Desktop Resource Management (GNOME)

Benjamin Berg <bberg@redhat.com>

August 25, 2020

Section 1

Motivation

Red Hat

Resource distribution

- CPU time
- Memory (caches, data, ...)
- IO access

Red Hat

Resource distribution – status quo

- Resources are distributed between processes
- Various controls available:
  - process nice value
  - ulimit

⇒ All processes are usually treated equally

Red Hat

Resource distribution – status quo

- Resources are distributed between processes
- Various controls available:
  - process nice value
  - ulimit

⇒ All processes are usually treated equally

Red Hat

Resource distribution – what we want

- Treat users equally
- Treat applications equally
- Keep the desktop responsive
- Possibly discriminate based on
  - how important a service is
  - whether a user is active
  - whether an application is focused
- Improved power management
  - improve power attribution
  - freeze background application
  - ...

Red Hat

Resource distribution – what we want

- Treat users equally
- Treat applications equally
- Keep the desktop responsive
- Possibly discriminate based on
  - how important a service is
  - whether a user is active
  - whether an application is focused
- Improved power management
  - improve power attribution
  - freeze background application
  - ...

Resource distribution – what we want

- Treat users equally
- Treat applications equally
- Keep the desktop responsive
- Possibly discriminate based on
  - how important a service is
  - whether a user is active
  - whether an application is focused
- Improved power management
  - improve power attribution
  - freeze background application
  - ...

Red Hat

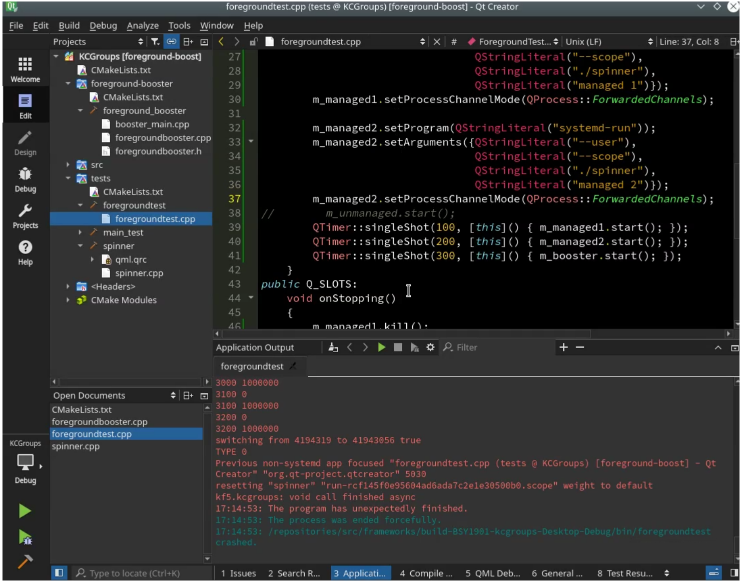
Resource distribution – what we want

- Treat users equally
- Treat applications equally
- Keep the desktop responsive
- Possibly discriminate based on
  - how important a service is
  - whether a user is active
  - whether an application is focused
- Improved power management
  - improve power attribution
  - freeze background application
  - ...

Red Hat

Resource distribution – what we want

- Treat users equally
- Treat applications equally
- Keep the desktop responsive
- Possibly discriminate based on
  - how important a service is
  - whether a user is active
  - whether an application is focused
- Improved power management
  - improve power attribution
  - freeze background application
  - ...



Spinner demo created by David Edmundson (video)

Red Hat

Thrashing and OOM handling

- Still a problem in 2020
- Shell and graphical applications are susceptible
- Various approaches exist:
  - MemoryAvailable based (e.g. EarlyOOM, nohang)
  - PSI based (e.g. nohang, low-memory-monitor, oomd)
  - Faster swap (e.g. swap on zram)

Red Hat

Thrashing and OOM handling

- Still a problem in 2020
- Shell and graphical applications are susceptible
- Various approaches exist:
  - MemoryAvailable based (e.g. EarlyOOM, nohang)
  - PSI based (e.g. nohang, low-memory-monitor, oomd)
  - Faster swap (e.g. swap on zram)

⇒

Red Hat

Thrashing and OOM handling

- Still a problem in 2020
- Shell and graphical applications are susceptible
- Various approaches exist:
  - MemoryAvailable based (e.g. EarlyOOM, nohang)
  - PSI based (e.g. nohang, low-memory-monitor, oomd)
  - Faster swap (e.g. swap on zram)

$\Rightarrow$ Reasonably fast
Effectively ensures the kernel has enough space for (file) caches

Red Hat

Thrashing and OOM handling

- Still a problem in 2020
- Shell and graphical applications are susceptible
- Various approaches exist:
  - MemoryAvailable based (e.g. EarlyOOM, nohang)
  - PSI based (e.g. nohang, low-memory-monitor, oomd)
  - Faster swap (e.g. swap on zram)

$\Rightarrow$ PSI is inherently slow (>10 s)
Good at identifying thrashing workloads

Red Hat

Thrashing and OOM handling

- Still a problem in 2020
- Shell and graphical applications are susceptible
- Various approaches exist:
  - MemoryAvailable based (e.g. EarlyOOM, nohang)
  - PSI based (e.g. nohang, low-memory-monitor, oomd)
  - Faster swap (e.g. swap on zram)

⇒ Shown to help with interactivity

Red Hat

Thrashing and OOM handling

- Still a problem in 2020
- Shell and graphical applications are susceptible
- Various approaches exist:
  - MemoryAvailable based (e.g. EarlyOOM, nohang)
  - PSI based (e.g. nohang, low-memory-monitor, oomd)
  - Faster swap (e.g. swap on zram)

$\Rightarrow$ Not effective at protecting graphical session

Red Hat

Thrashing and OOM handling

What do we really need?

- Responsive shell and task manager
- Ability to identify and kill problematic tasks
- Isolate runaway applications

Red Hat

Thrashing and OOM handling

What do we really need?

- Responsive shell and task manager
- Ability to identify and kill problematic tasks
- Isolate runaway applications

- cgroups can be used to protect these tasks
  e.g. `memory.low`, CPU controller, IO controller

⇒ Prevent problematic situations from getting worse!

Red Hat

Thrashing and OOM handling

What do we really need?

- Responsive shell and task manager
- Ability to identify and kill problematic tasks
- Isolate runaway applications

- Memory pressure based (PSI)
- systemd-oomd

Red Hat

Thrashing and OOM handling

What do we really need?

- Responsive shell and task manager
- Ability to identify and kill problematic tasks
- Isolate runaway applications

- Place each application into a cgroup

Section 2

systemd

Red Hat

systemd

- Allows managing kernel cgroups
- Desktop Environments were not ready until recently

Red Hat

systemd – work that has happened

- DBus per-user session bus
- Fixes across the stack for session detection
- Services were ported to systemd
- GNOME session itself being ported
- VTE (gnome-terminal) creates a scope for each tab
- Other Desktop Environments are also working on this

Red Hat

systemd – work that has happened

- DBus per-user session bus
- Fixes across the stack for session detection
- Services were ported to systemd
- GNOME session itself being ported
- VTE (gnome-terminal) creates a scope for each tab
- Other Desktop Environments are also working on this

Red Hat

systemd – conventions

- A draft is available
  `https://systemd.io/DESKTOP_ENVIRONMENTS/`
- Split user cgroups into three parts:

`session.slice`	Essential session processes
`app.slice`	Normal applications
`background.slice`	Background tasks

  Everything should be moved into one of these.
- Encode application ID in systemd unit name

Red Hat

systemd – conventions

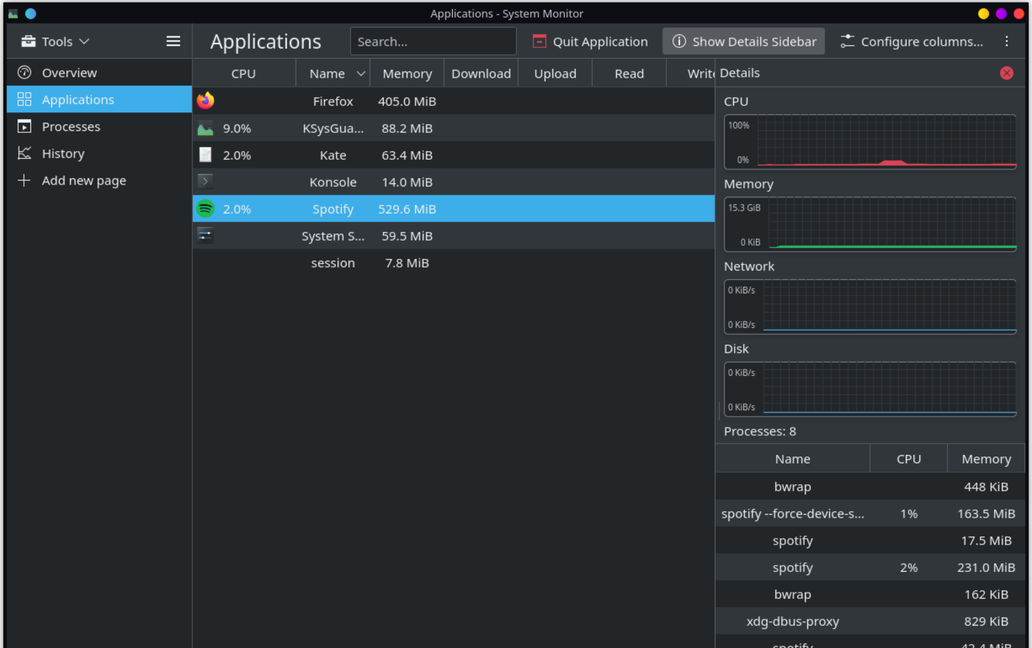
```
cgroupfs
├─ system.slice
└─ user.slice
   └─ user-1000.slice
      ├─ session-2.scope
      │     X server and a few other processes
      └─ user@1000.service
         ├─ session.slice
         │  ├─ org.gnome.Shell@wayland.service
         │  ├─ org.gnome.SettingsDaemon.*.service
         │  └─ ...
         ├─ app.slice
         │  └─ Applications should go here
         └─ background.slice
```

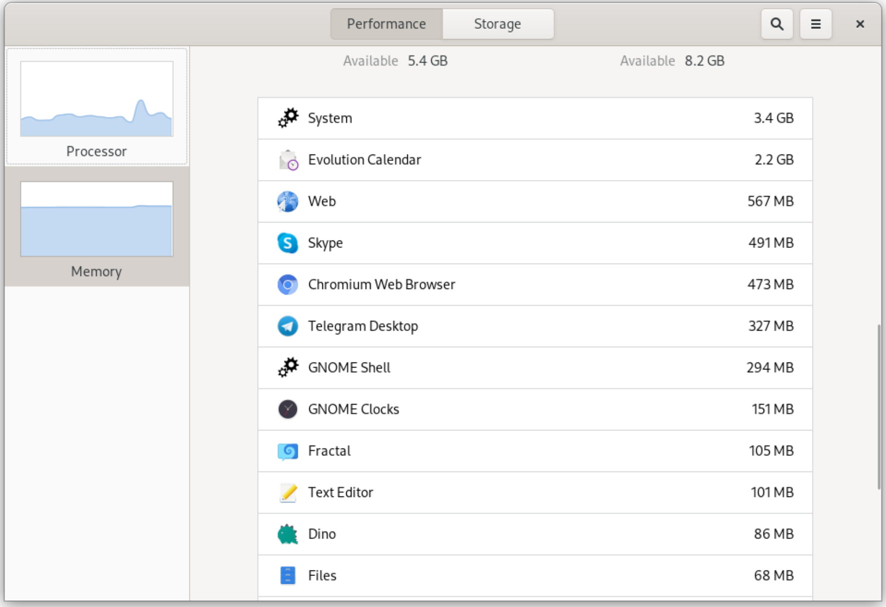
Red Hat

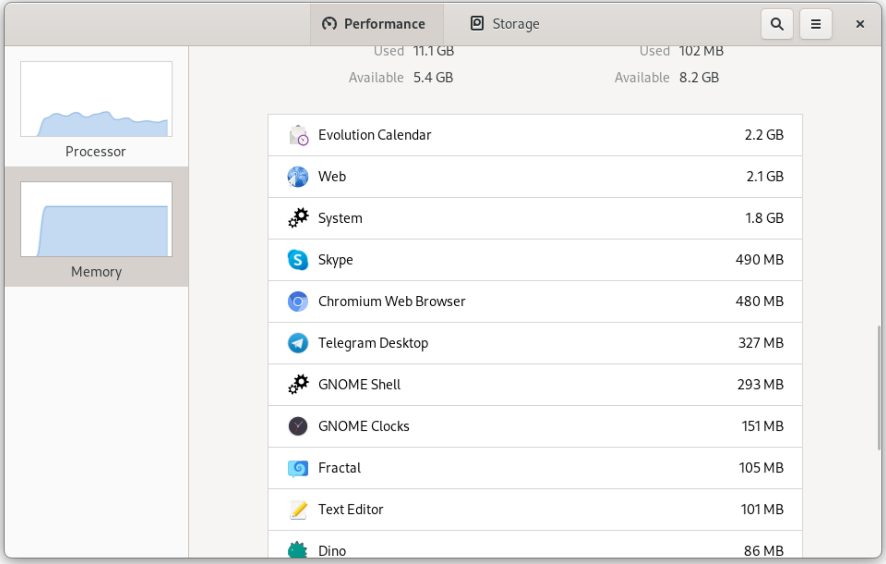
systemd – what we can do

- Modify cgroup attributes per-slice and per-application
- Manage per-application resources
- Create a task manager that properly shows applications rather than processes
  https://gitlab.gnome.org/GNOME/gnome-usage/-/merge_requests/72

  Example done in KDE:
  http://blog.davidedmundson.co.uk/blog/modern-process-management-on-the-desktop/


Applications - System Monitor
Tools
Applications
Search...
Quit Application
Show Details Sidebar
Configure columns...
Overview
Applications
Processes
History
Add new page
CPU
Name
Memory
Download
Upload
Read
Writ
Firefox 405.0 MiB
9.0% KSysGua... 88.2 MiB
2.0% Kate 63.4 MiB
Konsole 14.0 MiB
2.0% Spotify 529.6 MiB
System S... 59.5 MiB
session 7.8 MiB
Details
CPU
100%
0%
Memory
15.3 GiB
0 KiB
Network
0 KiB/s
0 KiB/s
Disk
0 KiB/s
0 KiB/s
Processes: 8
Name
CPU
Memory
bwrap 448 KiB
spotify --force-device-s... 1% 163.5 MiB
spotify 17.5 MiB
spotify 2% 231.0 MiB
bwrap 162 KiB
xdg-dbus-proxy 829 KiB


Performance
Storage
Processor
Memory
Available 5.4 GB
Available 8.2 GB
System 3.4 GB
Evolution Calendar 2.2 GB
Web 567 MB
Skype 491 MB
Chromium Web Browser 473 MB
Telegram Desktop 327 MB
GNOME Shell 294 MB
GNOME Clocks 151 MB
Fractal 105 MB
Text Editor 101 MB
Dino 86 MB
Files 68 MB


Performance
Storage
Used 11.1 GB
Available 5.4 GB
Used 102 MB
Available 8.2 GB
Processor
Memory
Evolution Calendar 2.2 GB
Web 2.1 GB
System 1.8 GB
Skype 490 MB
Chromium Web Browser 480 MB
Telegram Desktop 327 MB
GNOME Shell 293 MB
GNOME Clocks 151 MB
Fractal 105 MB
Text Editor 101 MB
Dino 86 MB

systemd – ongoing tasks

- We want to rely on systemd for more purposes
  e.g. launching XDG autostart applications
- APIs are needed to correctly launch applications
  - KDE has working ApplicationLauncherJob/CommandLauncherJob APIs[1]
  - GLib APIs will be updated to use scopes[2]
- However, it is already useful as is!

---

[1] https://api.kde.org/frameworks/kio/html/classKIO_1_1ApplicationLauncherJob.html
https://api.kde.org/frameworks/kio/html/classKIO_1_1CommandLauncherJob.html

[2] https://gitlab.gnome.org/GNOME/glib/-/merge_requests/1596

Red Hat

systemd – ongoing tasks

- We want to rely on systemd for more purposes
  e.g. launching XDG autostart applications
- APIs are needed to correctly launch applications
  - KDE has working ApplicationLauncherJob/CommandLauncherJob APIs[1]
  - GLib APIs will be updated to use scopes[2]
- However, it is already useful as is!

---

[1] https://api.kde.org/frameworks/kio/html/classKIO_1_1ApplicationLauncherJob.html
https://api.kde.org/frameworks/kio/html/classKIO_1_1CommandLauncherJob.html
[2] https://gitlab.gnome.org/GNOME/glib/-/merge_requests/1596

Section 3

uresourced

Red Hat

uresourced – taking the next step

- low-level functionality is mostly ready
- none of the features are currently enabled
- It is easy and safe to do though!

Red Hat

uresourced – taking the next step

- low-level functionality is mostly ready
- none of the features are currently enabled
- It is **easy and safe** to do though!

Red Hat

uresourced – taking the next step

- Makes current GNOME conform closer to systemd convention (changes will be upstreamed)
- Enables CPU and IO controllers for applications
- Tracks active sessions on graphical seats
- Allocates 250 MiB `memory.low` to the active user (capped at 10 % of system memory)
- Forwards allocation to `session.slice`
  Disables memory controller for children, `memory_recursiveprot`[3] will fix that
- Sets `CPUWeight=500`, `IOWeight=500` for active user

Configure it using `/etc/uresourced.conf`

[3] https://github.com/systemd/systemd/pull/16559

Red Hat

uresourced – what does this mean

- Applications are equal when competing for CPU
- The active user will receive a greater share of CPU
- The core session is protected from thrashing

Red Hat

uresourced – what is problematic

- IO controller is not fully configured
- A new daemon is likely overkill
- Opaque for the Desktop Environment
  (e.g. let DE choose memory allocation)
- Works best with wayland (X server not protected)

⇒ Good start, probably should be superseded eventually

Red Hat

uresourced – what is problematic

- IO controller is not fully configured
- A new daemon is likely overkill
- Opaque for the Desktop Environment
  (e.g. let DE choose memory allocation)
- Works best with wayland (X server not protected)

⇒ Good start, probably should be superseded eventually

Red Hat

uresourced – try it

- Will be shipped in Fedora 33
- On Fedora 32, simply install it:

  ```
  $ sudo dnf install uresourced
  $ sudo systemctl enable uresourced.service
  ```

  and reboot
- Otherwise, install from source:
  `https://gitlab.freedesktop.org/benzea/uresourced`
- You should not notice any difference in most cases

Section 4

Discussion

Red Hat

Discussion

Will systemd-oomd work for the desktop?

- Is PSI sufficient to detect problematic workloads?
- How should we react to problematic workloads?
  - Is it good to simply kill problematic workloads?
  - Should the user have a choice on whether to kill or not?
  - Should we actively contain the problematic workload? (e.g. by setting `memory.high`, `cpu.max`, `io.max`, ...)
- Not aware of sufficient testing

⇒ Hopefully systemd-oomd is good enough!

Red Hat

Discussion

Will systemd-oomd work for the desktop?

- Is PSI sufficient to detect problematic workloads?
- How should we react to problematic workloads?
  - Is it good to simply kill problematic workloads?
  - Should the user have a choice on whether to kill or not?
  - Should we actively contain the problematic workload? (e.g. by setting `memory.high`, `cpu.max`, `io.max`, ...)
- Not aware of sufficient testing

⇒ Hopefully systemd-oomd is good enough!

Red Hat

Discussion

Are we isolating the user session sufficently?

- We use `cpu.weight`, `io.weight` and `memory.low`
- Should give enough guarantees (i.e. CPU time, few and fast page faults)
- Setups may easily be crippled if controllers are not working well
  - Kernel not being ready
    e.g. LUKS, LVM and ext4 are common
  - Insufficient configuration due to lack of systemd features
    e.g. systemd does not set `io.cost.model`[4]

⇒ Kernel features are good, but may not be fully usable!

[4] https://github.com/systemd/systemd/issues/16403

Red Hat

Discussion

Are we isolating the user session sufficently?

- We use `cpu.weight`, `io.weight` and `memory.low`
- Should give enough guarantees (i.e. CPU time, few and fast page faults)
- Setups may easily be crippled if controllers are not working well
  - Kernel not being ready
    e.g. LUKS, LVM and ext4 are common
  - Insufficient configuration due to lack of systemd features
    e.g. systemd does not set `io.cost.model`[4]

⇒ Kernel features are good, but may not be fully usable!

[4] https://github.com/systemd/systemd/issues/16403

Red Hat

Discussion

What else can and should we do?

- Improve interactivity of applications
  e.g. prioritize focused application
- Power saving (can we learn from mobile?)
  e.g. freeze tasks, change timer accuracy, identify problematic applications
- Improved developer tools
- Any other ideas?

Red Hat

Discussion

What else can and should we do?

- Improve interactivity of applications
  e.g. prioritize focused application
- Power saving (can we learn from mobile?)
  e.g. freeze tasks, change timer accuracy, identify problematic applications
- Improved developer tools
- Any other ideas?