

Next-gen resource management

Authors

David Edmundson <kde@davidedmundson.co.uk>

Henri Chain <henri.chain@enioka.com>

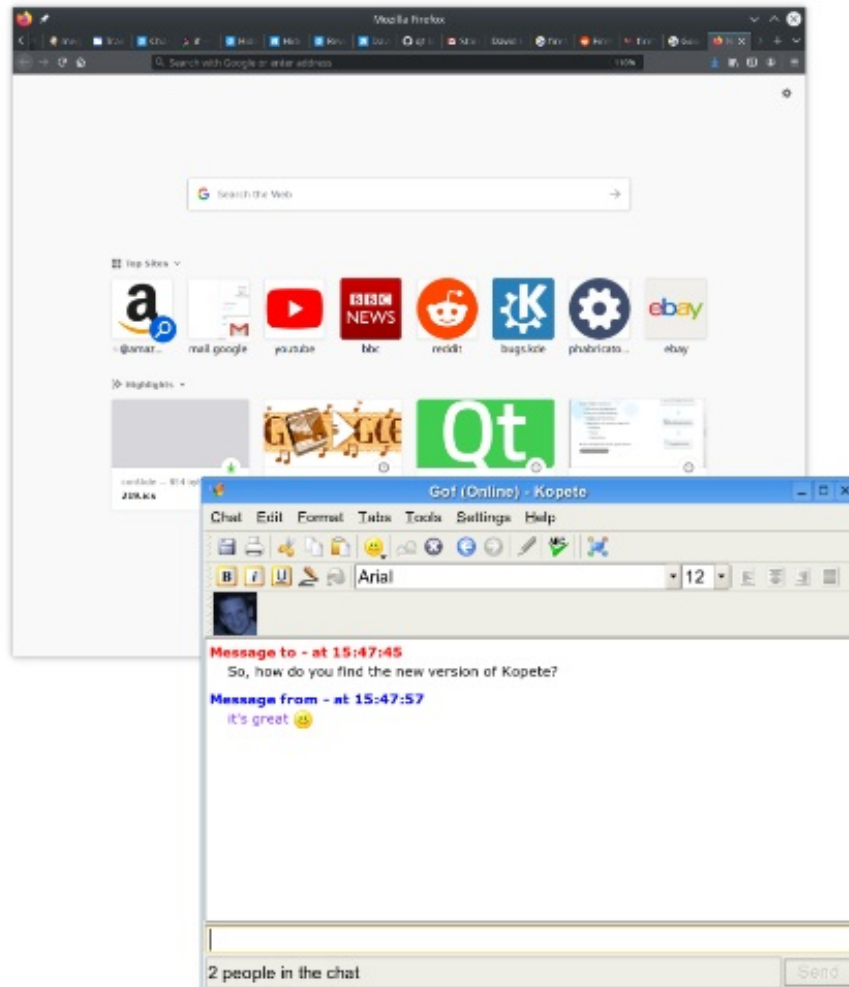
The problem

Managing running processes

Yesteryear

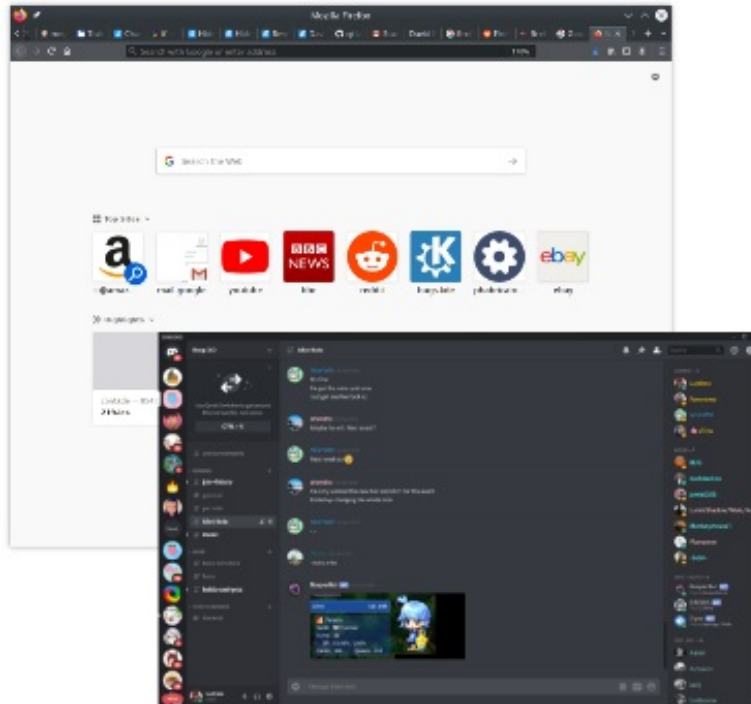
`ps` ux

firefox
kopete



Now

`ps ux` output



```
firefox  
firefox -contentproc -childID 50  
firefox -contentproc -childID 51  
firefox -contentproc -childID 52  
firefox -contentproc -childID 78  
firefox -contentproc -childID 91  
firefox -contentproc -childID 120  
pingsender  
discord  
xdg-dbus-proxy --args=37  
xdg-dbus-proxy --args=37  
discord  
/bin/bash /app/bin/discord  
socat UNIX-LISTEN:/run/user/1000/app/  
com.discordapp.Discord/discord-  
ipc-0,forever,fork UNIX-CONNECT:/run/user/  
1000/discord-ipc-0  
/app/discord/Discord  
/app/discord/chrome-sandbox /app/discord/  
Discord --type=zygote
```

Managing processes

- Understanding what's going on is difficult
- Aggregated resource stats are effectively meaningless

We need some metadata

Fair resource distribution

Being fair

- Discord is 13 processes
- Krita is 1
- A scheduler just sees opaque processes...

We need some metadata

Mapping processes to apps

Mapping processes to apps

- The manager tries to map up windows to .desktop files
- Hoping they report the right things
- We match up audio (by PID) to windows
- With multi processes this is a guessing game

We need some metadata

Solution

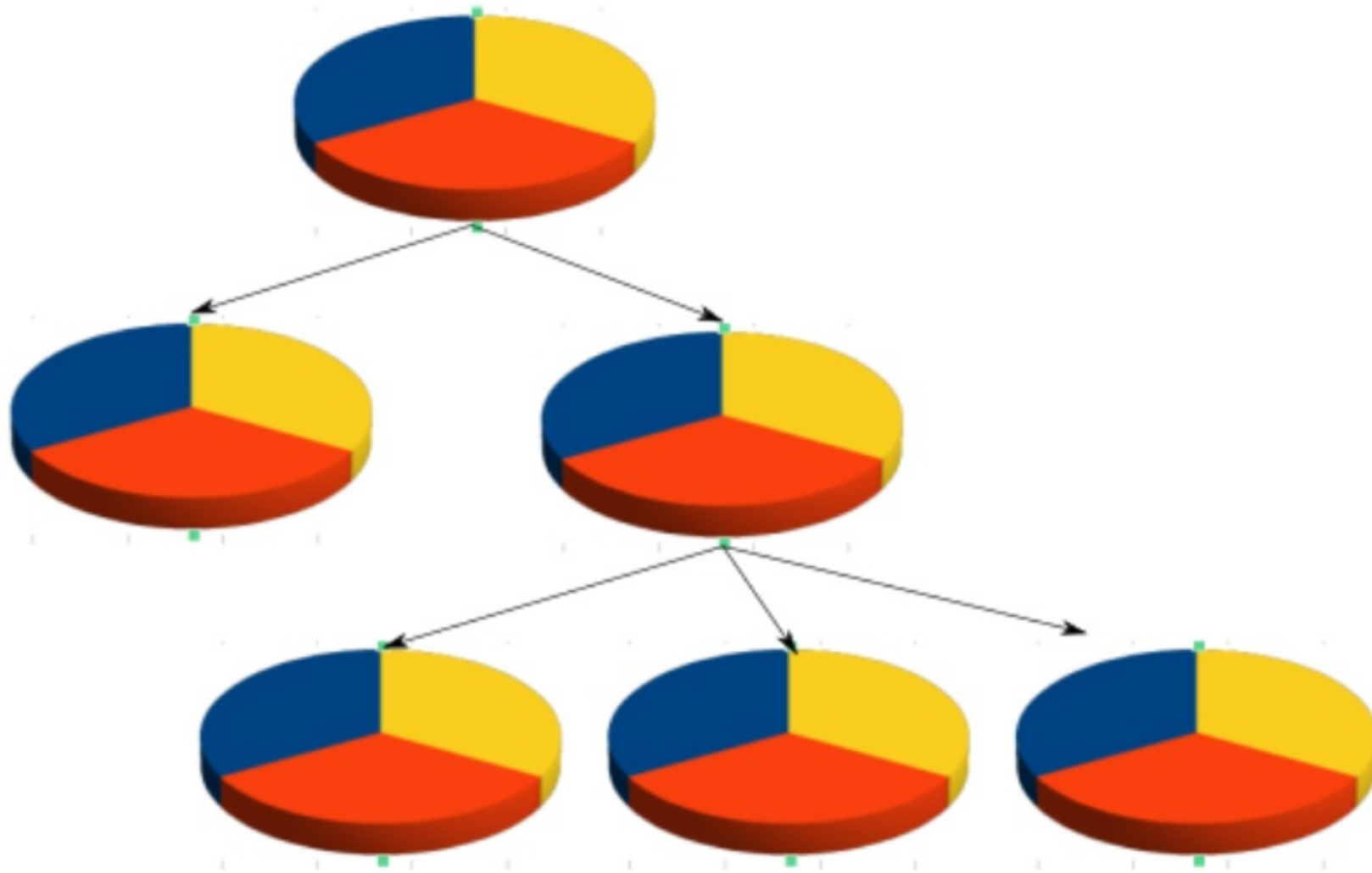
Cgroups

- Partitions the process space
- Hierarchical process groups that can have controllers assigned to them
- Battle tested on servers, container runtimes and flatpak

Resource control

- cpu, mem, io controllers
- Limiting, scheduling, accounting, finer OOM control
- io controller: meant for servers, doesn't work out of the box yet

Weight-based scheduling: better than nice!



Managing Cgroups

- A job for the service manager
- Every systemd unit already runs in its own cgroup
- Systemd's user instance is the owner of the user cgroup tree
- Dbus API
- Drop ins: user and distribution -level configuration

Systemd cgroup tree

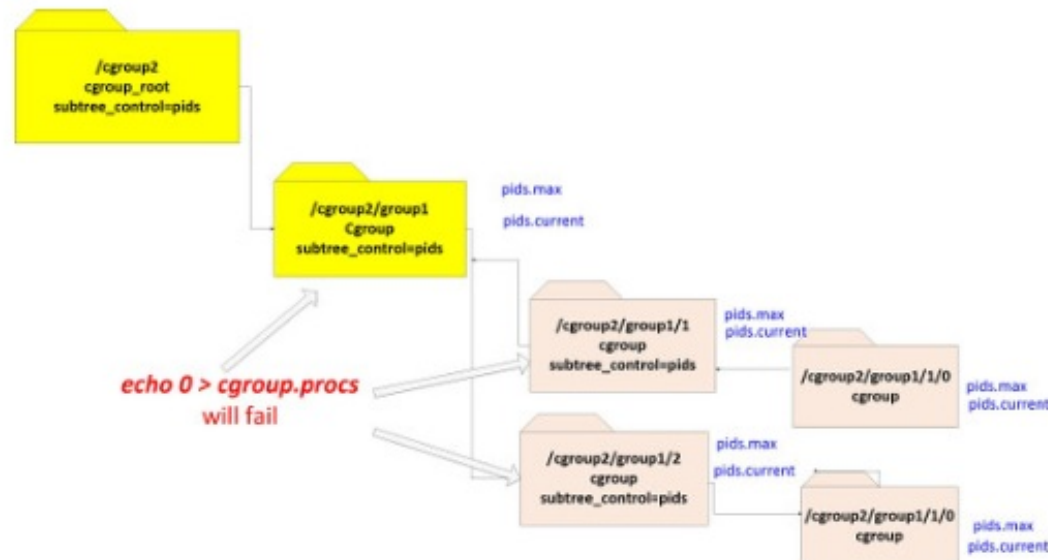
```
$ systemd-cgls
-.slice
├─user.slice
│   └─user-1000.slice
│       ├──user@1000.service
│       │   ├──app.slice
│       │   │   ├──app-org.kde.konsole-86ba3864a585460eba07df01d8d0a514.scope
│       │   │   │   ├──33146 /usr/bin/konsole
│       │   │   │   └─33158 /bin/bash
│       │   │   └─app-firefox-bf4861eb03f54a5f9c92c5da35698f96.scope
│       │   │       ├──2165 /usr/lib/firefox/firefox
│       │   │       └─5215 /usr/lib/firefox/firefox -contentproc -childID 11 -isForBrowse...
│       │   ├──background.slice
│       │   │   └─app-org.kde.krunner.service
│       │   ├──ssh-agent.service
│       │   │   └─1112 /usr/bin/ssh-agent -D -a /run/user/1000/ssh-agent.socket
│       │   └─dbus.service
│       │       └─20233 /usr/bin/ksysguardd
│       └─session-2.scope
│           ├──1103 /usr/lib/sddm/sddm-helper --socket /tmp/sddm-authf5d4f987-45a0-45...
│           └─1122 /usr/bin/startplasma-x11
```

cgroups v2: status of support

- Processes can belong to only one cgroup
- Multiple controllers on a cgroup
- Unified hierarchy: the future, not supported by docker though
- Grouping works in all cases
- Attaching controllers with user systemd not supported for cgroups v1

Slices

- v2 inner leaves cannot contain processes
- Called slices in systemd
- Can also have drop ins



Scopes vs Services

- systemd can also handle transient units
- Scopes - we launch a process, then tag it in a cgroup
- Services - the cgroup controller launches the process

Current rollout

- Plasma 5.19 has every app spawned in it's own cgroup
- Lots of effort needed to put all KDE code through the same API to launch processes
- There are dozens of edge cases to identify and clean up

Current rollout

- Goal is to do everything safely
- No regressions!
- First we deploy us using this
- Then start using this metadata

Work using this

- Linux task scheduler is now using this already
- Taskmanager patches are on review
- libksysguard can surface this information

System Monitor

File View Settings Help

Process Table System Load

End Process... Quick search User Processes Tools

| Name | Username | CPU % | Memory | Shared Mem | Window Title |
|------------------------|----------|-------|--------|------------|--------------|
| xdg-document-portal | david | | 580 K | 5,564 K | |
| xdg-dbus-proxy | david | | 568 K | 4,404 K | |
| xdg-dbus-proxy | david | | 564 K | 4,200 K | |
| flatpak-session-helper | david | | 524 K | 5,352 K | |
| dconf-service | david | | 492 K | | |
| discord | david | | 460 K | | |
| socat | david | | 452 K | | |
| startplasma-dev | david | | 436 K | | |
| agent | david | | 420 K | 4,380 K | |
| xdg-permission-store | david | | 404 K | 4,292 K | |
| bwrap | david | | 384 K | 1,012 K | |
| bwrap | david | | 384 K | 988 K | |
| dbus-daemon | david | | 332 K | 3,216 K | |

76 processes CPU: 8% Memory: 1.5 GiB / 15.3 GiB No swap space available

Memory usage: 524.0 KiB out of 15.3 GiB (0 %)
 RSS Memory usage: 5.7 MiB out of 15.3 GiB (0 %)

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 | Write | Details |
|------|-------------|-----------|----------|--------|------|-------|---------|
| | 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 | | | | | |

CPU

Memory

Network

Disk

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 |
| spotify | | 42.4 MiB |
| spotify | | 73.7 MiB |

Making a standard



Adding more features on top

Unfair resource distribution

Statically assigned resources

Default slices

- Session: kwin, plasmashell, etc
- Apps: firefox, konversation, etc.
- Background: kded, xembedsniproxy, etc.
- Each slice will have a natural "boost" applied

Extra configurability

- Limits can take us further
- An app, distro, or user can add "drop in" to configure individual limits of a background services
- eg limit a file indexer to only 10% CPU ever
- Set OOM stats
- Limit forkbombs

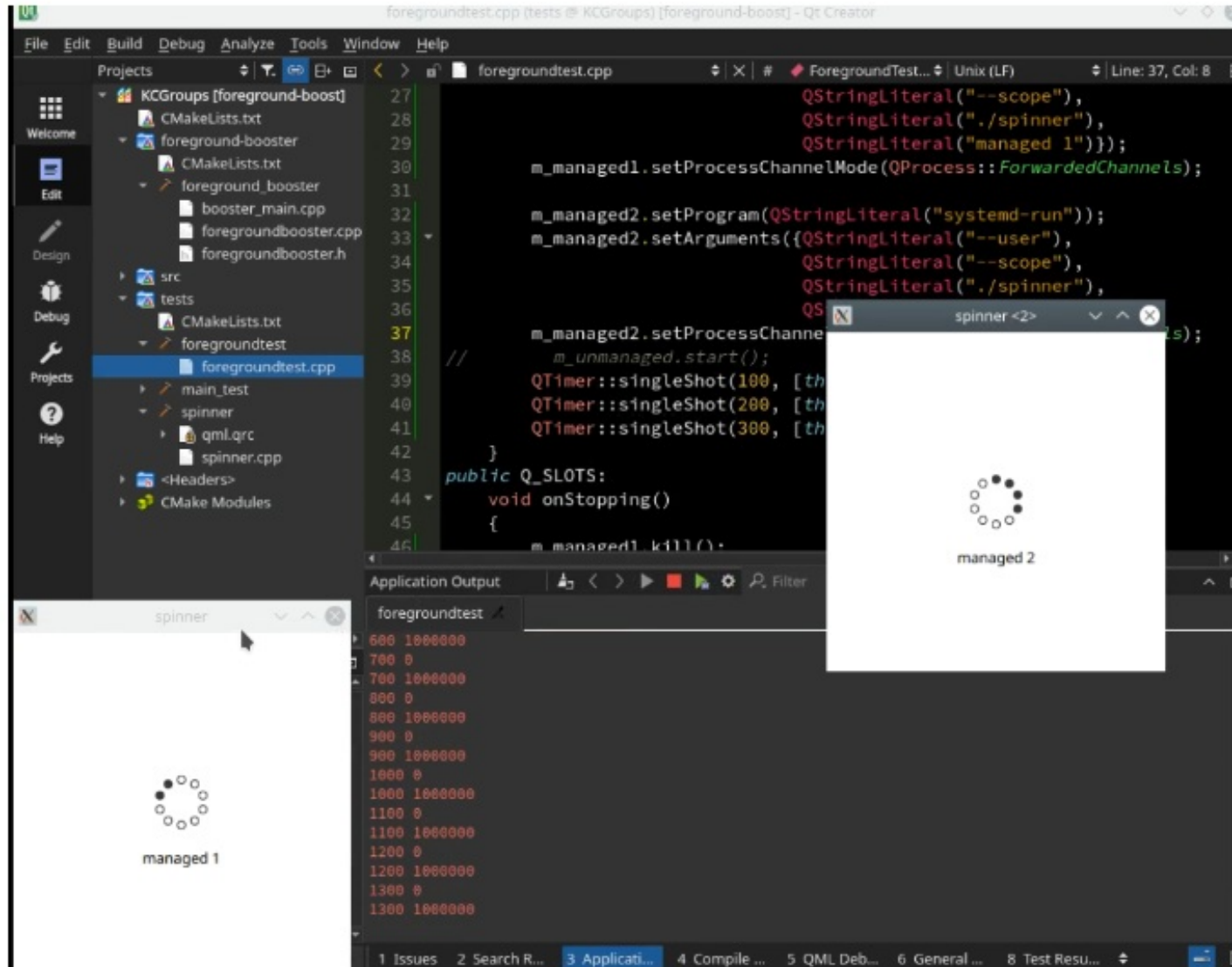
Favouring interactive applications

- We can boost the active application relative to other applications

KCGroups

- Wraps the systemd dbus api
- Exposes resource control
- Fully async, can be used from QML
- Foreground booster demo app
- <https://invent.kde.org/libraries/kcgroups>

Foreground app booster



Move to systemd services

- fully managed by systemd
- Improve logging
- Some drop-in configuration only works with services
- User scopes buggy for old systemd

Move to `systemd` services

- New implementation mostly ready, guarded behind flag
- Due to current `systemd` limitations for non-daemon services
- More work needed to get crash handling to work

Startup

- Have entire plasma session startup be managed by systemd
- Brings all the resource management to the session too
- Logs for key session processes in the right place
- Familiar experience for sysadmins
- Seamless regression free port

Further reading

- Upstream specification - https://systemd.io/DESKTOP_ENVIRONMENTS/
- Code examples - <https://invent.kde.org/snippets/1111>
- Guadec - https://www.youtube.com/watch?v=cmYCM3S_YEY
- Foreground booster demo - https://youtu.be/o6_uq5AQOHg

Q&A