Converting a DRM driver to Rust

Maíra Canal

Richmond, VA - LPC 2023

What is the VGEM driver?

- VGEM (Virtual GEM provider) is a minimal non-hardware-backed GEM service.
- It was written in C and introduced in 2015.
- Fairly small driver (~400 lines): GEM service + 2 IOCTLs
- **Use case:** no real GPU available in setups with QEMU and llvmpipe

Why are we rewriting VGEM?

- **Proof of Concept**
- It is a GPU-agnostic driver
- It is a compact driver
- Uses a lot of the DRM framework

What is *rustgem*?

- Rustgem is a driver written in Rust with the exactly same functionality as VGEM
- It was written using Asahi Lina's DRM bindings + RfL bindings
  - Thanks RfL folks!
- I wrote bindings for legacy platform device initialization and *dma-resv*

Technical Hurdles

- Managing unsafe code
  - SAFETY review
  - **Q: How can we encourage SAFETY review inside the subsystems?**
  - **Q: Can a beginner spot subtle safety issues?**

```
+   /// Returns the pointer to reservation object associated with this GEM object.
+   fn resv(&self) -> DmaResv {
+       // SAFETY: Every GEM object holds a reference to a reservation object
+       unsafe { DmaResv::from_raw(self.gem_obj().resv) }
+   }
```


Technical Hurdles

- How to write good safe abstractions?
  - Rust For Linux: Writing Safe Abstractions & Drivers was a good resource for me at that time
  - **Q: Maybe we could include more documentation about writing safe abstractions?**

Technical Hurdles

- Problems with macro expansion
  - **Q: How can we make this easier?**

```
// include/uapi/drm/vgem_drm.h
#define DRM_IOCTL_VGEM_FENCE_ATTACH    DRM_IOWR( DRM_COMMAND_BASE + DRM_VGEM_FENCE_ATTACH, struct drm_vgem_fence_attach)
#define DRM_IOCTL_VGEM_FENCE_SIGNAL    DRM_IOW( DRM_COMMAND_BASE + DRM_VGEM_FENCE_SIGNAL, struct drm_vgem_fence_signal)
```

```
// include/uapi/drm/vgem_drm.h
/* Note: this is an enum so that it can be resolved by Rust bindgen. */
enum {
   DRM_IOCTL_VGEM_FENCE_ATTACH   = DRM_IOWR(DRM_COMMAND_BASE + DRM_VGEM_FENCE_ATTACH, struct drm_vgem_fence_attach),
   DRM_IOCTL_VGEM_FENCE_SIGNAL   = DRM_IOW(DRM_COMMAND_BASE + DRM_VGEM_FENCE_SIGNAL, struct drm_vgem_fence_signal),
};
```


Next step: Upstream

- We have the DRM bindings
- We have the two upstreamable drivers (Asahi and *rustgem*)
- **Q: What could help us to upstream Rust for DRM?**
- **Q: What is the next step?**

Next step: Upstream

- If we accept Rust in the DRM, it means that everyone is responsible for it
- People writing bindings might need to touch C code
- Rust has well-documented benefits that we might want as a community
- If we want to see things moving forward, we need to compromise

Discussion

- Maybe we could include documentation about writing safe abstractions?
- How can we encourage SAFETY review inside the subsystems?
- How can we improve macro expansion?
- **What could help us to upstream Rust for DRM?**
- What is the next step to upstream Rust for DRM?

We're hiring!

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