A Rusty Odyssey: A Timeline of Rust in the DRM subsystem

Maíra Canal <mcanal@igalia.com>

Kernel Recipes 2025 (Paris, France)

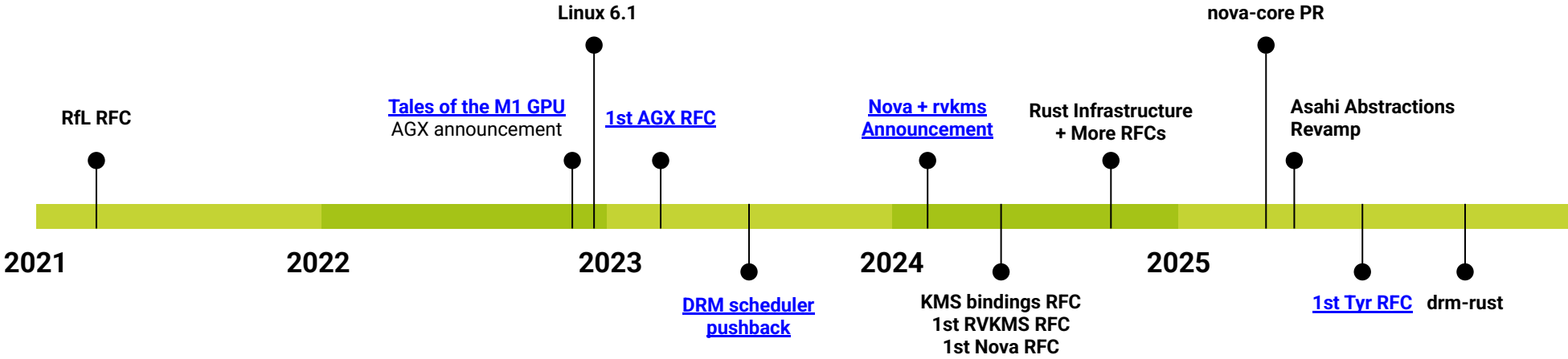
igalia

Agenda

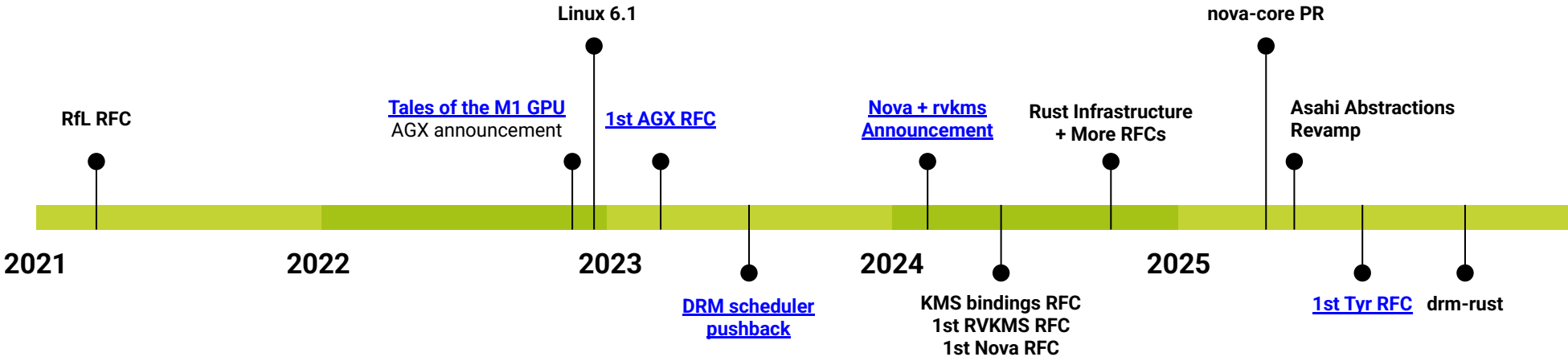
- **Timeline Review:** sometimes, we can't have a good perception when living through events in real-time.
- **Kernel scale:** many subsystems; partial visibility.
- **Noise vs. Trend:** in between RFCs, talks, and heated arguments, what's the current status?
- **Goal:** take a step back; review what has been going on the DRM in last few years.
    - Focus on GPU drivers specifically.

Brief Timeline

Reviewing the last few years

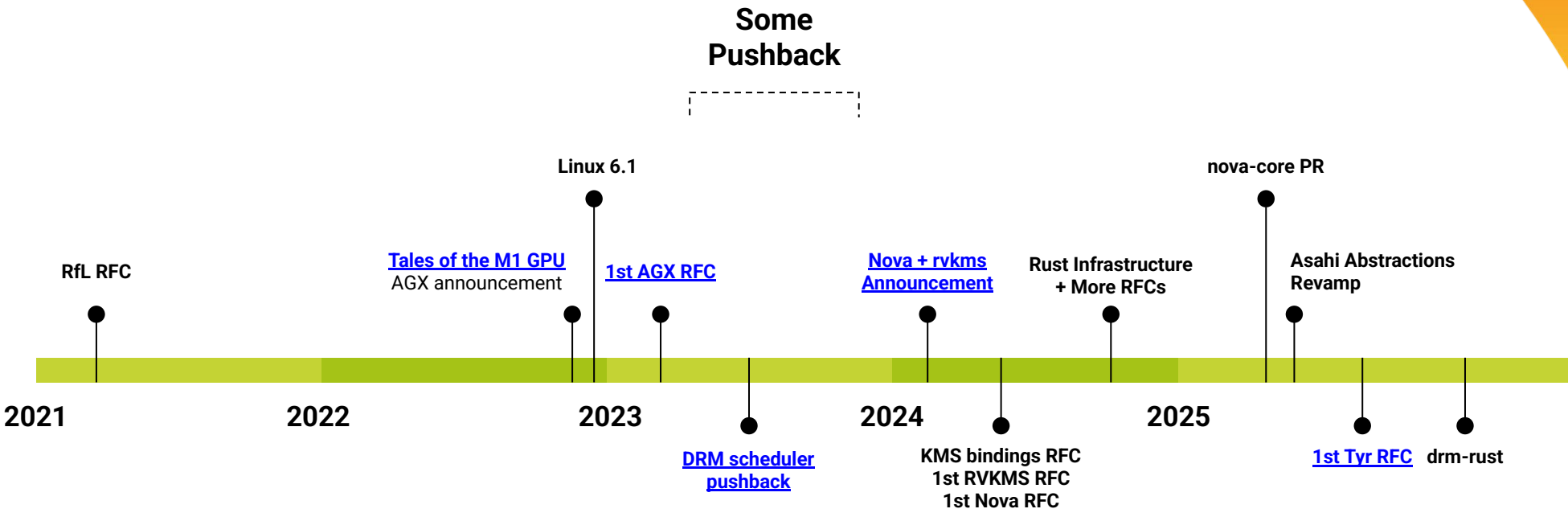

Linux 6.1
nova-core PR
RfL RFC
Tales of the M1 GPU
AGX announcement
1st AGX RFC
Nova + rvkms
Announcement
Rust Infrastructure
+ More RFCs
Asahi Abstractions
Revamp
2021
2022
2023
2024
2025
DRM scheduler
pushback
KMS bindings RFC
1st RVKMS RFC
1st Nova RFC
1st Tyr RFC
drm-rust

The Asahi Era



The Asahi Era

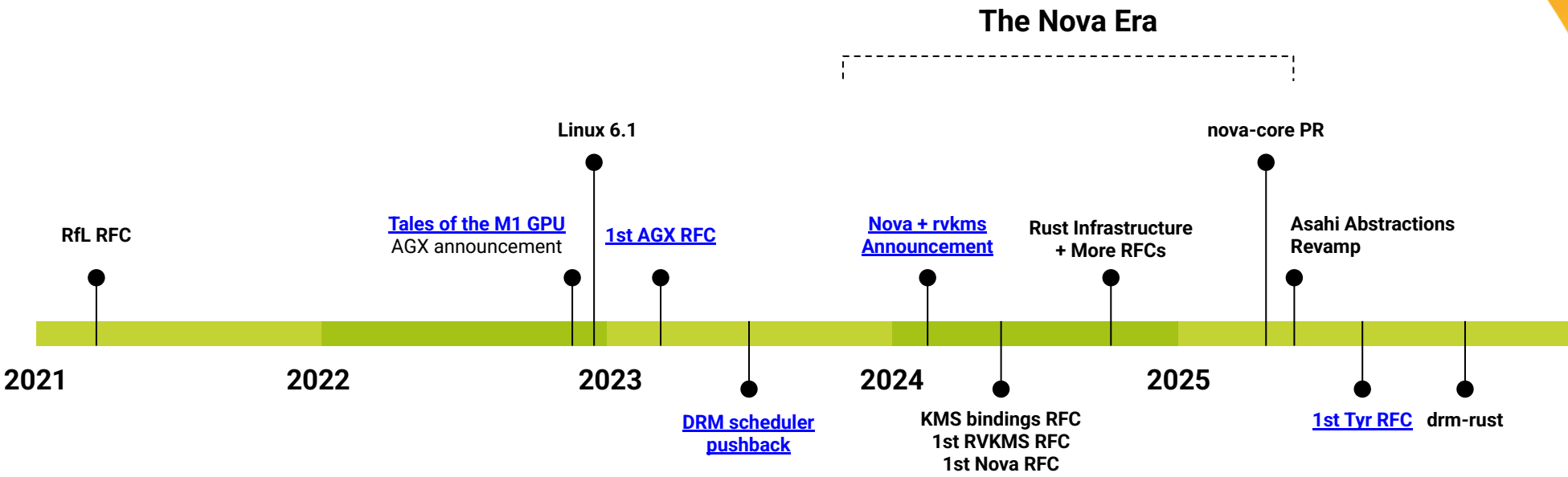
- A reverse-engineered driver for Apple M1, M1 Pro, M1 Max, M1 Ultra, M2, M2 Pro, M2 Max, and M2 Ultra GPUs across a couple of **firmware** revisions each.
- Developed by Asahi Lina between 2022 and 2023.
- 1st GPU driver written in Rust + Development of several safe abstractions for DRM structures.
- RFC was sent in March 2023. The expectation was "get this upstream for 6.5, and the driver for 6.6."
- **Current Status**: Not Upstreamed - Maintained out-of-tree


Some Pushback
Linux 6.1
nova-core PR
RfL RFC
Tales of the M1 GPU
AGX announcement
1st AGX RFC
Nova + rvkms Announcement
Rust Infrastructure + More RFCs
Asahi Abstractions Revamp
2021
2022
2023
2024
2025
DRM scheduler pushback
KMS bindings RFC
1st RVKMS RFC
1st Nova RFC
1st Tyr RFC
drm-rust

Lifetimes from C to Rust

1. Model C's rules first
   - ❏ Who owns/frees?
   - ❏ How long is the pointer valid?
   - ❏ Can it alias? Can it mutate?
   - ❏ Is it *thread-safe*?
2. Encode them in types: lifetimes, `Drop`, `PhantomData`, `NonNull`, typestates, `Send`/`Sync`, etc.
3. Document Safety assumptions

The DRM Scheduler


The Nova Era
Linux 6.1
nova-core PR
RfL RFC
Tales of the M1 GPU
AGX announcement
1st AGX RFC
Nova + rvkms
Announcement
Rust Infrastructure
+ More RFCs
Asahi Abstractions
Revamp
2021
2022
2023
2024
2025
DRM scheduler
pushback
KMS bindings RFC
1st RVKMS RFC
1st Nova RFC
1st Tyr RFC
drm-rust

The Nova Era

- Two drivers (Nova and RVKMS) developed by Red Hat and NVIDIA developers [1] since 2023.
- Nova is a driver for GSP (GPU system processor) based Nvidia GPUs (soon to be, successor of Nouveau).
  - Nouveau was mostly designed for pre-GSP hardware.
  - **Nova Status:** A stub driver was merged in mainline / Still In Development
- RVKMS is a port of the virtual KMS (VKMS) driver to Rust.
  - **RVKMS Status:** Not upstreamed / Still In Development

[1] Danilo Krummrich (Red Hat), Lyude Paul (Red Hat), Alexandre Courbot (NVIDIA), Joel Fernandes (NVIDIA), Timur Tabi (NVIDIA), etc.

drivers/gpu/nova-core

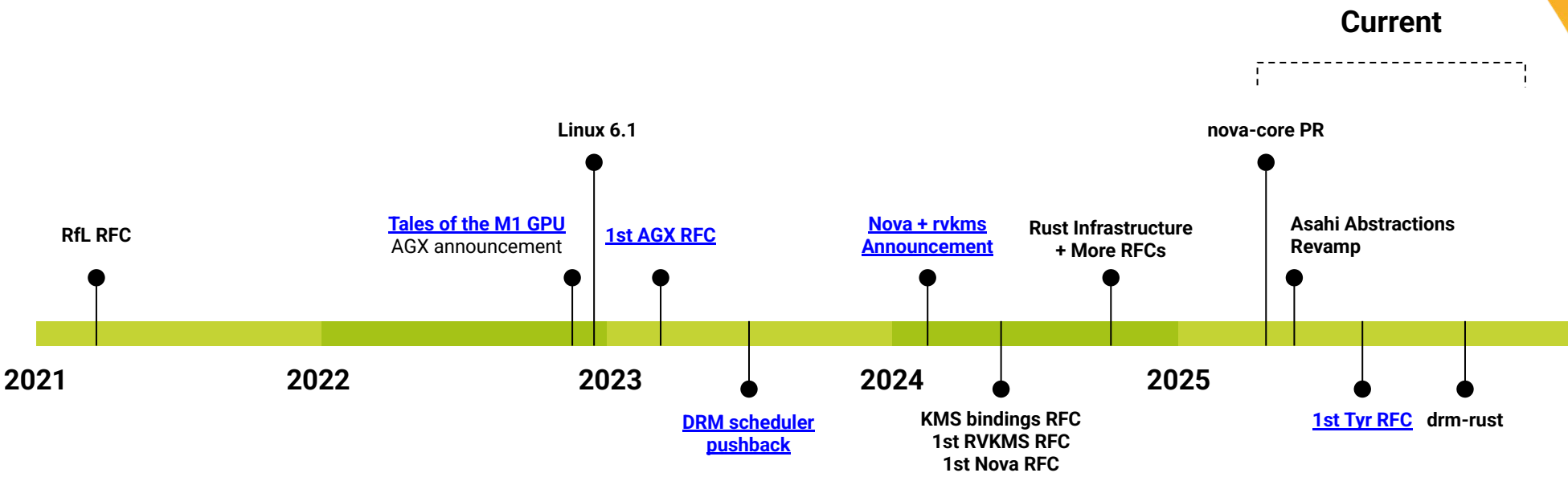
```
├── dma.rs
├── driver.rs
├── falcon
├── falcon.rs
├── fb
├── fb.rs
├── firmware
├── firmware.rs
├── gfw.rs
├── gpu.rs
├── Kconfig
├── Makefile
├── nova_core.rs
├── regs
├── regs.rs
├── util.rs
└── vbios.rs
```

drivers/gpu/drm/nova

```
├── driver.rs
├── file.rs
├── gem.rs
├── Kconfig
├── Makefile
├── nova.rs
└── uapi.rs
```

Nova Driver

- Status in Linux 6.17
- Merging smaller bits, working with the community and growing with upstream.


Current
Linux 6.1
nova-core PR
RfL RFC
Tales of the M1 GPU
AGX announcement
1st AGX RFC
Nova + rvkms
Announcement
Rust Infrastructure
+ More RFCs
Asahi Abstractions
Revamp
2021
2022
2023
2024
2025
DRM scheduler
pushback
KMS bindings RFC
1st RVKMS RFC
1st Nova RFC
1st Tyr RFC
drm-rust

Current Status

From the "Rust for Linux" talk in FOSDEM 2025 (Feb 1, 2025):

*"**2025 will be the year of Rust GPU drivers.** I am confident that a lot will be achieved once the main abstractions are in place, and so far, progress has been steady, with engineers from multiple companies joining together to tackle each piece of the puzzle."*

Daniel Almeida (Tyr Kernel Driver Maintainer)

Current Status

- Daniel Almeida and Alice Rhyl announced Tyr, Rust-based DRM driver for CSF-based Arm Mali GPUs, in June → Stub driver was queued for 6.18.
- Several of Lina's abstractions are being revamped, improved, and merged.
- More and more functionality is being added to Nova and Tyr.
- Can I run PanVK with Tyr on my Rock 5B? Can I run NVK with Nova on my GeForce RTX20?
    - Not quite there yet.

Current Status

- Multiple efforts to improve the DRM scheduler with documentation and correction of UB and leaks + Fair DRM Scheduler
  - However, a new scheduler is being developed in Rust.
- A new drm-rust tree was added recently to the drm repo.

Why Rust in the DRM subsystem?

What does Rust do that C doesn't?

Many benefits...

- That you're probably already familiar with it.
- Ownership and lifetime model are great for asynchronous systems (as GPUs).
- APIs that catch logic errors at compile time.
- **But why is it so suitable for GPU drivers?**

GPU architecture is evolving

- More and more GPUs have dedicated coprocessors inside of it.
- Communication with the GPU happens through those processors via firmware.
- Scheduling, power management, context switching, security, ... it's all being progressively offloaded from the CPU.
- **NVIDIA:** GPU System Processor (GSP) (RISC-V core inside the GPU)
- **Apple:** ASC (likely, Apple Silicon Coprocessor) (ARM64 core running RTKit)
- **ARM Mali:** Command Stream Frontend (CSF)
- **AMDGPU:** SMC/SMU, PSP, SDMA...

Unstable Firmware ABIs

- GPUs use proprietary firmwares →Unstable ABIs
- **How to deal with unstable ABIs?**
  - In C: file duplication; function duplication; or complicated code generation with macros (aka "macro hell").

```
struct psp_firmware_header_v1_1 psp_v1_1;
struct psp_firmware_header_v1_2 psp_v1_2;
struct psp_firmware_header_v1_3 psp_v1_3;
struct psp_firmware_header_v2_0 psp_v2_0;
struct psp_firmware_header_v2_0 psp_v2_1;
```

(Extracted from AMDGPU)

```
struct rlc_firmware_header_v1_0 rlc;
struct rlc_firmware_header_v2_0 rlc_v2_0;
struct rlc_firmware_header_v2_1 rlc_v2_1;
struct rlc_firmware_header_v2_2 rlc_v2_2;
struct rlc_firmware_header_v2_3 rlc_v2_3;
struct rlc_firmware_header_v2_4 rlc_v2_4;
```


```
pub(crate) struct RunFragment<'a> {
    pub(crate) tag: workqueue::CommandType,

    #[ver(V >= V13_0B4)]
    pub(crate) counter: U64,

    pub(crate) vm_slot: u32,
    pub(crate) unk_8: u32,
    pub(crate) microsequence: GpuPointer<'a, &'a [u8]>,
    pub(crate) microsequence_size: u32,

    [...]

    #[ver(G < G14X)]
    pub(crate) job_params1: JobParameters1::ver<'a>,
    #[ver(G < G14X)]
    pub(crate) job_params2: JobParameters2,
    #[ver(G >= G14X)]
    pub(crate) registers: job::raw::RegisterArray,

    pub(crate) job_params3: JobParameters3::ver,

    [...]

    #[ver(V >= V13_3)]
    pub(crate) unk_v13_3: u32,
    pub(crate) meta: job::raw::JobMeta,
    pub(crate) unk_after_meta: u32,
    pub(crate) unk_buf_0: U64,
    pub(crate) unk_buf_8: U64,
    pub(crate) unk_buf_10: U64,
    pub(crate) command_time: U64,

    [...]

    #[ver(V >= V13_0B4)]
    pub(crate) unk_ts: U64,

    #[ver(V >= V13_0B4)]
    pub(crate) unk_92d_8: Array<0x1b, u8>,
}
```

(Extracted from asahi/fw/fragment.rs)

```rust
#[repr(C)]
pub(crate) struct RunFragmentG13V12_3<'a> {
    [...]
    pub(crate) job_params1: JobParameters1G13V12_3<'a>,
    pub(crate) job_params2: JobParameters2,
    pub(crate) job_params3: JobParameters3G13V12_3,
    [...]
}
#[repr(C)]
pub(crate) struct RunFragmentG14V12_4<'a> {
    [...]
    pub(crate) job_params1: JobParameters1G14V12_4<'a>,
    pub(crate) job_params2: JobParameters2,
    pub(crate) job_params3: JobParameters3G14V12_4,
    [...]
}
#[repr(C)]
pub(crate) struct RunFragmentG13V13_5<'a> {
    pub(crate) counter: U64,
    [...]
    pub(crate) job_params1: JobParameters1G13V13_5<'a>,
    pub(crate) job_params2: JobParameters2,
    pub(crate) job_params3: JobParameters3G13V13_5,
    [...]
}
#[repr(C)]
pub(crate) struct RunFragmentG14V13_5<'a> {
    pub(crate) counter: U64,
    [...]
    pub(crate) job_params1: JobParameters1G14V13_5<'a>,
    pub(crate) job_params2: JobParameters2,
    pub(crate) job_params3: JobParameters3G14V13_5,
    [...]
}
#[repr(C)]
pub(crate) struct RunFragmentG14XV13_5<'a> {
    pub(crate) counter: U64,
    [...]
    pub(crate) registers: job::raw::RegisterArray,
    pub(crate) job_params3: JobParameters3G14XV13_5,
    [...]
}
```

(Macro expansion with rust-analyzer)

Why Rust succeeded in the DRM?

What makes the DRM subsystem special?

Why did it work?

- Maintainers were willing to bet on the success of Rust.
- Group maintainership model.
- Merge sizeable stub drivers and let the driver grow with the community.
  - Helps to decrease the delta between the development branch and the upstream branch.
  - Helps to breed a larger community and interest around it.

Is Rust the new standard in the DRM?

For new drivers, maybe.

We're hiring!

https://www.igalia.com/jobs/

Appendix

- **References used to build the timeline:**
  - 2022:
    - 2022-11-29: Tales of the M1 GPU - Asahi Linux
  - 2023:
    - 2023-03-07: [PATCH RFC 00/18] Rust DRM subsystem abstractions (& preview AGX driver) - Asahi Lina
    - 2023-03-17: [RFC PATCH 0/9] Rust version of the VGEM driver - Maíra Canal
    - 2023-07-14: [PATCH 0/3] DRM scheduler documentation & bug fixes - Asahi Lina
  - 2024:
    - 2024-02-06: Future of nouveau/nova's display driver, and rvkms introduction! - Lyude Paul
    - 2024-03-22: [RFC WIP 0/4] Rust bindings for KMS + RVKMS - Lyude Paul
    - 2024-03-20: Nova and staging Rust abstractions - Danilo Krummrich
    - 2024-05-20: [RFC PATCH 0/8] [RFC] DRM Rust abstractions and Nova - Danilo Krummrich
    - 2024-06-19: [PATCH v2 0/8] DRM Rust abstractions and Nova - Danilo Krummrich
    - 2024-09-25: What the Nova GPU driver needs [LWN.net]
    - 2024-09-30: [WIP RFC v2 00/35] Rust bindings for KMS + RVKMS - Lyude Paul
    - 2024-11-20: RVKMS and Rust KMS bindings [LWN.net]

- **References used to build the timeline:**
  - 2025:
    - 2025-01-31: [PATCH 1/2] gpu: nova-core: add initial driver stub - Danilo Krummrich
    - 2025-02-04: [PATCH v2 1/2] gpu: nova-core: add initial driver stub - Danilo Krummrich
    - 2025-02-09: [PATCH v3 1/2] gpu: nova-core: add initial driver stub - Danilo Krummrich
    - 2025-02-26: [PATCH v4 0/6] Initial Nova Core series - Danilo Krummrich
    - 2025-03-04: [PATCH v5 0/5] Initial Nova Core series - Danilo Krummrich
    - 2025-03-05: [RFC v3 00/33] Rust bindings for KMS + RVKMS - Lyude Paul
    - 2025-03-18: [PATCH 0/7] Rust abstractions for shmem-backed GEM objects - Daniel Almeida
    - 2025-03-09: [GIT PULL] Nova changes for v6.15 - Danilo Krummrich
    - 2025-03-25: [PATCH 0/2] Add a Rust GPUVM abstraction - Daniel Almeida
    - 2025-03-26: [PATCH 0/8] DRM Rust abstractions - Danilo Krummrich
    - 2025-03-26: [PATCH 0/2] Nova DRM skeleton driver - Danilo Krummrich
    - 2025-04-11: [PATCH v2 0/8] DRM Rust abstractions - Danilo Krummrich
    - 2025-04-22: [PATCH v2 0/2] Add a Rust GPUVM abstraction - Daniel Almeida
    - 2025-05-21: [PATCH v2 00/12] Rust abstractions for shmem-backed GEM objects - Lyude Paul
    - 2025-06-21: [PATCH v3] rust: drm: Add GPUVM abstraction - Daniel Almeida

- **References used to build the timeline:**
  - 2025:
    - 2025-06-23: [PATCH] rust: drm: mm: Add DRM MM Range Allocator abstraction - Daniel Almeida
    - 2025-06-27: [PATCH] Introduce Tyr - Daniel Almeida
    - 2025-07-07: Introducing Tyr, a new Rust DRM driver
    - 2025-08-12: [PATCH v2] rust: drm: Introduce the Tyr driver for Arm Mali GPUs - Daniel Almeida
    - 2025-08-29: [PATCH v3 00/14] Rust abstractions for shmem-backed GEM objects - Lyude Paul
    - 2025-09-10: [PATCH v3] rust: drm: Introduce the Tyr driver for Arm Mali GPUs - Daniel Almeida
    - 2025-09-01: [PATCH] MAINTAINERS: Add drm-rust tree for Rust DRM drivers and infrastructure - Danilo Krummrich