Getting more juice out from your Raspberry Pi GPU

Chema Casanova & Maíra Canal
<jmcasanova@igalia.com> <mcanal@igalia.com>

FOSDEM 2025

Who are we?

- We are open-source developers at Igalia working at the Graphics Team.
- We focus on enhancing the Raspberry Pi graphics stack by refining the Mesa user-space and kernel driver, and optimizing the overall desktop experience.

Chema Casanova
@txenoo@fosstodon.org

Maíra Canal
@mairacanal@fosstodon.org

Raspberry Pi 5

- GPU Broadcom V3D 7.1.7, same VideoCore architecture as RPi 4.
- Higher clock rate than RPi 4, up to 8 Render Targets, better support for subgroup operations, better instruction-level parallelism.
- Driver code merged into existing v3d and v3dv drivers in Mesa 23.3 and Linux Kernel 6.8.
- Same high-level feature support as Raspberry Pi 4.
- Launched October 2023

Raspberry Pi GPU driver stack

HW	GPU	Kernel Driver	Mesa Driver
Raspberry Pi 1-3	Broadcom VideoCore 4	vc4 (display+*render*)	*vc4 (GL/ES)*
Respberry Pi 4/5	Broadcom VideoCore 6/7	vc4 (display) *v3d (render)*	*v3d (GL/ES)* *v3dv (Vulkan)*

User space Mesa3D Drivers

Raspberry Pi 5 GPU graphics APIs

(v3d) OpenGL 3.1 & GLES 3.1

- OpenGL-ES 3.1 conformance since Raspberry Pi 5 product launch.
- Exposes non-conformant Desktop OpenGL 3.1 since 2023.

(v3dv) Vulkan 1.3

- Vulkan 1.3 Conformance since August 2024.
- Vulkan 1.2 at launch.

Performance improvements

- For last year, we focused on performance improvements on GPU limited scenarios using Full-HD target resolution.
- We have analyzed the performance of V3D using several GLES gfxbench traces, and we have achieved an average of **~103.44%** FPS improvement in these scenarios during the last year of Mesa development.
- All these performance optimizations are available in stable Mesa 24.3.

Benchmarking scenario

- **Hardware:** Raspberry Pi 5 8Gb (V3D 7.1 GPU)
- **SO:** Android 15
- **Kernel:** Linux 6.6
- **Benchmark:** GFXBench 5.0
- **Display:** Resolution 1920x1032

- **2023:** Mesa 23.3.2 (2023-12-27)
- **2024:** Mesa 25.0.0-devel (2024-12-31)

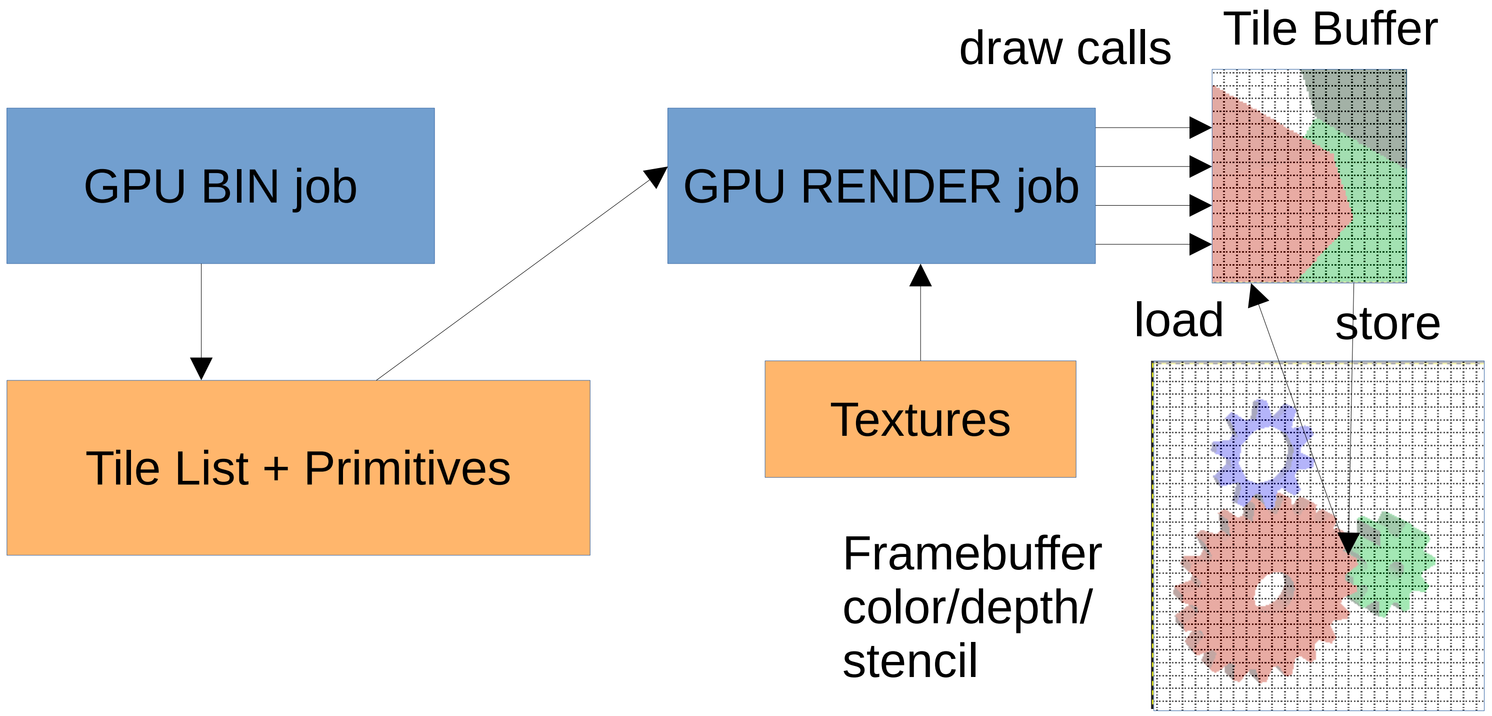
Performance improvements

Version: 5.1.1

High-Level Tests

	Test	Before	After	Improvement
	Aztec Ruins OpenGL (High Tier) ES 3.1 | V3D 7.1 | 1920 x 1032	142.8* Frames (2.2 Fps)	274.9* Frames (4.3 Fps)	**+92.50%**
	Aztec Ruins OpenGL (Normal Tier) ES 3.1 | V3D 7.1 | 1920 x 1032	231.3* Frames (3.6 Fps)	422.8* Frames (6.6 Fps)	**+82.79%**
	Manhattan 3.1 ES 3.1 | V3D 7.1 | 1920 x 1032	304.0 Frames (4.9 Fps)	517.0 Frames (8.3 Fps)	**+70.06%**
	Manhattan ES 3.0 | V3D 7.1 | 1920 x 1032	251.6 Frames (4.1 Fps)	832.5 Frames (13 Fps)	**+230.88%**
	T-Rex ES 2.0 | V3D 7.1 | 1920 x 1032	943.7 Frames (17 Fps)	1,330 Frames (24 Fps)	**+40.93%**

Tiled-based rendering



Reduce number of job flushes

- We identified that v3d was being too conservative during the implementation of ARB_texture_barrier as the driver passed all the tests with an empty implementation.
- v3d was flushing jobs that wrote to a resource that was going to be sampled.
- But there is no need in cases where the job reading the resource is the same one that was writing to it, as updates already are available in the cache.
- Merging draw calls in the same GPU jobs avoids extra loads/stores of the tile buffer and provides a significant performance improvement (**+40,39%**)

  c1: "v3d: Only flush jobs that write texture from different job submission."


2023
17 FPS
2024
24 FPS (+40.93%)
igalia
T-Rex | ES 2.0 | 1920x1032

Compiler backend optimizations

- We have implemented multiple compiler optimizations, reducing the total number of instructions more than 4%. And an average FPS improvement of **+3.57%**

total instructions in shared programs: 630354 -> 604028 (-4.18%)
instructions in affected programs: 572837 -> 546511 (-4.60%)

Avoid load/stores on invalidated framebuffers

- With the information of the invalidated framebuffers we can avoid the stores of the results of tile buffer rendering and the next load if they re-used in following jobs as any read value would be undefined.

- This gets us a **+1.1%** FPS Improvement

  c2: "v3d: avoid load/store of tile buffer on invalidated framebuffer"

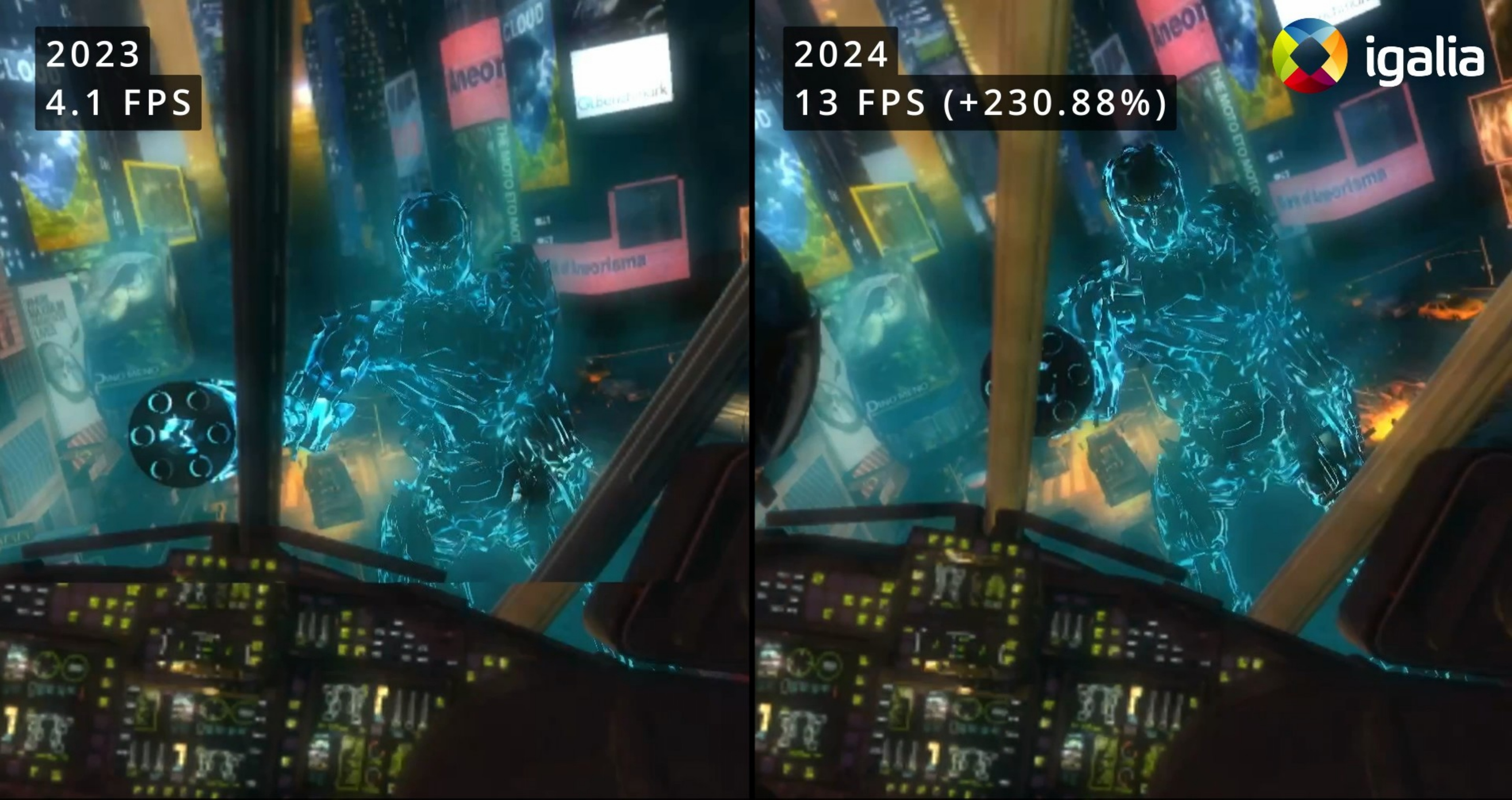
Take advantage of Early-Z optimization

- Early-Z optimization was disabled when there is a discard instruction in the draw call shader. But we can enable it at draw time if depth updates are disabled and there are no occlusion queries active.

- This got us an average performance improvement of **+14,87%**
  c3: “v3d: Enable Early-Z with discards when depth updates are disabled”

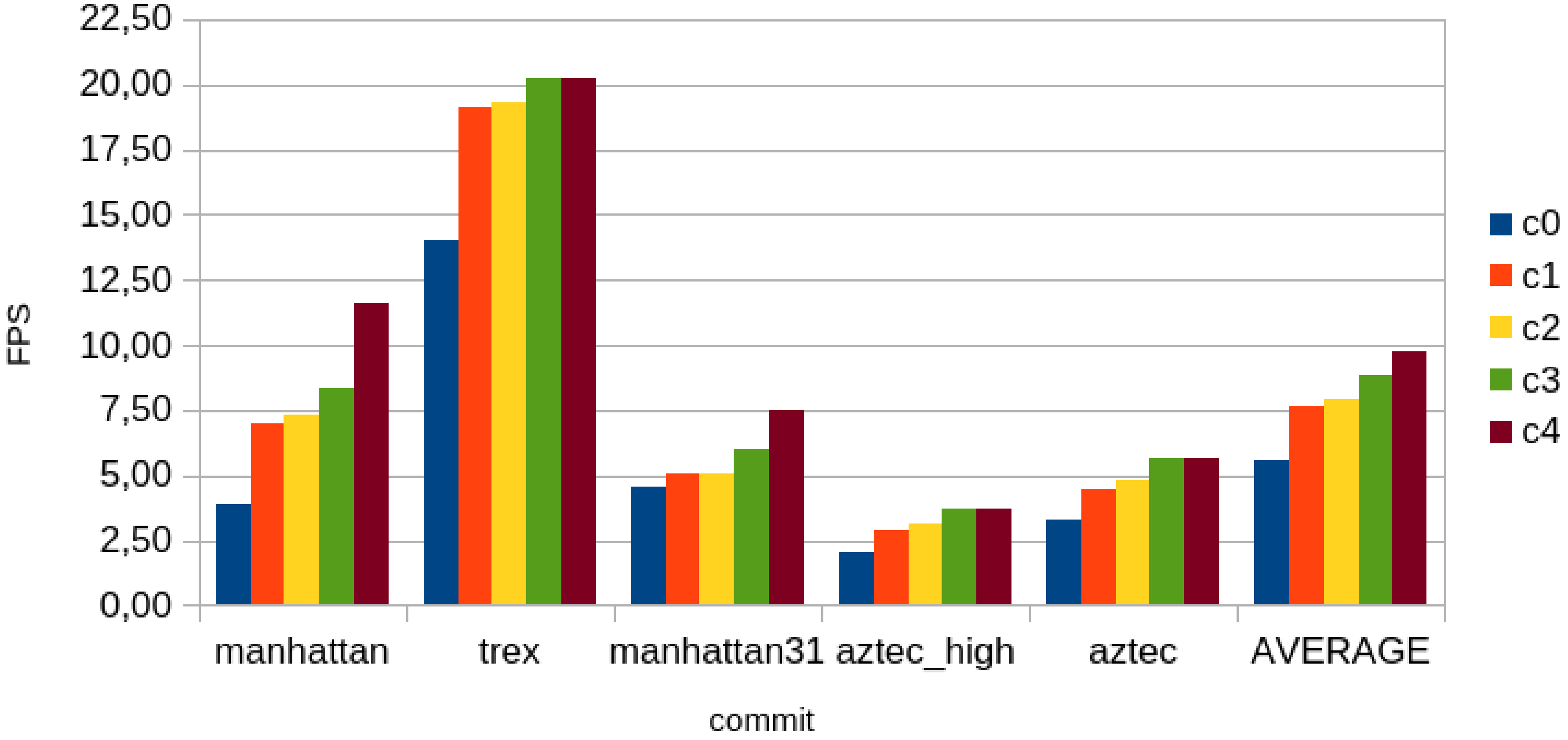
Avoid loads/stores with disabled rasterization

- If all draw calls submitted have the rasterizer discard enabled, we can avoid any tile buffer load/stores.
- This is specially helpful in scenarios where transform feedback is used, because the application is only interested in the geometry results.
- Test gets another **+12.58%** average performance improvement, but mainly affecting manhattan demos. manhattan (+38.62%) manhtattan31 (+24,46%)
  c4: “v3d: Don't load/store if rasterizer discard is enabled”

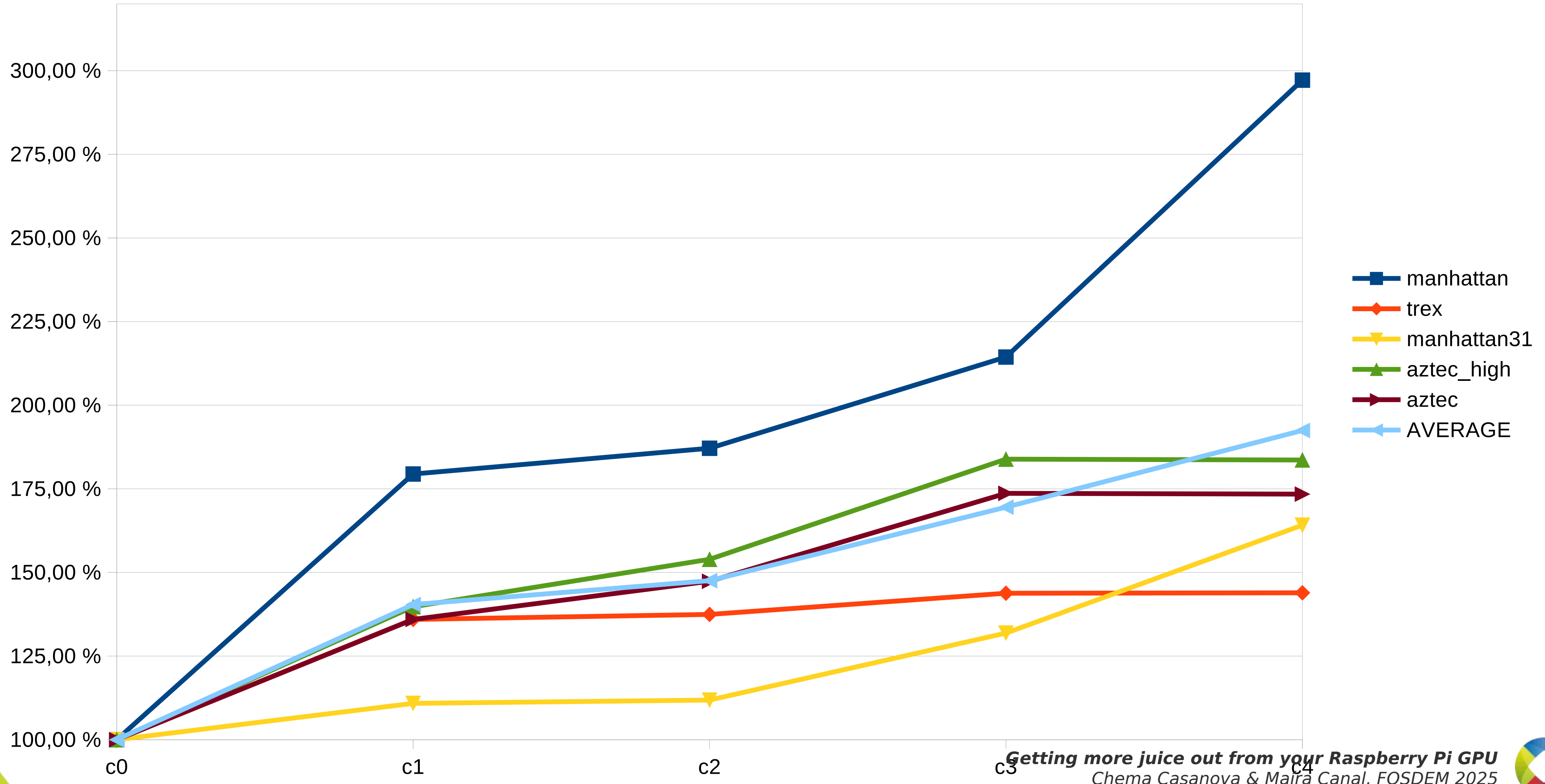


Manhattan | ES 3.0 | 1920x1032

FPS over time per benchmark



FPS improvement over time



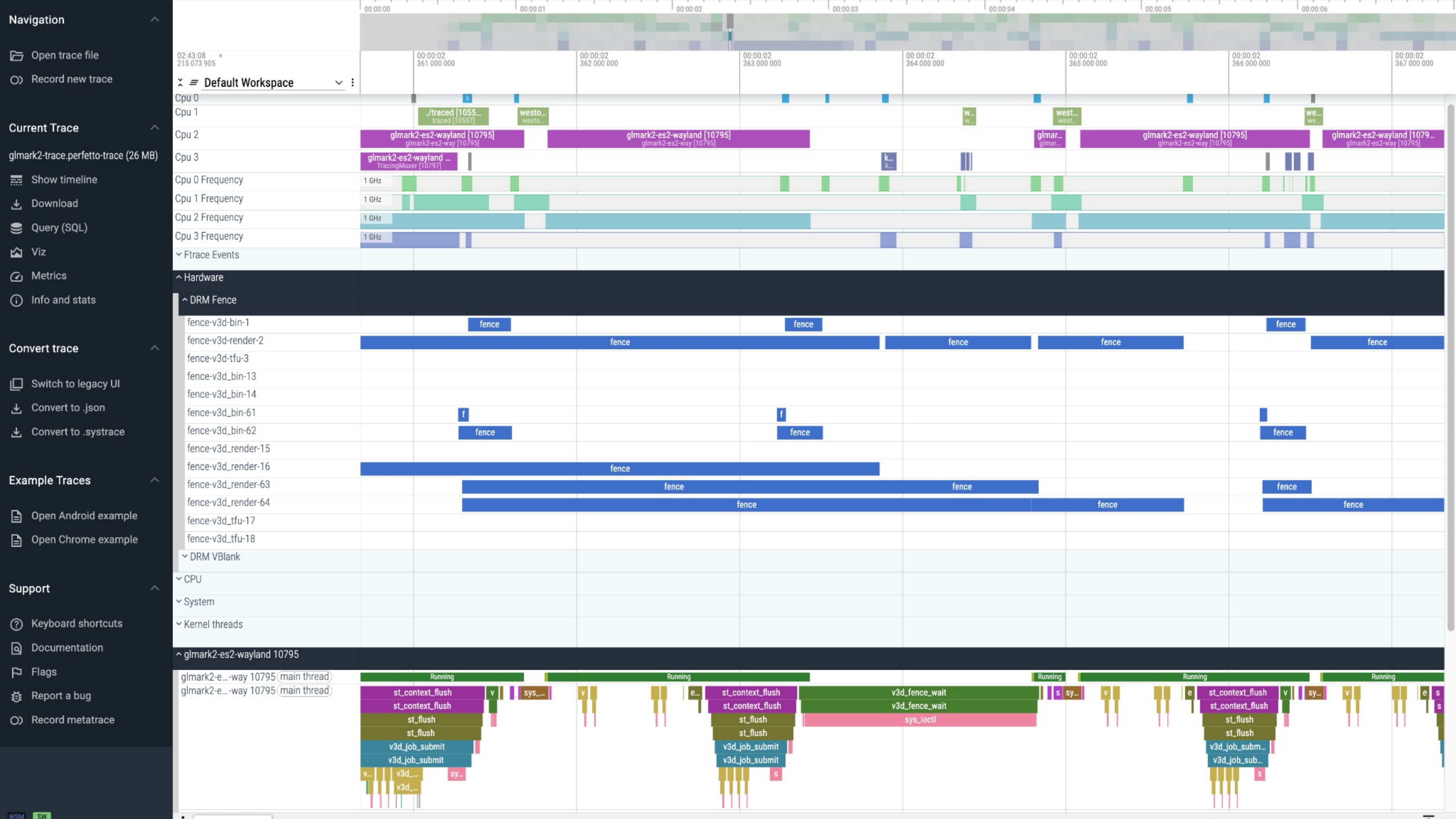
Performance Measurement Tools

CPU jobs and Timestamp Queries

- **FOSDEM 2024:** Some Vulkan commands cannot be performed by the GPU alone → CPU jobs
  - Moved CPU jobs to kernel space to avoid GPU flushes and CPU stalls.
  - Landed timestamp queries (and others) in V3DV.
- **Now:** The V3D GL driver also has support for timestamp queries on next Mesa 25.0
  - *GL_ARB_timer_query*
- **Usage:** Identify driver bottlenecks with timestamps accurately synchronized to the graphics pipeline.

Perfetto Support

- **Perfetto:** Open-source stack for performance instrumentation.
  - Records system-level and app-level traces collecting data from several data-sources (e.g. Ftrace) → Mesa data-sources
- **Mesa Perfetto:** Introduces additional producers for GPU performance visualization (frequency, utilization, performance counters, etc.) on a unified timeline for improved system-level performance tuning and debugging.
- **V3D Support:** Perfetto Data Source (!31751), CPU tracepoints (!31575, !33012)



Navigation
Open trace file
Record new trace
Current Trace
glmark2-trace.perfetto-trace (26 MB)
Show timeline
Download
Query (SQL)
Viz
Metrics
Info and stats
Convert trace
Switch to legacy UI
Convert to .json
Convert to .systrace
Example Traces
Open Android example
Open Chrome example
Support
Keyboard shortcuts
Documentation
Flags
Report a bug
Record metatrace
Default Workspace
Cpu 0
Cpu 1
Cpu 2
Cpu 3
Cpu 0 Frequency
Cpu 1 Frequency
Cpu 2 Frequency
Cpu 3 Frequency
Ftrace Events
Hardware
DRM Fence
fence-v3d-bin-1
fence-v3d-render-2
fence-v3d-tfu-3
fence-v3d_bin-13
fence-v3d_bin-14
fence-v3d_bin-61
fence-v3d_bin-62
fence-v3d_render-15
fence-v3d_render-16
fence-v3d_render-63
fence-v3d_render-64
fence-v3d_tfu-17
fence-v3d_tfu-18
DRM VBlank
CPU
System
Kernel threads
glmark2-es2-wayland 10795
glmark2-e...-way 10795 main thread
glmark2-es2-wayland [10795]
./traced [1055...
fence
Running
st_context_flush
st_flush
v3d_job_submit
v3d_fence_wait
sys_ioctl

Kernel Work

Super Pages

- V3D GPU has support for 4KB, 64KB "Big Pages", and 1MB "Super Pages" pages.
  - Contiguous memory blocks + Page table entries
- Linux driver didn't support Big or Super Pages → Unused hardware feature
- **Potential Benefit:** Improve performance by reducing MMU fetches, benefiting memory-intensive applications using large buffer objects (BOs).
- **The issue?** Allocating a contiguous block of memory using shmem.
- Let's check how we solved this problem and landed support in **6.13**.

Upstream first! All our kernel work is available in the mainline kernel since day 1.

Using THP for Super Pages

- By default, tmpfs/shmem only allocates memory in PAGE_SIZE chunks.
- **Our solution:** Create a new tmpfs mountpoint with `huge=within_size`.
  - Use **Transparent Huge Pages (THP)** to manage large memory pages.
- With the contiguous block of memory, it's only a matter of placing the PTEs.
  - 16 4KB pages (for big pages) or 256 4KB pages (for super pages)
- Reduce the VA alignment to 4KB (↓ memory pressure)

Using THP for Super Pages

- Average performance improvement of **1.33%** running GL and Vulkan traces and significant **performance boost** in some emulation use cases.
  - "*Embedded systems should enable hugepages only inside madvise regions to eliminate any risk of wasting any precious byte of memory and to only run faster.*" from [Transparent Hugepage Support — The Linux Kernel documentation]()
- You can test it in Linux 6.13 with CONFIG_TRANSPARENT_HUGEPAGE enabled!

G: 3.75 [P] | V: 3.75
266.56ms (288.56ms worst)
EE: 9.5% (25.39ms)
GS: 99.4% (264.98ms)
VU: 8.1% (21.55ms)
GPU: 15.7% (41.95ms)

G: 8.73 [P] | V: 8.73
114.60ms (129.32ms worst)
EE: 17.4% (19.89ms)
GS: 99.1% (113.52ms)
VU: 17.8% (20.41ms)
GPU: 36.8% (42.17ms)

G: 3.75 [P] | V: 3.75
266.56ms (288.56ms worst)
EE: 9.5% (25.39ms)
GS: 99.4% (264.98ms)
VU: 8.1% (21.55ms)
GPU: 15.7% (41.95ms)

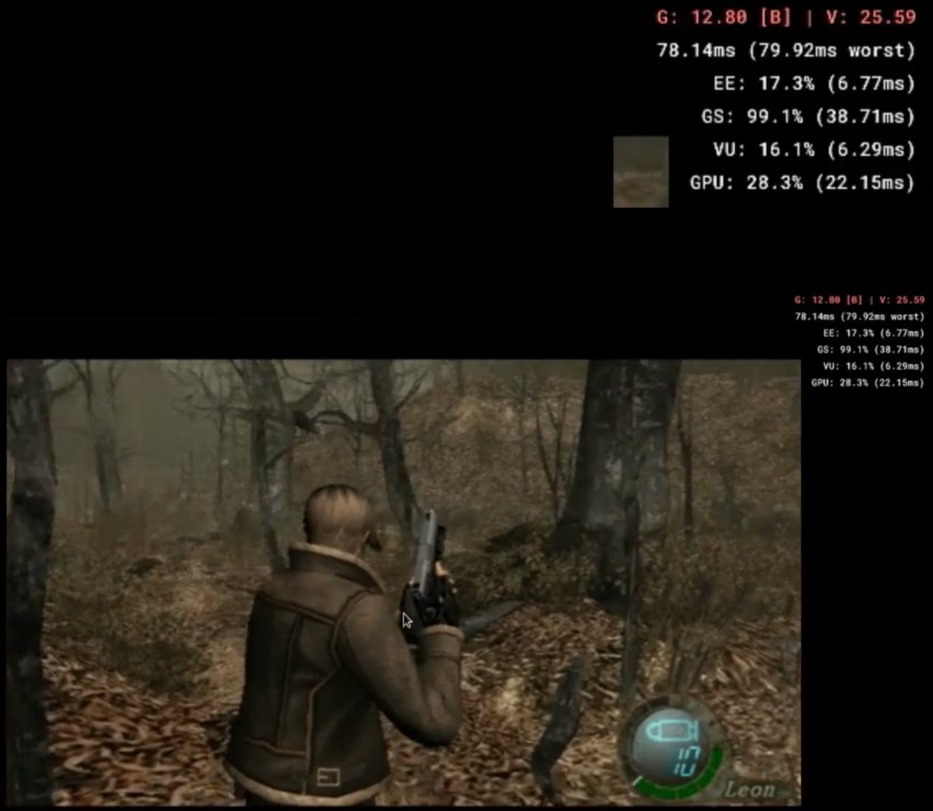
Without Super Pages

G: 8.73 [P] | V: 8.73
114.60ms (129.32ms worst)
EE: 17.4% (19.89ms)
GS: 99.1% (113.52ms)
VU: 17.8% (20.41ms)
GPU: 36.8% (42.17ms)

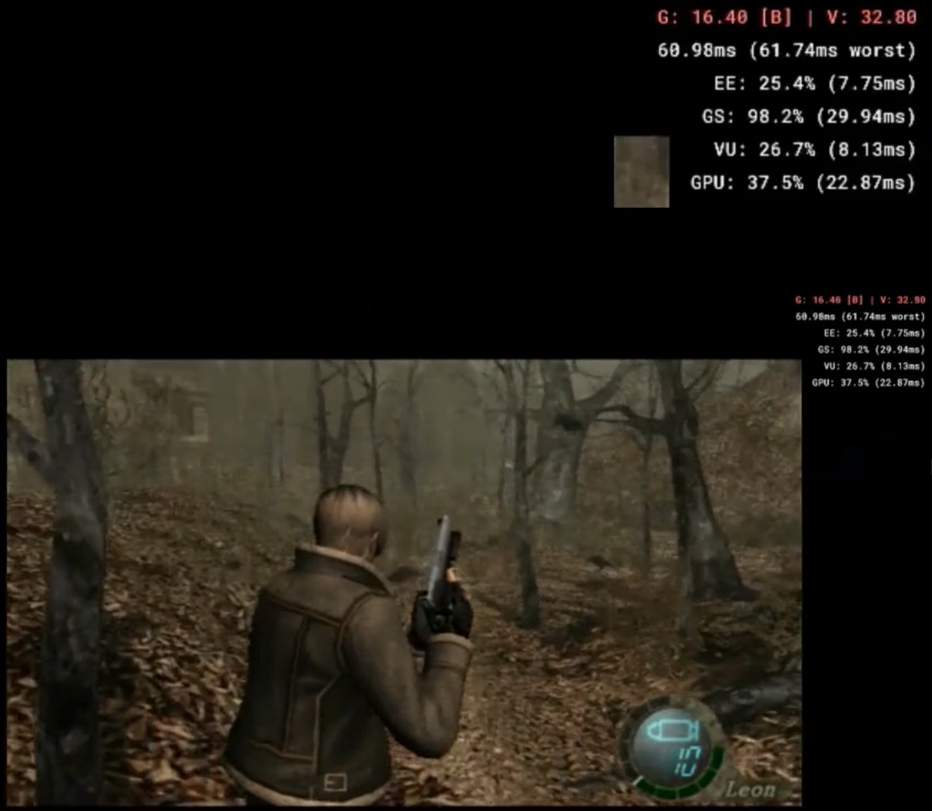
With Super Pages

Burnout 3: Takedown (PS2)

igalia



Without Super Pages



With Super Pages

Resident Evil 4 (PS2)

igalia

Tailoring THP

- **Our interest:** 4KB, 64KB, and 1MB blocks of contiguous memory.
  - But, THP uses huge pages of PMD-size (2MB for ARM64) → Unneeded memory fragmentation
- **Our solution:** Using multi-size THP (mTHP) to allow huge pages from 64KB up to 1MB.
  - mTHP introduces the ability to allocate memory in blocks that are bigger than a base page but smaller than traditional PMD-size.
- We created two kernel parameters to ease mTHP configuration on shmem: `transparent_hugepage_shmem=` and `thp_shmem=`.

Tailoring THP

```
// <policy> = always,never,within_size,advise
transparent_hugepage_shmem=<policy>

// different policies for different page sizes
// <policy> = always,inherit,never,within_size,advise
thp_shmem=16K-64K:always;128K,512K:inherit;256K:advise;1M-2M:neve
r;4M-8M:within_size
```

Questions?

Getting more juice out from your Raspberry Pi GPU

Chema Casanova & Maíra Canal
<jmcasanova@igalia.com> <mcanal@igalia.com>

FOSDEM 2025