DRM and Nova GPU Driver (Update)

Kangrejos '25

Danilo Krummrich, Red Hat
Joel Fernandes, Nvidia

Nova - Recap

What is Nova?

- Driver for NVIDIA GPUs based on the “GPU System Processor” (GSP)
    - GSP provides firmware API ➪ serves as HAL
- Successor of Nouveau for all GSP-based GPUs (Turing and later)
- Written in Rust

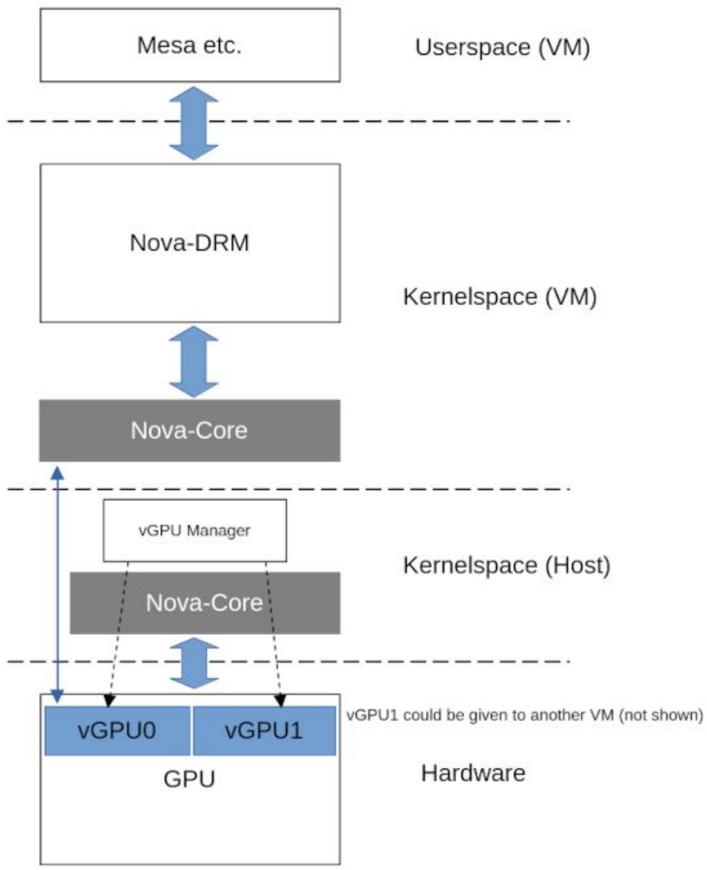
Nova - Recap

What’s the motivation for a rewrite?

- Various (design) problems in Nouveau
- Reduce complexity (GSP only gets us rid of legacy code)
- Make the driver accessible for (new) contributors
- Provide a common firmware and hardware abstraction layer as a separate driver module

Nova - Digression (Driver Stack)

- nova-core
  - Core driver module providing a firmware and hardware abstraction layer
- nova-drm
  - DRM driver connected to nova-core via auxiliary bus
- NVIDIA vGPU
  - VFIO driver managing PCI virtual functions (SR-IOV)



Nova - Recap

What’s the motivation for using Rust?

- Advance Rust in the kernel (and in DRM)
- Take advantage of Rust
  - memory safety features
  - powerful type system
- Maintainable abstraction layer for unstable firmware APIs (type system, procedural macros)

Nova - Recap

What were the challenges when Nova started 1 ½ years ago?

1. Convince people Nova is the way forward
   - NVIDIA posted first vGPU RFC
   - Dave Airlie wrote a nova-core PoC driver

2. Rust Driver infrastructure upstream
   - Fundamental Rust infrastructure in place
     - e.g. abstraction design, locking, reference counts, etc.
   - Rust Driver infrastructure missing (no user)
     - e.g. device/driver model, specific bus support, device resource management, memory-mapped I/O, memory allocation primitives, DMA, etc.

Nova - Upstream Strategy

- ⇨ chicken and egg problem
    - Nova (as a complete driver stack) is too big of a lift
    - Building on non-upstream infrastructure is a waste of time

- Develop Nova (nova-core, nova-drm) in-tree
    - Start with just skeleton drivers upstream

Nova - Upstream Strategy

- ⇨ initial Rust Driver infrastructure and Nova skeleton drivers upstream
  - Generic Device / Driver model
  - PCI, platform, auxiliary bus infrastructure
  - Device resource management
  - Memory-mapped I/O
  - Firmware loader
  - DMA & Scatterlist
  - Memory allocation API
    - Allocators (Kmalloc, Vmalloc, KVmalloc),
    - KBox, VBox, KVBox, KVec, etc.
  - DRM device / driver, GEM, File, IOCTL
  - nova-core, nova-drm
  - ⇨ Maintainer of the mentioned infrastructure (and quite some more)

Nova - Development

Who’s doing all the Nova driver work?

Nova - Development

- NVIDIA engineers contributing to the Nova driver project
    - Contribute the majority of the nova-core code
        - Thanks to Alexandre Courbot, John Hubbard, Joel Fernandes, Alistair Popple, et al.!
    - ➩ Alexandre Courbot Co-Maintainer of nova-core

- My role is leading the Nova project overall
    - Ensure we stay on track regarding original project goals
    - Tackle design topics, e.g. firmware abstraction, VM_BIND and page table management interactions, inter-driver APIs, etc.
    - Work on DRM infrastructure and more Nova (DRM) code

Nova - Development

First Project Milestone:

Run vGPU on top of nova-core.

DRM Rust - Status

- Tyr (Mali GPU) driver (Daniel Almeida, Alice Rhyl)
    - follows Nova's approach of in-tree development
    - Tyr skeleton driver just hit the DRM tree (goes to Linus for v6.18)

- Rust VKMS (Virtual Kernel Mode Setting) by Lyude Paul
    - paving the way for DRM KMS driver infrastructure
    - Nova is not ready for KMS yet

- Apple AGX (Janne Grunau)
    - originally OOT, but tries to land skeleton in-tree as well
    - patches should be posted soon

DRM Rust - Status

- Two DRM drivers in development in-tree
  - rVKMS and Apple AGX upcoming
- Nova had its own tree; Try was targeting drm-misc
- ⇨ drm-misc tree does not scale (initially)
  - DRM core infrastructure + (small) drivers with low patch traffic
- ⇨ drm-rust tree (M: Alice Ryhl, M: Danilo Krummrich)
  - Shared (open comitter) tree for
    - (Rust) DRM core infrastructure
    - in-tree DRM Rust drivers (in development)
    - external dependencies (case by case and when it makes sense)

DRM Rust - Upcoming

- DRM GPUVM Rust infrastructure
  - GPU virtual address space manager
  - Alice Ryhl works on C improvements
    - Streamline different locking schemes

- DRM Jobqueue (Philipp Stanner)
  - Replacement for DRM GPU scheduler
  - GPUs with Firmware scheduler support only
  - First native Rust (DRM) component

Bring-up status

Firmware is now up (booting various microprocessors, bringing up RPC)

- Ampere (RTX 30 series.) – works
- Ada (RTX 40 series) – works
- Blackwell (RTX 50 series) – working but not yet posted
- Turing GPUs (RTX 20 series) – WIP

debugfs

- Matthew Maurer submitted v11 - should be close to merge.
- Support for binary files needed
  - GSP logs into coherent DMA buffers
  - Log format not open source but helps Nvidia engineers
  - Log can be provided to Nvidia as needed for decoding
- Already instrumental in GPU bringup debugging.

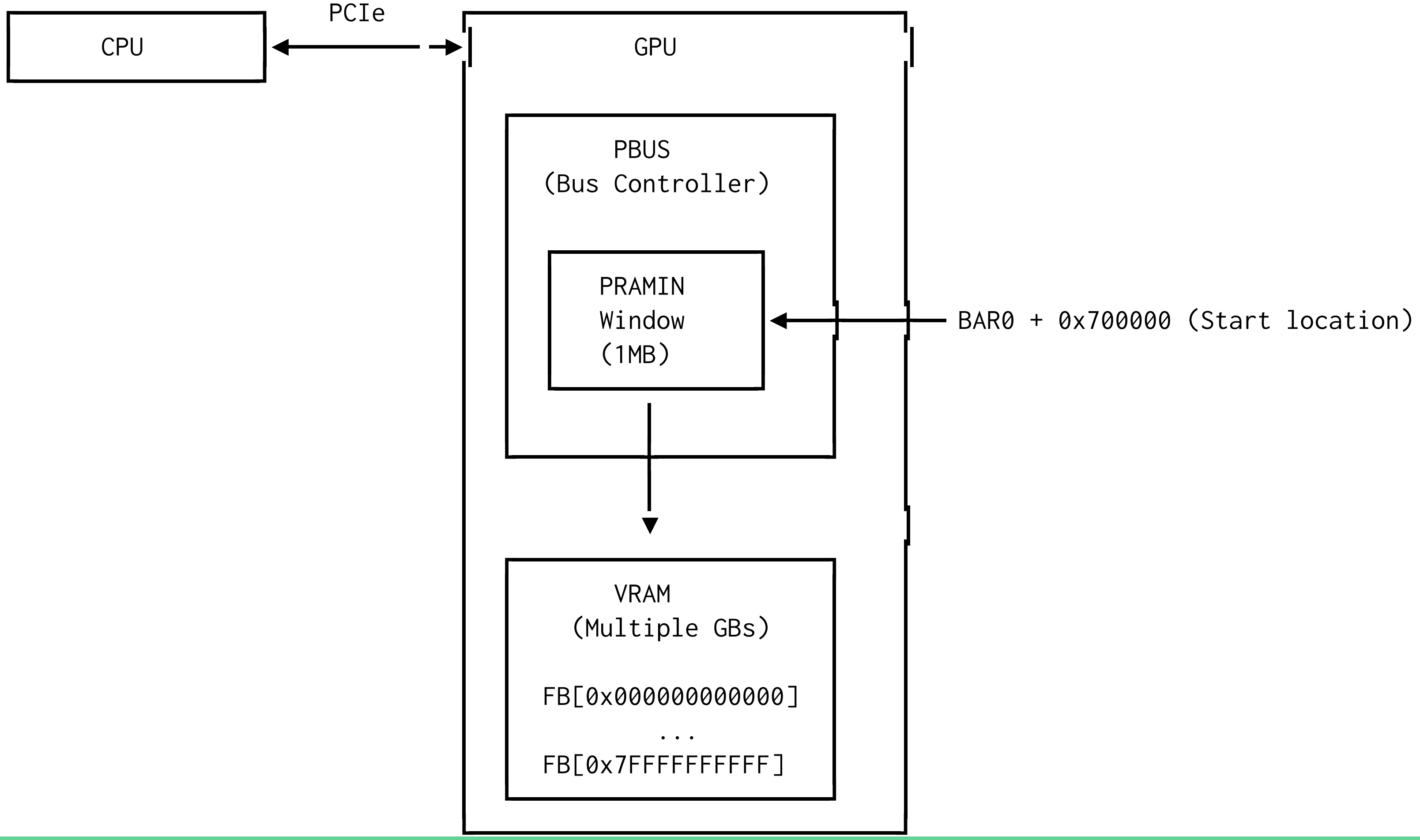
MM updates

Buddy allocator prototype - completed but not yet posted.

- Simple zoned buddy system.
- Tested with Nova-core page table allocations.
- Q: Do we keep this within nova-core, or move it to rust/kernel/mm/ ?
  - Can do it if other drivers use it, OR nova-core is Ok to mature it more.
  - Has some Nvidia-isms like zones specific to us. But lot of code is generic.
- Similar to drm_buddy, but simpler (no binary tree)
- Does not require interfacing with C code/bindings.

MM updates

○ PRAMIN aperture helps write directly to VRAM (bootstrap).



MM updates: Bitfield support for Rust structs (v3)

- Required for bitfield-packed structures, such as page table entries.
- An example:

```
bitfield! {
    pub struct ControlReg: u32 {
        3:0       mode        as u8;
        7         state       as bool;
    }
}
// let reg = ControlReg::default().set_mode(3);
```

MM updates: Bitfield open question 1 - hidden bits

- Required for bitfield-packed structures, such as page table entries.
- An example:

```
bitfield! {
    pub struct ControlReg: u32 {
        3:0       mode        as u8;
    }
}
// let reg = ControlReg::from(0xFF);
// What happens to bits 4-7
```

MM updates: Bitfield open question 2 - size of field exceeds

- Required for bitfield-packed structures, such as page table entries.
- An example:

```
bitfield! {
    pub struct ControlReg: u8 {
        3:0       mode        as u32; // needs to be ‘as u8 => u32’
    }
}
// Won’t compile: ControlReg::default().set_mode(10).mode()
```

MM updates: Page table walk: BAR1 mapping support

- Required for Virtual Address space access to VRAM
- 256MB on most GPUs.
- This is how BOs are accessed directly from CPU without DMA.
    - Prototype of GPU VA to PA translation completed
    - Next: upstreaming of page table/directory entry structures
    - Next: upstreaming of low-level page table walker (Nvidia specific).
    - Q: What’s the next logical step:
        - Do we integrate into GPUVM? What’s the next logical step.
        - For VA allocation/free, do we reuse maple tree?

MM updates: Upstream Stages

1. Bitfield
2. Buddy allocator
3. PRAMIN support
4. Page table structures
5. Page walker (depends on all 4 above)
6. Bar1 mappings (depends on all 5 above)
7. VA allocation/free (for kernel carve outs of VA space using Maple Tree)
8. Exposing low-level page table operations to users (GPUVM)
9. TTM integration

IRQ updates

- Daniel Almeida’s patchset for request_irq is now in -next (yayy!).
- Next: MSI/MSI-X IRQ vector allocation
    - Joel posted patch (v1)
    - Comments posted by Danilo:
        - Devres integration - WIP
        - Next: need better representation of IRQ numbers, vector idx
            - TODO: Update this bullet with more info from latest discussion.

IRQ updates

- VFN (virtual function notifier) is the latest incarnation of IRQ controller in Nvidia hardware.
- VFN (Virtual Function Notifier) - prototyped
  - Next: need to post it upstream.
  - Next: GSP RPC as the first user of IRQ support (currently polling)

- Joel has working prototype of GSP interrupt handling in nova-core using VFN, MSI vector, request_irq patches, etc.

Documentation efforts

- Nova-core emphasizes high documentation quality
- Clearly defined registers
- Code comments
- No magic numbers
- Mostly readable code over past open source drivers.

Nova: Calling kernel modules from C to Rust

- Nova has to be supported as a kernel module and other kernel modules need to be able to call into it.
- This is central to Nvidia driver deployments, an installer uses DKMS to load modules.
- Simple experiments show function calling from C to Rust works well even across loadable kernel modules.
- Any other pitfalls?