Section 1
Motivation
Resource distribution

- CPU time
- Memory (caches, data, ...)
- IO access
Resource distribution – status quo

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

⇒ All processes are usually treated equally
Resource distribution – status quo

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

⇒ All processes are usually treated equally
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
  - ...

Improved power management
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
  - ...

...
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
  - ...
Spinner demo created by David Edmundson (video)
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)
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)
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)

⇒ Reasonably fast
   Effectively ensures the kernel has enough space for (file) caches
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)

⇒ PSI is inherently slow (>10s)
   Good at identifying thrashing workloads
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
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)

⇒ Not effective at protecting graphical session
Thrashing and OOM handling

What do we really need?

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

- Allows managing kernel cgroups
- Desktop Environments were not ready until recently
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
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
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
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
**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.davedmundson.co.uk/blog/modern-process-management-on-the-desktop/
<table>
<thead>
<tr>
<th>Name</th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox</td>
<td>5.0%</td>
<td>405.0 MiB</td>
</tr>
<tr>
<td>KSysGuard</td>
<td>9.0%</td>
<td>88.2 MiB</td>
</tr>
<tr>
<td>Kate</td>
<td>2.0%</td>
<td>63.4 MiB</td>
</tr>
<tr>
<td>Konsole</td>
<td>2.0%</td>
<td>14.0 MiB</td>
</tr>
<tr>
<td>Spotify</td>
<td>2.0%</td>
<td>529.6 MiB</td>
</tr>
<tr>
<td>System Session</td>
<td></td>
<td>59.5 MiB</td>
</tr>
<tr>
<td>session</td>
<td></td>
<td>7.8 MiB</td>
</tr>
<tr>
<td>Application</td>
<td>Usage</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>3.4 GB</td>
<td></td>
</tr>
<tr>
<td>Evolution Calendar</td>
<td>2.2 GB</td>
<td></td>
</tr>
<tr>
<td>Web</td>
<td>567 MB</td>
<td></td>
</tr>
<tr>
<td>Skype</td>
<td>491 MB</td>
<td></td>
</tr>
<tr>
<td>Chromium Web Browser</td>
<td>473 MB</td>
<td></td>
</tr>
<tr>
<td>Telegram Desktop</td>
<td>327 MB</td>
<td></td>
</tr>
<tr>
<td>GNOME Shell</td>
<td>294 MB</td>
<td></td>
</tr>
<tr>
<td>GNOME Clocks</td>
<td>151 MB</td>
<td></td>
</tr>
<tr>
<td>Fractal</td>
<td>105 MB</td>
<td></td>
</tr>
<tr>
<td>Text Editor</td>
<td>101 MB</td>
<td></td>
</tr>
<tr>
<td>Dino</td>
<td>86 MB</td>
<td></td>
</tr>
<tr>
<td>Files</td>
<td>68 MB</td>
<td></td>
</tr>
</tbody>
</table>
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
\(^{2}\)https://gitlab.gnome.org/GNOME/glib/-/merge_requests/1596
systemd – ongoing tasks

- We want to rely on systemd for more purposes	ep.e. 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
\(^2\) https://gitlab.gnome.org/GNOME/glib/-/merge_requests/1596
Section 3

uresourced
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!
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!
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` 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
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
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
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
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
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!
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!
Discussion

Are we isolating the user session sufficiently?

- 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
Discussion

Are we isolating the user session sufficiently?

- 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
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?
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?