Managing resources with cgroups in systemd

Cgroups manage resources per application rather than by the individual processes that make up an application.

29 Oct 2020 David Both (Correspondent) 35

Image credits : *Image by Mapbox Uncharted ERG, CC-BY 3.0 US*

There is little more frustrating to me as a sysadmin than unexpectedly running out of a computing resource. On more than one occasion, I have filled all available disk space in a partition, run out of RAM, and not had enough CPU time to perform my tasks in a reasonable amount of time. Resource management is one of the most important tasks that sysadmins do.

The point of resource management is to ensure that all processes have relatively equal access to the system resources they need. Resource management also involves ensuring that RAM, hard drive space, and CPU capacity are added when necessary or rationed when that is not possible. In addition, users who hog system resources, whether intentionally or accidentally, should be prevented from doing so.

There are tools that enable sysadmins to monitor and manage various system resources. For example, top and similar tools allow you to monitor the use of memory, I/O, storage (disk, SSD, etc.), network, swap space, CPU usage, and more. These tools, particularly those that are CPU-centric, are mostly based on the paradigm that the running process is the unit of control. At best, they provide a way to adjust the nice number–and through that, the priority—or to kill a running process. (For information about nice numbers, see *Monitoring Linux and Windows hosts with Glances*.)

More on sysadmins

- Enable Sysadmin blog
- The Automated Enterprise: a guide to managing IT with automation
- eBook: Ansible Automation for SysAdmins
- Latest sysadmin articles

Other tools based on traditional resource management in a SystemV environment are managed by the `/etc/security/limits.conf` file and the local configuration files located in the `/etc/security/limits.d` directory. Resources can be limited in a fairly crude but useful manner by user or group. Resources that can be managed include various aspects of RAM, total CPU time per day, total amount of data, priority, nice number, number of concurrent logins, number of processes, maximum file size, and more.

Using cgroups for process management

One major difference between systemd and SystemV is how they handle processes. SystemV treats each process as an entity unto itself. systemd collects related processes into control groups, called cgroups (short for control groups), and manages system resources for the cgroup as a whole. This means resources can be managed per application rather than by the individual processes that make up an application.

The control units for cgroups are called slice units. Slices are a conceptualization that allows systemd to order processes in a tree format for ease of management.

Viewing cgroups

I'll start with some commands that allow you to view various types of information about cgroups. The `systemctl status <service>` command displays slice information about a specified service, including its slice. This example shows the `at` daemon:

```
[root@testvm1 ~]# systemctl status atd.service
● atd.service - Deferred execution scheduler
     Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled; vendor pre
     Active: active (running) since Wed 2020-09-23 12:18:24 EDT; 1 day 3h ago
       Docs: man:atd(8)
   Main PID: 1010 (atd)
      Tasks: 1 (limit: 14760)
     Memory: 440.0K
        CPU: 5ms
     CGroup: /system.slice/atd.service
             └─1010 /usr/sbin/atd -f

Sep 23 12:18:24 testvm1.both.org systemd[1]: Started Deferred execution sched
[root@testvm1 ~]#
```

This is an excellent example of one reason that I find systemd more usable than SystemV and the old init program. There is so much more information here than SystemV could provide. The cgroup entry includes the hierarchical structure where the `system.slice` is systemd (PID 1), and the `atd.service` is one level below and part of the `system.slice`. The second line of the cgroup entry also shows the process ID (PID) and the command used to start the daemon.

The `systemctl` command shows multiple cgroup entries. The `--all` option shows all slices, including those that are not currently active:

```
[root@testvm1 ~]# systemctl -t slice --all
  UNIT                            LOAD   ACTIVE   SUB    DESCRIPTION
  -.slice                         loaded active   active Root Slice
```

```
  system-getty.slice                 loaded active   active system-getty.slice
  system-lvm2\x2dpvscan.slice        loaded active   active system-lvm2\x2dpvsc
  system-modprobe.slice              loaded active   active system-modprobe.sli
  system-sshd\x2dkeygen.slice        loaded active   active system-sshd\x2dkeyg
  system-systemd\x2dcoredump.slice loaded inactive dead   system-systemd\x2dc
  system-systemd\x2dfsck.slice       loaded active   active system-systemd\x2df
  system.slice                       loaded active   active System Slice
  user-0.slice                       loaded active   active User Slice of UID 0
  user-1000.slice                    loaded active   active User Slice of UID 1
  user.slice                         loaded active   active User and Session Sl

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

11 loaded units listed.
To show all installed unit files use 'systemctl list-unit-files'.
[root@testvm1 ~]#
```

The first thing to notice about this data is that it shows user slices for UIDs 0 (root) and 1000, which is my user login. This shows only the slices and not the services that are part of each slice. This data shows that a slice is created for each user at the time they log in. This can provide a way to manage all of a user's tasks as a single cgroup entity.

Explore the cgroup hierarchy

All is well and good so far, but cgroups are hierarchical, and all of the service units run as members of one of the cgroups. Viewing that hierarchy is easy and uses one old command and one new one that is part of systemd.

The `ps` command can be used to map the processes and their locations in the cgroup hierarchy. Note that it is necessary to specify the desired data columns when using the `ps` command. I significantly reduced the volume of output from this command below, but I tried to leave enough so you can get a feel for what you might find on your systems:

```
[root@testvm1 ~]# ps xawf -eo pid,user,cgroup,args
    PID USER     CGROUP                              COMMAND
```

```
      2 root     -                            [kthreadd]
      3 root     -                             \_ [rcu_gp]
      4 root     -                             \_ [rcu_par_gp]
      6 root     -                             \_ [kworker/0:0H-kblockd]
      9 root     -                             \_ [mm_percpu_wq]
     10 root     -                             \_ [ksoftirqd/0]
     11 root     -                             \_ [rcu_sched]
     12 root     -                             \_ [migration/0]
     13 root     -                             \_ [cpuhp/0]
     14 root     -                             \_ [cpuhp/1]
<SNIP>
 625406 root     -                             \_ [kworker/3:0-ata_sff]
 625409 root     -                             \_ [kworker/u8:0-events_unbound
      1 root     0::/init.scope               /usr/lib/systemd/systemd --switc
    588 root     0::/system.slice/systemd-jo /usr/lib/systemd/systemd-journal
    599 root     0::/system.slice/systemd-ud /usr/lib/systemd/systemd-udevd
    741 root     0::/system.slice/auditd.ser /sbin/auditd
    743 root     0::/system.slice/auditd.ser  \_ /usr/sbin/sedispatch
    764 root     0::/system.slice/ModemManag /usr/sbin/ModemManager
    765 root     0::/system.slice/NetworkMan /usr/sbin/NetworkManager --no-da
    767 root     0::/system.slice/irqbalance /usr/sbin/irqbalance --foregroun
    779 root     0::/system.slice/mcelog.ser /usr/sbin/mcelog --ignorenodev -
    781 root     0::/system.slice/rngd.servi /sbin/rngd -f
    782 root     0::/system.slice/rsyslog.se /usr/sbin/rsyslogd -n
<SNIP>
    893 root     0::/system.slice/sshd.servi sshd: /usr/sbin/sshd -D [listene
   1130 root     0::/user.slice/user-0.slice  \_ sshd: root [priv]
   1147 root     0::/user.slice/user-0.slice  |   \_ sshd: root@pts/0
   1148 root     0::/user.slice/user-0.slice  |       \_ -bash
   1321 root     0::/user.slice/user-0.slice  |           \_ screen
   1322 root     0::/user.slice/user-0.slice  |               \_ SCREEN
   1323 root     0::/user.slice/user-0.slice  |                   \_ /bin/bas
 498801 root     0::/user.slice/user-0.slice  |                   |   \_ man
 498813 root     0::/user.slice/user-0.slice  |                   |       \_
   1351 root     0::/user.slice/user-0.slice  |                   \_ /bin/bas
 123293 root     0::/user.slice/user-0.slice  |                   |   \_ man
 123305 root     0::/user.slice/user-0.slice  |                   |       \_
   1380 root     0::/user.slice/user-0.slice  |                   \_ /bin/bas
 625412 root     0::/user.slice/user-0.slice  |                   |   \_ ps x
 625413 root     0::/user.slice/user-0.slice  |                   |   \_ less
 246795 root     0::/user.slice/user-0.slice  |                   \_ /bin/bas
 625338 root     0::/user.slice/user-0.slice  |                       \_ /usr
 625340 root     0::/user.slice/user-0.slice  |                           \_
   1218 root     0::/user.slice/user-1000.sl  \_ sshd: dboth [priv]
   1233 dboth    0::/user.slice/user-1000.sl      \_ sshd: dboth@pts/1
   1235 dboth    0::/user.slice/user-1000.sl          \_ -bash
```

```
<SNIP>
   1010 root     0::/system.slice/atd.servic /usr/sbin/atd -f
   1011 root     0::/system.slice/crond.serv /usr/sbin/crond -n
   1098 root     0::/system.slice/lxdm.servi /usr/sbin/lxdm-binary
   1106 root     0::/system.slice/lxdm.servi  \_ /usr/libexec/Xorg -backgroun
 370621 root     0::/user.slice/user-1000.sl  \_ /usr/libexec/lxdm-session
 370631 dboth    0::/user.slice/user-1000.sl      \_ xfce4-session
 370841 dboth    0::/user.slice/user-1000.sl          \_ /usr/bin/ssh-agent /
 370911 dboth    0::/user.slice/user-1000.sl          \_ xfwm4 --display :0.0
 370930 dboth    0::/user.slice/user-1000.sl          \_ xfce4-panel --displa
 370942 dboth    0::/user.slice/user-1000.sl          |   \_ /usr/lib64/xfce4
ay Notification Area Area where notification icons appear
 370943 dboth    0::/user.slice/user-1000.sl          |   \_ /usr/lib64/xfce4
8681 pulseaudio PulseAudio Plugin Adjust the audio volume of the PulseAudio s
 370944 dboth    0::/user.slice/user-1000.sl          |   \_ /usr/lib64/xfce4
8682 power-manager-plugin Power Manager Plugin Display the battery levels of
 370945 dboth    0::/user.slice/user-1000.sl          |   \_ /usr/lib64/xfce4
3068683 notification-plugin Notification Plugin Notification plugin for the X
 370948 dboth    0::/user.slice/user-1000.sl          |   \_ /usr/lib64/xfce4
ons Action Buttons Log out, lock or other system actions
 370934 dboth    0::/user.slice/user-1000.sl          \_ Thunar --sm-client-i
 370939 dboth    0::/user.slice/user-1000.sl          \_ xfdesktop --display
 370962 dboth    0::/user.slice/user-1000.sl          \_ nm-applet
<SNIP>
```

You can view the entire hierarchy with the `systemd-cgls` command, which is a bit simpler because it does not require any complex options.

I have shortened this tree view considerably. as well, but I left enough to give you some idea of the amount of data as well as the types of entries you should see when you do this on your system. I did this on one of my virtual machines, and it is about 200 lines long; the amount of data from my primary workstation is about 250 lines:

```
[root@testvm1 ~]# systemd-cgls
Control group /:
-.slice
├─user.slice
| ├─user-0.slice
| | ├─session-1.scope
| | | ├─  1130 sshd: root [priv]
| | | ├─  1147 sshd: root@pts/0
```

```
| | | ├─  1148 -bash
| | | ├─  1321 screen
| | | ├─  1322 SCREEN
| | | ├─  1323 /bin/bash
| | | ├─  1351 /bin/bash
| | | ├─  1380 /bin/bash
| | | ├─123293 man systemd.slice
| | | ├─123305 less
| | | ├─246795 /bin/bash
| | | ├─371371 man systemd-cgls
| | | ├─371383 less
| | | ├─371469 systemd-cgls
| | | └─371470 less
| | └─user@0.service …
| |   ├─dbus-broker.service
| |   | ├─1170 /usr/bin/dbus-broker-launch --scope user
| |   | └─1171 dbus-broker --log 4 --controller 12 --machine-id 3bccd1140fca4
| |   ├─gvfs-daemon.service
| |   | └─1173 /usr/libexec/gvfsd
| |   └─init.scope
| |     ├─1137 /usr/lib/systemd/systemd --user
| |     └─1138 (sd-pam)
| └─user-1000.slice
|   ├─user@1000.service …
|   | ├─dbus\x2d:1.2\x2dorg.xfce.Xfconf.slice
|   | | └─dbus-:1.2-org.xfce.Xfconf@0.service
|   | |   └─370748 /usr/lib64/xfce4/xfconf/xfconfd
|   | ├─dbus\x2d:1.2\x2dca.desrt.dconf.slice
|   | | └─dbus-:1.2-ca.desrt.dconf@0.service
|   | |   └─371262 /usr/libexec/dconf-service
|   | ├─dbus-broker.service
|   | | ├─1260 /usr/bin/dbus-broker-launch --scope user
|   | | └─1261 dbus-broker --log 4 --controller 11 --machine-id
<SNIP>
|   | └─gvfs-mtp-volume-monitor.service
|   |   └─370987 /usr/libexec/gvfs-mtp-volume-monitor
|   ├─session-3.scope
|   | ├─1218 sshd: dboth [priv]
|   | ├─1233 sshd: dboth@pts/1
|   | └─1235 -bash
|   └─session-7.scope
|     ├─370621 /usr/libexec/lxdm-session
|     ├─370631 xfce4-session
|     ├─370805 /usr/bin/VBoxClient --clipboard
|     ├─370806 /usr/bin/VBoxClient --clipboard
|     ├─370817 /usr/bin/VBoxClient --seamless
```

```
|      ├─370818 /usr/bin/VBoxClient --seamless
|      ├─370824 /usr/bin/VBoxClient --draganddrop
|      ├─370825 /usr/bin/VBoxClient --draganddrop
|      ├─370841 /usr/bin/ssh-agent /bin/sh -c exec -l bash -c "/usr/bin/startx
|      ├─370910 /bin/gpg-agent --sh --daemon --write-env-file /home/dboth/.cac
|      ├─370911 xfwm4 --display :0.0 --sm-client-id 2dead44ab-0b4d-4101-bca4-e
|      ├─370923 xfsettingsd --display :0.0 --sm-client-id 261b4a437-3029-461c-
|      ├─370930 xfce4-panel --display :0.0 --sm-client-id 2ce38b8ef-86fd-4189-
|      ├─370934 Thunar --sm-client-id 2cfc809d8-4e1d-497a-a5c5-6e4fa509c3fb --
|      ├─370939 xfdesktop --display :0.0 --sm-client-id 299be0608-4dca-4055-b4
<SNIP>
└─system.slice
  ├─rngd.service
  | └─1650 /sbin/rngd -f
  ├─irqbalance.service
  | └─1631 /usr/sbin/irqbalance --foreground
  ├─fprintd.service
  | └─303383 /usr/libexec/fprintd
  ├─systemd-udevd.service
  | └─956 /usr/lib/systemd/systemd-udevd
<SNIP>
  ├─systemd-journald.service
  | └─588 /usr/lib/systemd/systemd-journald
  ├─atd.service
  | └─1010 /usr/sbin/atd -f
  ├─system-dbus\x2d:1.10\x2dorg.freedesktop.problems.slice
  | └─dbus-:1.10-org.freedesktop.problems@0.service
  |   └─371197 /usr/sbin/abrt-dbus -t133
  ├─sshd.service
  | └─893 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
  ├─vboxservice.service
  | └─802 /usr/sbin/VBoxService -f
  ├─crond.service
  | └─1011 /usr/sbin/crond -n
  ├─NetworkManager.service
  | └─765 /usr/sbin/NetworkManager --no-daemon
  ├─switcheroo-control.service
  | └─787 /usr/libexec/switcheroo-control
 <SNIP>
```

This tree view shows all of the user and system slices and the services and programs running in each cgroup. Notice the units called "scopes," which group related programs into a management unit, within the `user-1000.slice` in the listing above. The `user-1000.slice/session-7.scope` cgroup contains the GUI

desktop program hierarchy, starting with the LXDM display manager session and all of its subtasks, including things like the Bash shell and the Thunar GUI file manager.

Scope units are not defined in configuration files but are generated programmatically as the result of starting groups of related programs. Scope units do not create or start the processes running as part of that cgroup. All processes within the scope are equal, and there is no internal hierarchy. The life of a scope begins when the first process is created and ends when the last process is destroyed.

Open several windows on your desktop, such as terminal emulators, LibreOffice, or whatever you want, then switch to an available virtual console and start something like `top` or Midnight Commander. Run the `systemd-cgls` command on your host, and take note of the overall hierarchy and the scope units.

The `systemd-cgls` command provides a more complete representation of the cgroup hierarchy (and details of the units that make it up) than any other command I have found. I prefer its cleaner representation of the tree than what the `ps`command provides.

With a little help from my friends

After covering these basics, I had planned to go into more detail about cgroups and how to use them, but I discovered a series of four excellent articles by Red Hat's Steve Ovens on Opensource.com's sister site Enable Sysadmin. Rather then basically rewriting Steve's articles, I decided it would be much better to take advantage of his cgroup expertise by linking to them:

1. A Linux sysadmin's introduction to cgroups
2. How to manage cgroups with CPUShares
3. Managing cgroups the hard way—manually
4. Managing cgroups with systemd

Enjoy and learn from them, as I did.

Other resources

There is a great deal of information about systemd available on the internet, but much is terse, obtuse, or even misleading. In addition to the resources mentioned in this article, the following webpages offer more detailed and reliable information about systemd startup. This list has grown since I started this series of articles to reflect the research I have done.

- The Fedora Project has a good, practical guide to systemd. It has pretty much everything you need to know in order to configure, manage, and maintain a Fedora computer using systemd.
- The Fedora Project also has a good cheat sheet that cross-references the old SystemV commands to comparable systemd ones.
- The systemd.unit(5) manual page contains a nice list of unit file sections and their configuration options along with concise descriptions of each.
- Red Hat documentation contains a good description of the Unit file structure as well as other important information.
- For detailed technical information about systemd and the reasons for creating it, check out Freedesktop.org's description of systemd. This page is one of the best I have found because it contains many links to other important and accurate documentation.
- Linux.com's "More systemd fun" offers more advanced systemd information and tips.
- See the man page for systemd.resource-control(5).
- In *The Linux kernel user's and administrator's guide*, see the Control Group v2 entry.

There is also a series of deeply technical articles for Linux sysadmins by Lennart Poettering, the designer and primary developer of systemd. These articles were written between April 2010 and September 2011, but they are just as relevant now as they were then. Much of everything else good that has been written about systemd and its ecosystem is based on these papers.

- Rethinking PID 1
- systemd for Administrators, Part I

- systemd for Administrators, Part II
- systemd for Administrators, Part III
- systemd for Administrators, Part IV
- systemd for Administrators, Part V
- systemd for Administrators, Part VI
- systemd for Administrators, Part VII
- systemd for Administrators, Part VIII
- systemd for Administrators, Part IX
- systemd for Administrators, Part X
- systemd for Administrators, Part XI