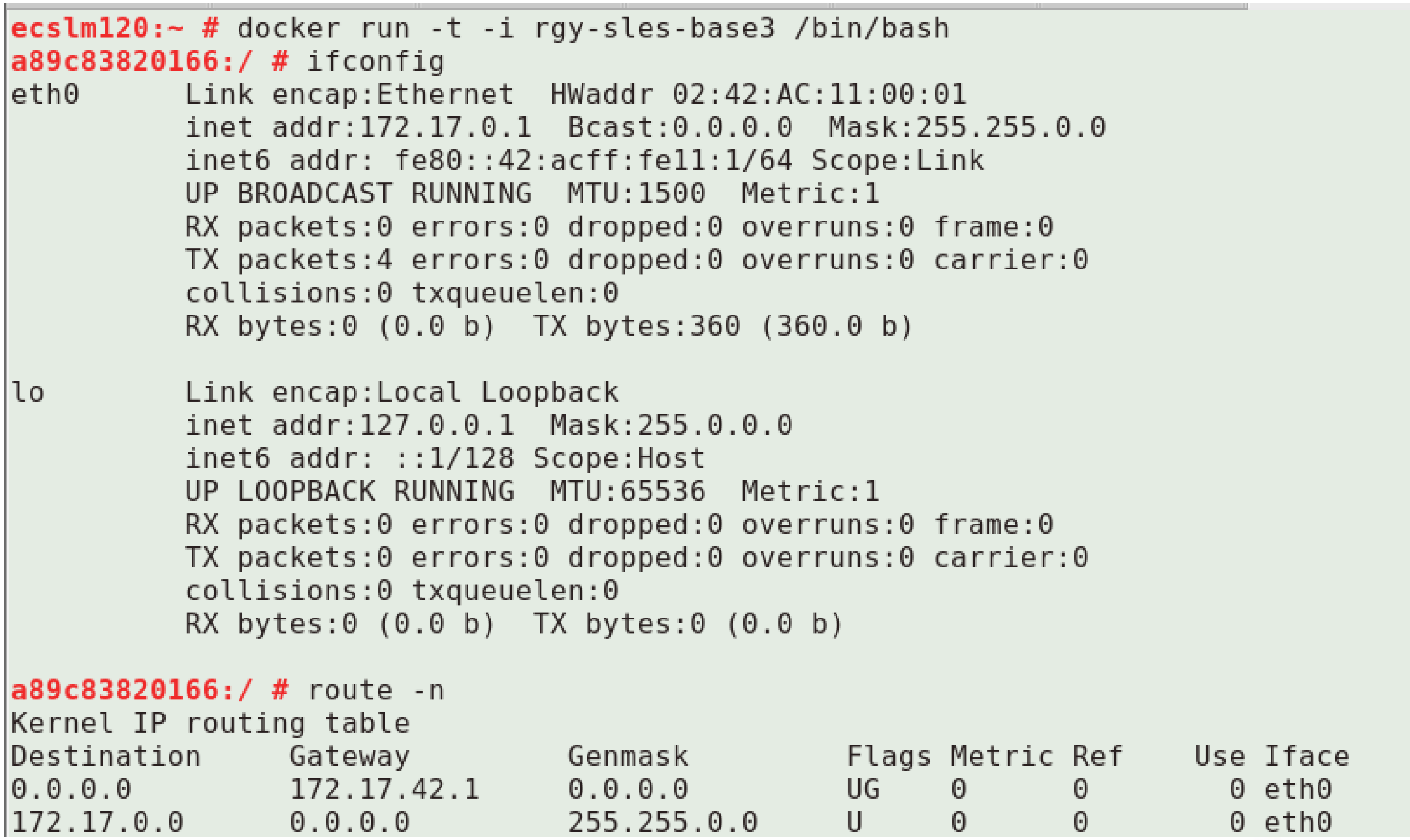
- The host has an automatically assigned interface and IP for the container to talk to and through



```
ecslm120:~ # ifconfig
docker0   Link encap:Ethernet  HWaddr 56:84:7A:FE:97:99
          inet addr:172.17.42.1  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

eth0      Link encap:Ethernet  HWaddr 02:11:00:00:00:13
          inet addr:172.110.150.133  Bcast:172.110.150.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1483 errors:0 dropped:0 overruns:0 frame:0
          TX packets:859 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:106968 (104.4 Kb)  TX bytes:116140 (113.4 Kb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Network Namespace

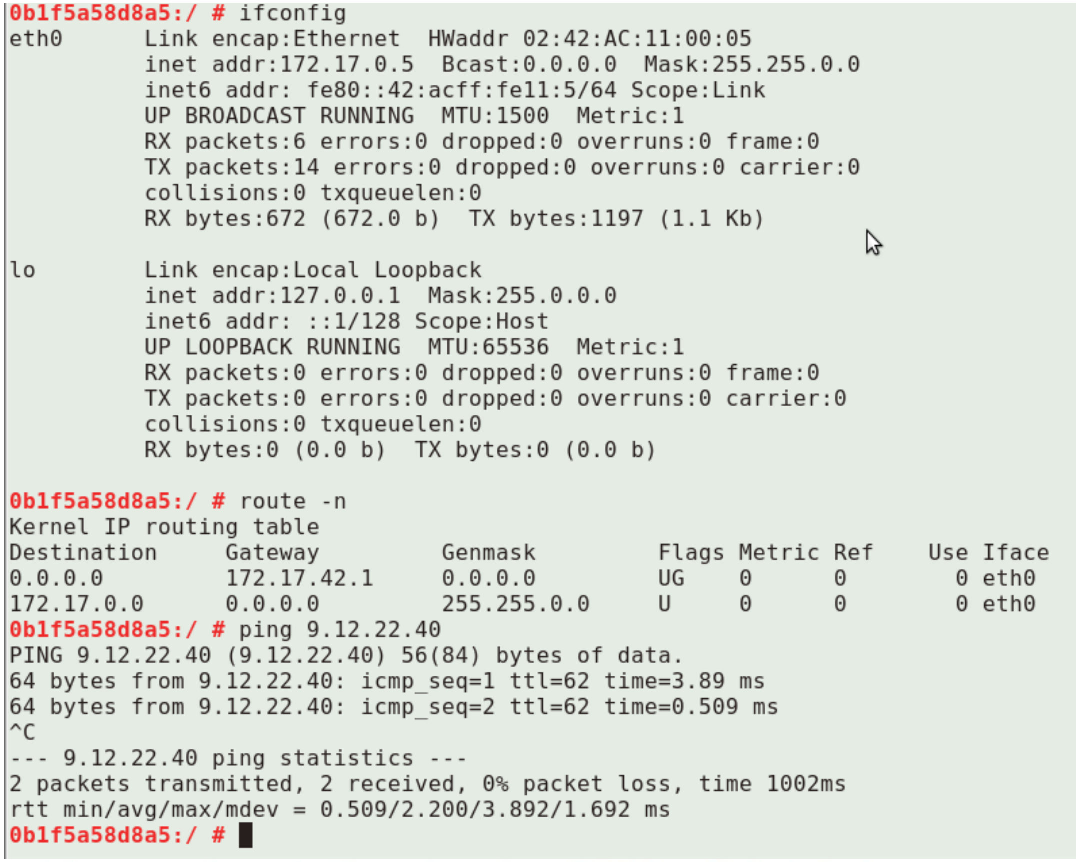
- Unique IP namespaces used for host and guest container
- IP, hostname, and routing are all unique



```
ecslm120:~ # docker run -t -i rgy-sles-base3 /bin/bash
a89c83820166:/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:01
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe11:1/64 Scope:Link
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:360 (360.0 b)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

a89c83820166:/ # route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.17.42.1     0.0.0.0         UG    0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth0
```

Network Namespace

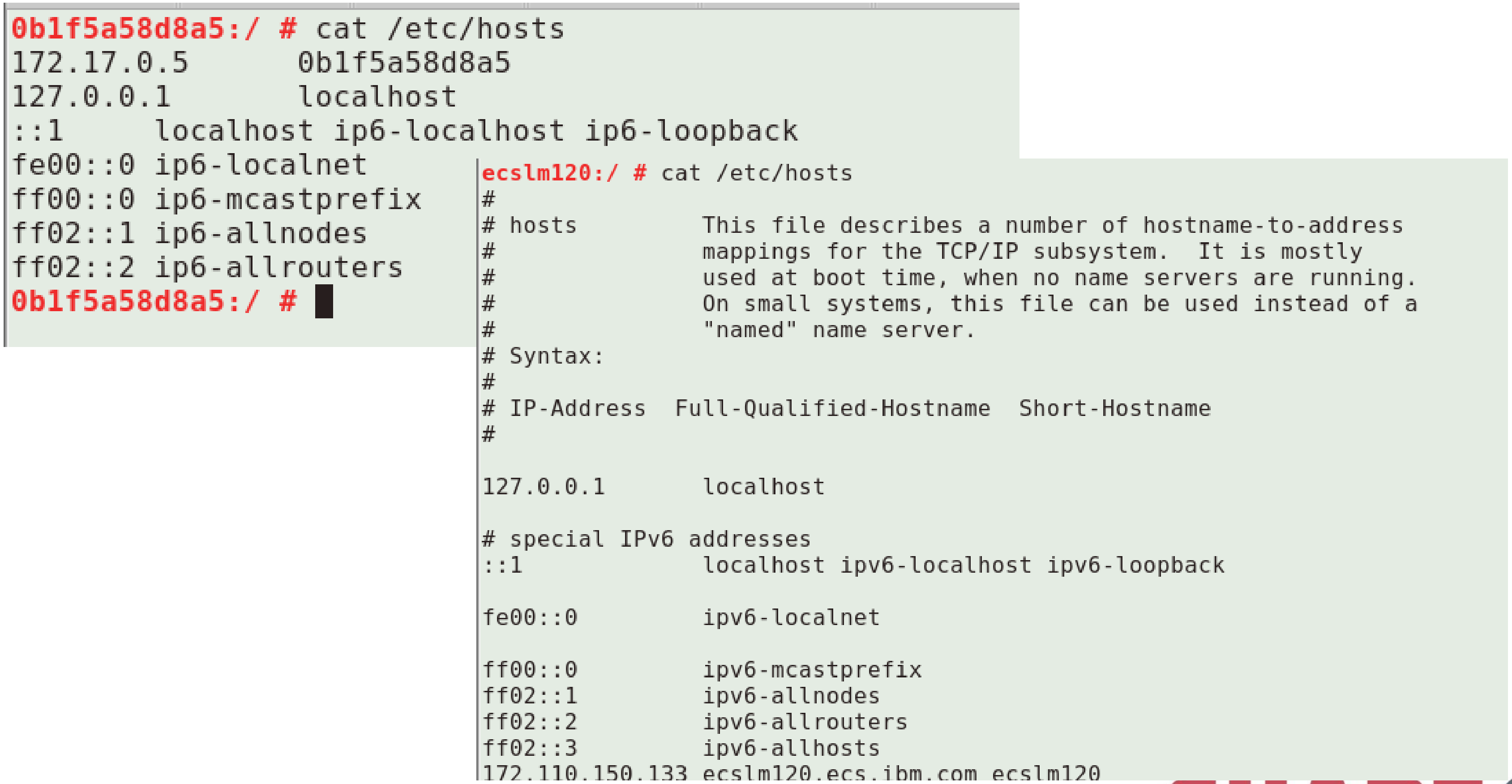
- Guest container can communicate thru host and beyond if allowed
- Docker uses firewall rules and not cgroups to control IP connectivity
- Cgroup control of network traffic is about resource priority



```
0b1f5a58d8a5:/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:05
          inet addr:172.17.0.5  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe11:5/64 Scope:Link
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:672 (672.0 b)  TX bytes:1197 (1.1 Kb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

0b1f5a58d8a5:/ # route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.17.42.1     0.0.0.0         UG    0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth0
0b1f5a58d8a5:/ # ping 9.12.22.40
PING 9.12.22.40 (9.12.22.40) 56(84) bytes of data.
64 bytes from 9.12.22.40: icmp_seq=1 ttl=62 time=3.89 ms
64 bytes from 9.12.22.40: icmp_seq=2 ttl=62 time=0.509 ms
^C
--- 9.12.22.40 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.509/2.200/3.892/1.692 ms
0b1f5a58d8a5:/ # █
```

UTS hostname and filesystem namespace

- Unique hostfiles are used via a combination of network and filesystem namesspaces

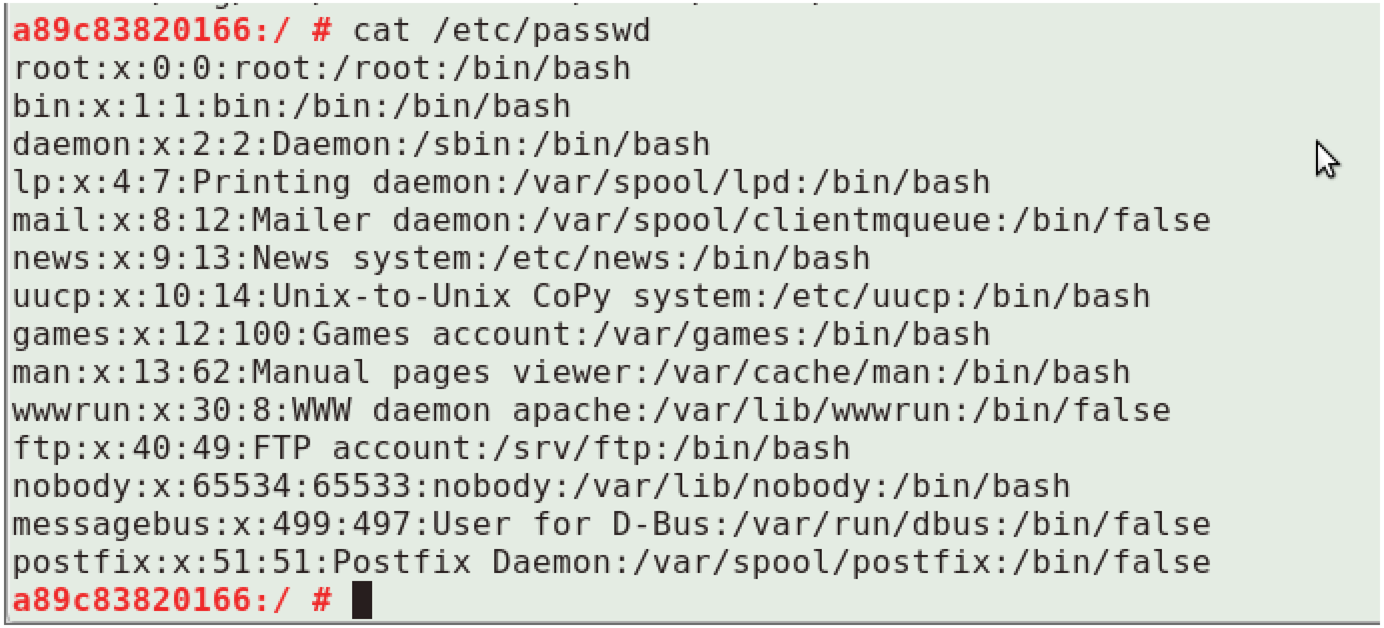


```
0b1f5a58d8a5:/ # cat /etc/hosts
172.17.0.5      0b1f5a58d8a5
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
0b1f5a58d8a5:/ #
```

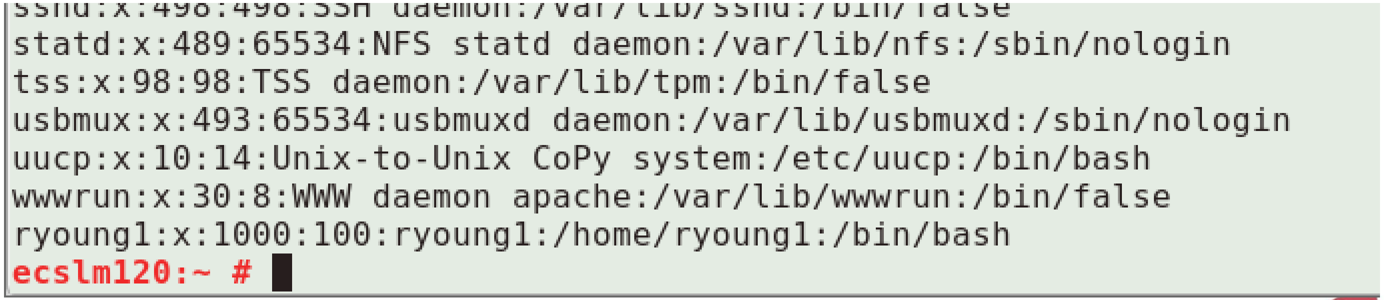
```
ecslm120:/ # cat /etc/hosts
#
# hosts         This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#

127.0.0.1       localhost

# special IPv6 addresses
::1             localhost ipv6-localhost ipv6-loopback

fe00::0         ipv6-localnet

ff00::0         ipv6-mcastprefix
ff02::1         ipv6-allnodes
ff02::2         ipv6-allrouters
ff02::3         ipv6-allhosts
172.110.150.133 ecslm120.ecs.ibm.com ecslm120
```

UID name space

- The container has a unique uid name space and unique /etc/passwd files



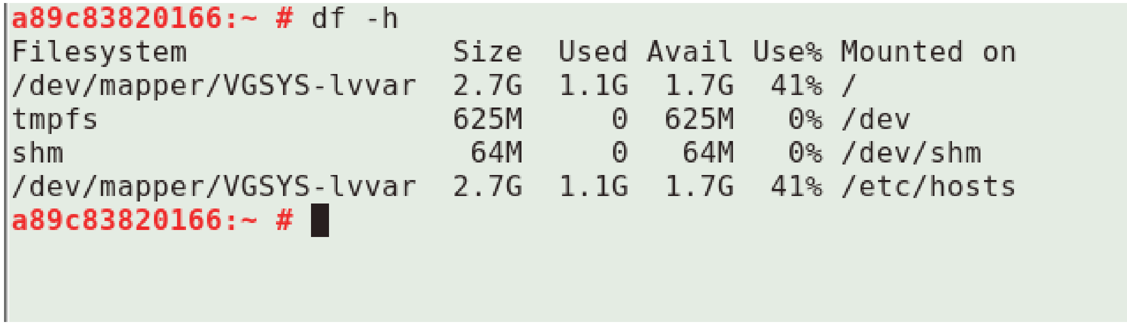
```
a89c83820166:/ # cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:Daemon:/sbin:/bin/bash
lp:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash
mail:x:8:12:Mailer daemon:/var/spool/clientmqueue:/bin/false
news:x:9:13:News system:/etc/news:/bin/bash
uucp:x:10:14:Unix-to-Unix CoPy system:/etc/uucp:/bin/bash
games:x:12:100:Games account:/var/games:/bin/bash
man:x:13:62:Manual pages viewer:/var/cache/man:/bin/bash
wwwrun:x:30:8:WWW daemon apache:/var/lib/wwwrun:/bin/false
ftp:x:40:49:FTP account:/srv/ftp:/bin/bash
nobody:x:65534:65533:nobody:/var/lib/nobody:/bin/bash
messagebus:x:499:497:User for D-Bus:/var/run/dbus:/bin/false
postfix:x:51:51:Postfix Daemon:/var/spool/postfix:/bin/false
a89c83820166:/ #
```



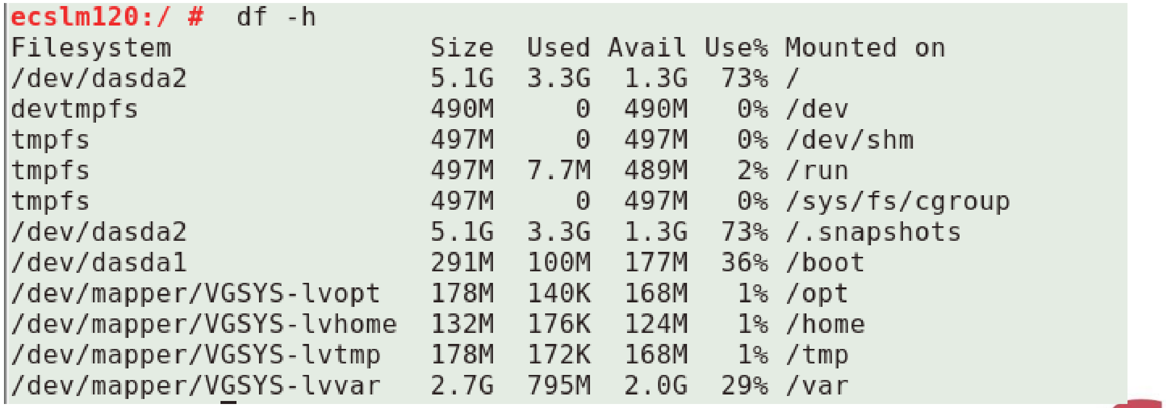
```
sshd:x:498:498:SSH daemon:/var/lib/sshd:/bin/false
statd:x:489:65534:NFS statd daemon:/var/lib/nfs:/sbin/nologin
tss:x:98:98:TSS daemon:/var/lib/tpm:/bin/false
usbmux:x:493:65534:usbmuxd daemon:/var/lib/usbmuxd:/sbin/nologin
uucp:x:10:14:Unix-to-Unix CoPy system:/etc/uucp:/bin/bash
wwwrun:x:30:8:WWW daemon apache:/var/lib/wwwrun:/bin/false
ryoung1:x:1000:100:ryoung1:/home/ryoung1:/bin/bash
ecslm120:~ #
```

Filesystem Namespace

- The host and guest container have unique filesystem namespaces
- Because the docker image lives in /var, that same filesystem is mounted in to the containers namespace
- Notice the /etc/hosts mountpoint in the container display



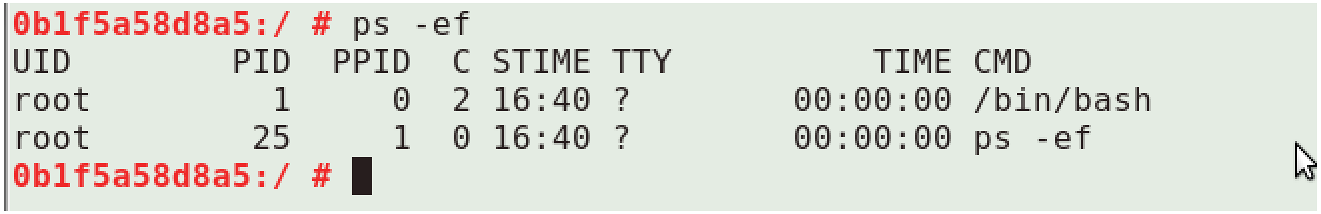
```
a89c83820166:~ # df -h
Filesystem             Size  Used Avail Use% Mounted on
/dev/mapper/VGSYS-lvvar  2.7G  1.1G  1.7G  41% /
tmpfs                  625M     0  625M   0% /dev
shm                     64M     0   64M   0% /dev/shm
/dev/mapper/VGSYS-lvvar  2.7G  1.1G  1.7G  41% /etc/hosts
a89c83820166:~ #
```



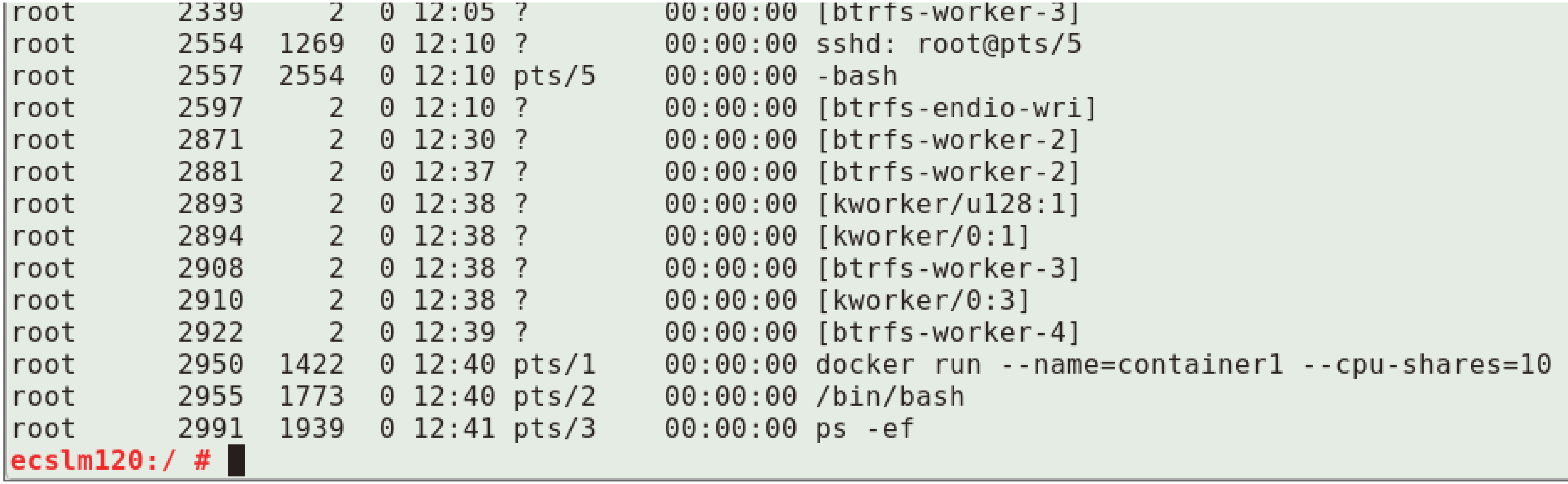
```
ecslm120:/ #  df -h
Filesystem              Size  Used Avail Use% Mounted on
/dev/dasda2             5.1G  3.3G  1.3G  73% /
devtmpfs                490M     0  490M   0% /dev
tmpfs                   497M     0  497M   0% /dev/shm
tmpfs                   497M  7.7M  489M   2% /run
tmpfs                   497M     0  497M   0% /sys/fs/cgroup
/dev/dasda2             5.1G  3.3G  1.3G  73% /.snapshots
/dev/dasda1             291M  100M  177M  36% /boot
/dev/mapper/VGSYS-lvopt  178M  140K  168M   1% /opt
/dev/mapper/VGSYS-lvhome 132M  176K  124M   1% /home
/dev/mapper/VGSYS-lvtmp  178M  172K  168M   1% /tmp
/dev/mapper/VGSYS-lvvar  2.7G  795M  2.0G  29% /var
```

PID name space

- Guest container vs host processes



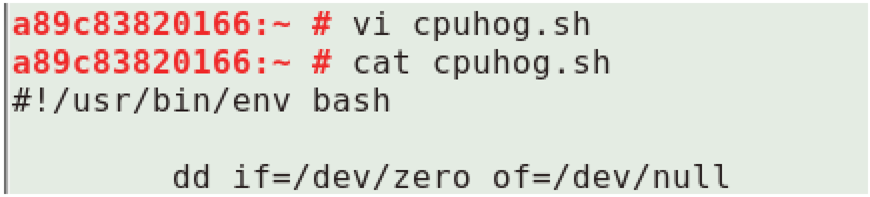
```
0b1f5a58d8a5:/ # ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  2 16:40 ?        00:00:00 /bin/bash
root          25     1  0 16:40 ?        00:00:00 ps -ef
0b1f5a58d8a5:/ #
```



```
root        2339     2  0 12:05 ?        00:00:00 [btrfs-worker-3]
root        2554  1269  0 12:10 ?        00:00:00 sshd: root@pts/5
root        2557  2554  0 12:10 pts/5    00:00:00 -bash
root        2597     2  0 12:10 ?        00:00:00 [btrfs-endio-wri]
root        2871     2  0 12:30 ?        00:00:00 [btrfs-worker-2]
root        2881     2  0 12:37 ?        00:00:00 [btrfs-worker-2]
root        2893     2  0 12:38 ?        00:00:00 [kworker/u128:1]
root        2894     2  0 12:38 ?        00:00:00 [kworker/0:1]
root        2908     2  0 12:38 ?        00:00:00 [btrfs-worker-3]
root        2910     2  0 12:38 ?        00:00:00 [kworker/0:3]
root        2922     2  0 12:39 ?        00:00:00 [btrfs-worker-4]
root        2950  1422  0 12:40 pts/1    00:00:00 docker run --name=container1 --cpu-shares=10
root        2955  1773  0 12:40 pts/2    00:00:00 /bin/bash
root        2991  1939  0 12:41 pts/3    00:00:00 ps -ef
ecslm120:/ #
```

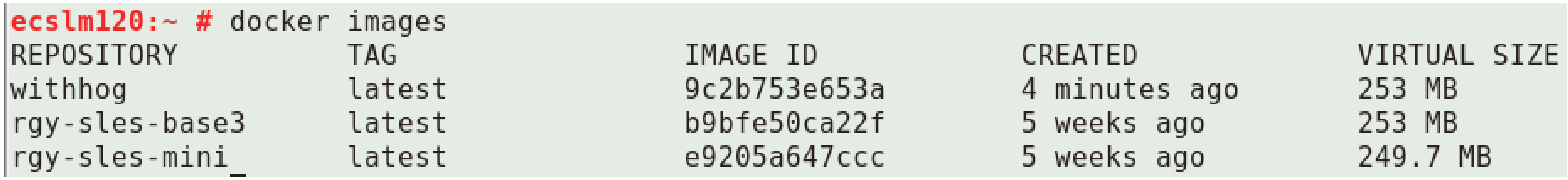
Cgroup resource controls

- Install our application simulator program



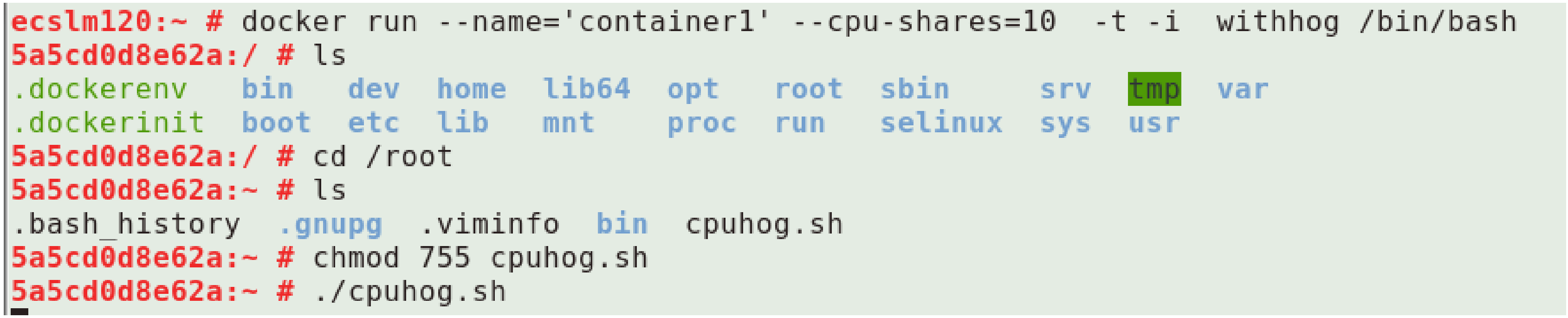
```
a89c83820166:~ # vi cpuhog.sh
a89c83820166:~ # cat cpuhog.sh
#!/usr/bin/env bash

        dd if=/dev/zero of=/dev/null
```

- Image “withhog” created with our application simulator



```
ecslm120:~ # docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
withhog             latest              9c2b753e653a        4 minutes ago       253 MB
rgy-sles-base3      latest              b9bfe50ca22f        5 weeks ago         253 MB
rgy-sles-mini_      latest              e9205a647ccc        5 weeks ago         249.7 MB
```

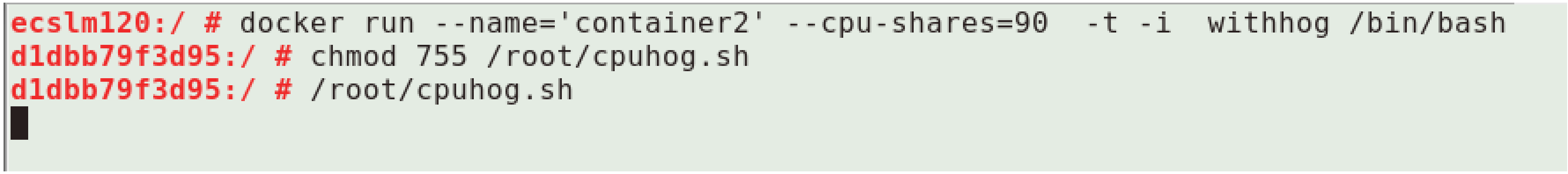
- Launch container with cpu-share resource assignment



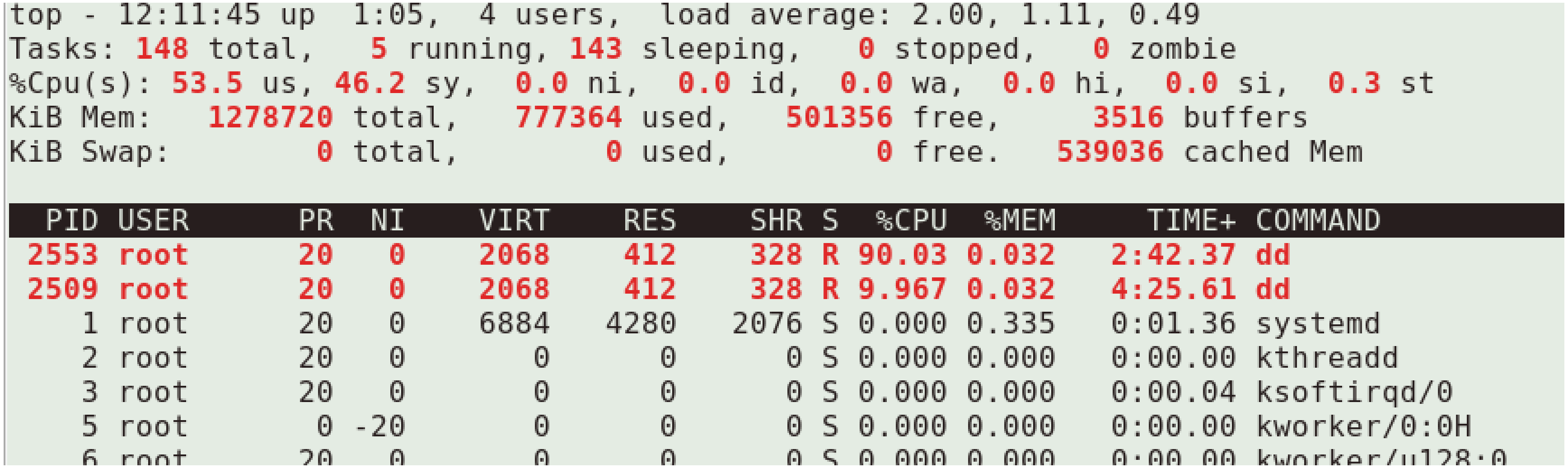
```
ecslm120:~ # docker run --name='container1' --cpu-shares=10  -t -i  withhog /bin/bash
5a5cd0d8e62a:/ # ls
.dockerenv   bin   dev   home   lib64   opt    root   sbin      srv   tmp   var
.dockerinit  boot  etc   lib    mnt     proc   run    selinux   sys   usr
5a5cd0d8e62a:/ # cd /root
5a5cd0d8e62a:~ # ls
.bash_history  .gnupg  .viminfo  bin  cpuhog.sh
5a5cd0d8e62a:~ # chmod 755 cpuhog.sh
5a5cd0d8e62a:~ # ./cpuhog.sh
```

Cgroups resource controls

- Add 2nd container and run in a single CP environment, to observe contention



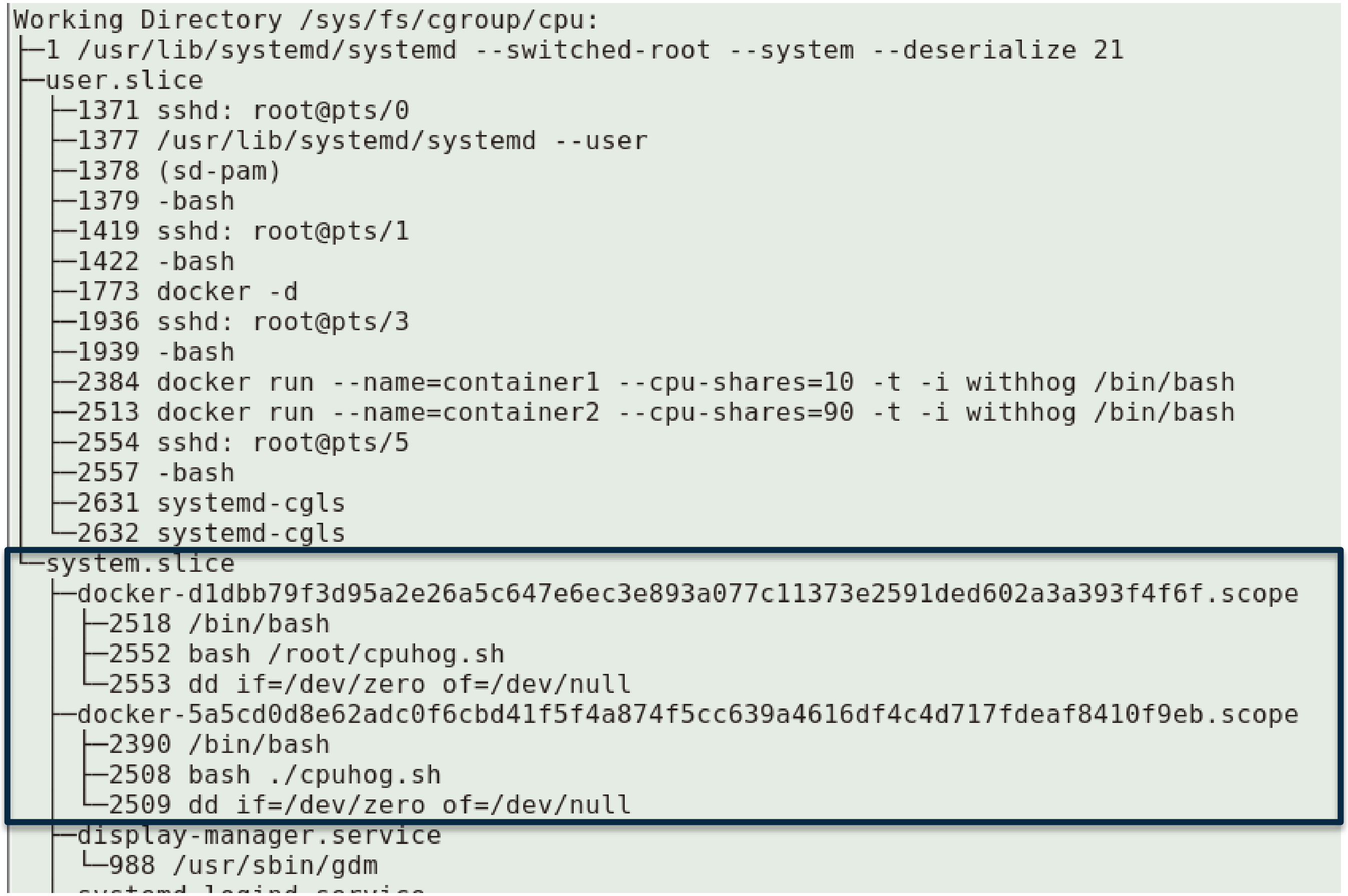
```
ecslm120:/ # docker run --name='container2' --cpu-shares=90  -t -i  withhog /bin/bash
d1dbb79f3d95:/ # chmod 755 /root/cpuhog.sh
d1dbb79f3d95:/ # /root/cpuhog.sh
█
```



```
top - 12:11:45 up  1:05,  4 users,  load average: 2.00, 1.11, 0.49
Tasks: 148 total,   5 running, 143 sleeping,   0 stopped,   0 zombie
%Cpu(s): 53.5 us, 46.2 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.3 st
KiB Mem:   1278720 total,   777364 used,   501356 free,     3516 buffers
KiB Swap:        0 total,        0 used,        0 free.   539036 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2553 root      20   0    2068    412    328 R 90.03 0.032   2:42.37 dd
 2509 root      20   0    2068    412    328 R 9.967 0.032   4:25.61 dd
    1 root      20   0    6884   4280   2076 S 0.000 0.335   0:01.36 systemd
    2 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kthreadd
    3 root      20   0       0      0      0 S 0.000 0.000   0:00.04 ksoftirqd/0
    5 root       0 -20       0      0      0 S 0.000 0.000   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S 0.000 0.000   0:00.00 kworker/u128:0
```

systemd-cgls with containers

- Docker container in “system-slice”



```
Working Directory /sys/fs/cgroup/cpu:
├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 21
├─user.slice
│ ├─1371 sshd: root@pts/0
│ ├─1377 /usr/lib/systemd/systemd --user
│ ├─1378 (sd-pam)
│ ├─1379 -bash
│ ├─1419 sshd: root@pts/1
│ ├─1422 -bash
│ ├─1773 docker -d
│ ├─1936 sshd: root@pts/3
│ ├─1939 -bash
│ ├─2384 docker run --name=container1 --cpu-shares=10 -t -i withhog /bin/bash
│ ├─2513 docker run --name=container2 --cpu-shares=90 -t -i withhog /bin/bash
│ ├─2554 sshd: root@pts/5
│ ├─2557 -bash
│ ├─2631 systemd-cgls
│ └─2632 systemd-cgls
└─system.slice
  ├─docker-d1dbb79f3d95a2e26a5c647e6ec3e893a077c11373e2591ded602a3a393f4f6f.scope
  │ ├─2518 /bin/bash
  │ ├─2552 bash /root/cpuhog.sh
  │ └─2553 dd if=/dev/zero of=/dev/null
  ├─docker-5a5cd0d8e62adc0f6cbd41f5f4a874f5cc639a4616df4c4d717fdeaf8410f9eb.scope
  │ ├─2390 /bin/bash
  │ ├─2508 bash ./cpuhog.sh
  │ └─2509 dd if=/dev/zero of=/dev/null
  ├─display-manager.service
  │ └─988 /usr/sbin/gdm
  │ systemd-logind.service
```

Docker command parameter resource controls

- -m, --memory="": Memory limit (format: <number><optional unit>, where unit = b, k, m or g)
- --memory-swap="": Total memory limit (memory + swap, format: <number><optional unit>, where unit = b, k, m or g)
- -c, --cpu-shares=0: CPU shares (relative weight)
- --cpu-period=0: Limit the CPU CFS (Completely Fair Scheduler) period
- --cpuset-cpus="": CPUs in which to allow execution (0-3, 0,1)
- --cpuset-mems="": Memory nodes (MEMs) in which to allow execution (0-3, 0,1). Only effective on NUMA systems.
- --cpu-quota=0: Limit the CPU CFS (Completely Fair Scheduler) quota
- --blkio-weight=0: Block IO weight (relative weight) accepts a weight value between 10 and 1000.
- --oom-kill-disable=true|false: Whether to disable OOM Killer for the container or not.

Docker command parameter resource controls

- docker run -m 128m (MB of memory)
  - by default the memory.memsw.limit_in_bytes value is set to twice as much as the memory parameter specified while starting a container
  - memory.memsw.limit_in_bytes is the sum of memory and swap

Questions ???

Thanks for attending today!

References

- Docker on z Systems
  - ➢ http://www.ibm.com/developerworks/linux/linux390/docker.html
- Github Linux Kernel Documentation
  - ➢ https://github.com/torvalds/linux/blob/master/Documentation/cgroups/cgroups.txt
- SLES 12
  - ➢ https://www.suse.com/documentation/sles12/book_sle_tuning/data/cha_tuning_cgroups.html
- SLES 11 System Analysis and Tuning Guide
  - ➢ https://www.suse.com/documentation/sles11/singlehtml/book_sle_tuning/book_sle_tuning.html
- RHEL 7
  - ➢ https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Resource_Management_Guide/chap-Introduction_to_Control_Groups.html
- RHEL 6 Resource Management Guide
  - ➢ https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/
- Fedora Wiki
  - ➢ http://fedoraproject.org/wiki/Features/ControlGroups
- OpenSuSE
  - ➢ http://doc.opensuse.org/documentation/html/openSUSE/opensuse-tuning/cha.tuning.cgroups.html

Richard G. Young

Executive I.T. Specialist

IBM STG Lab Services

Virtualization & Linux on z Team Lead

IBM

777 East Wisconsin Ave
Milwaukee, WI 53202

Tel 414 921 4276
Fax 414 921 4276
Mobile 262 893 8662
Email: ryoung1@us.ibm.com