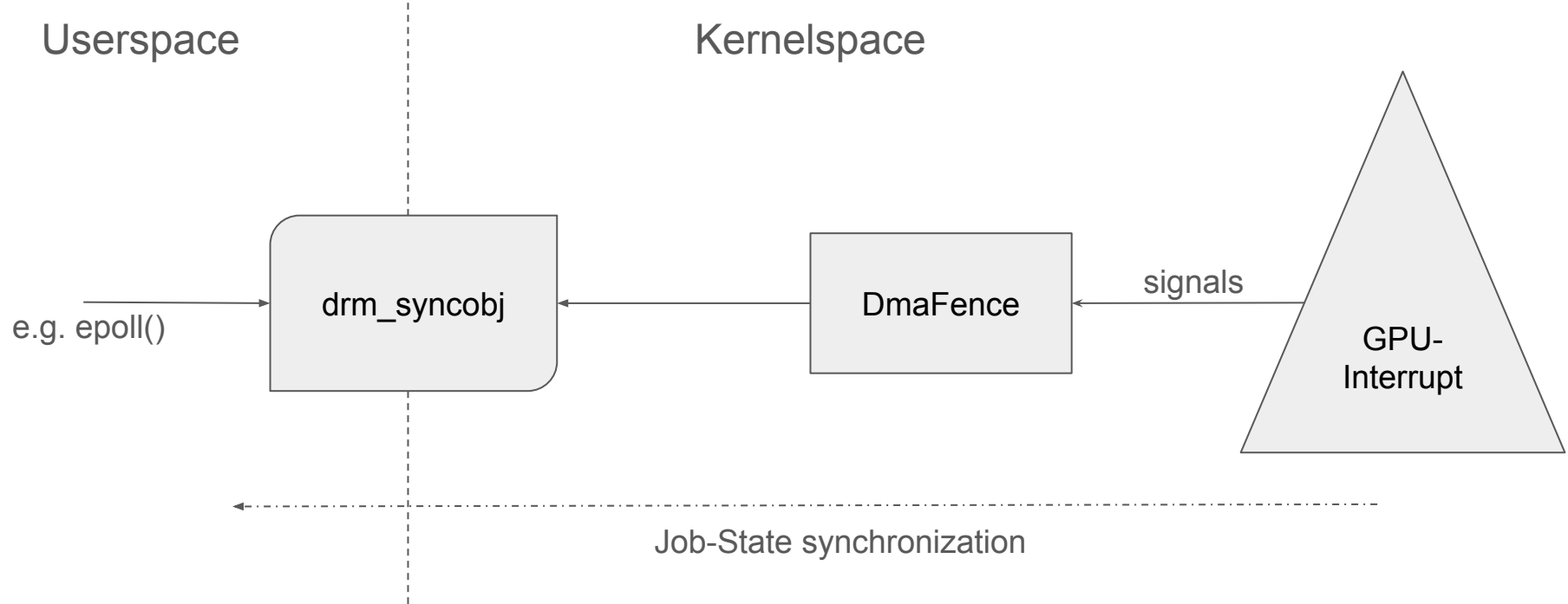


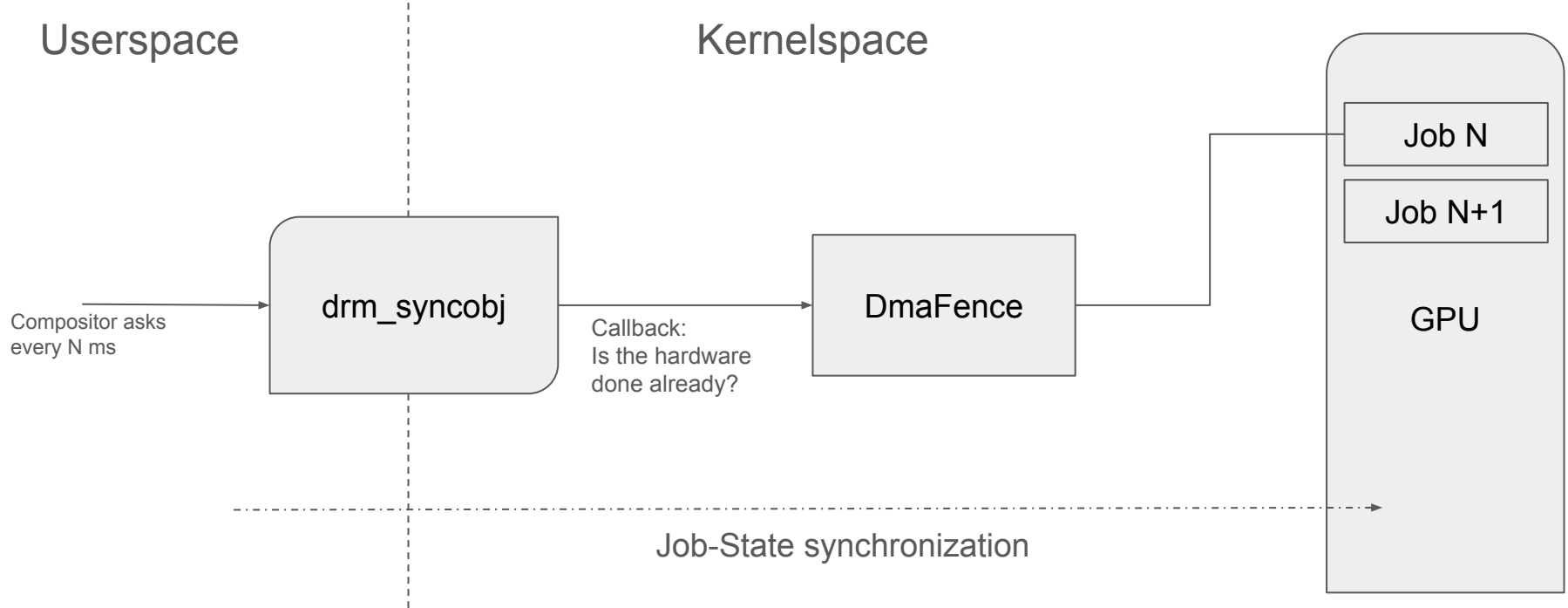
DmaFence vs core::mem::forget()

Philipp Stanner (Red Hat)

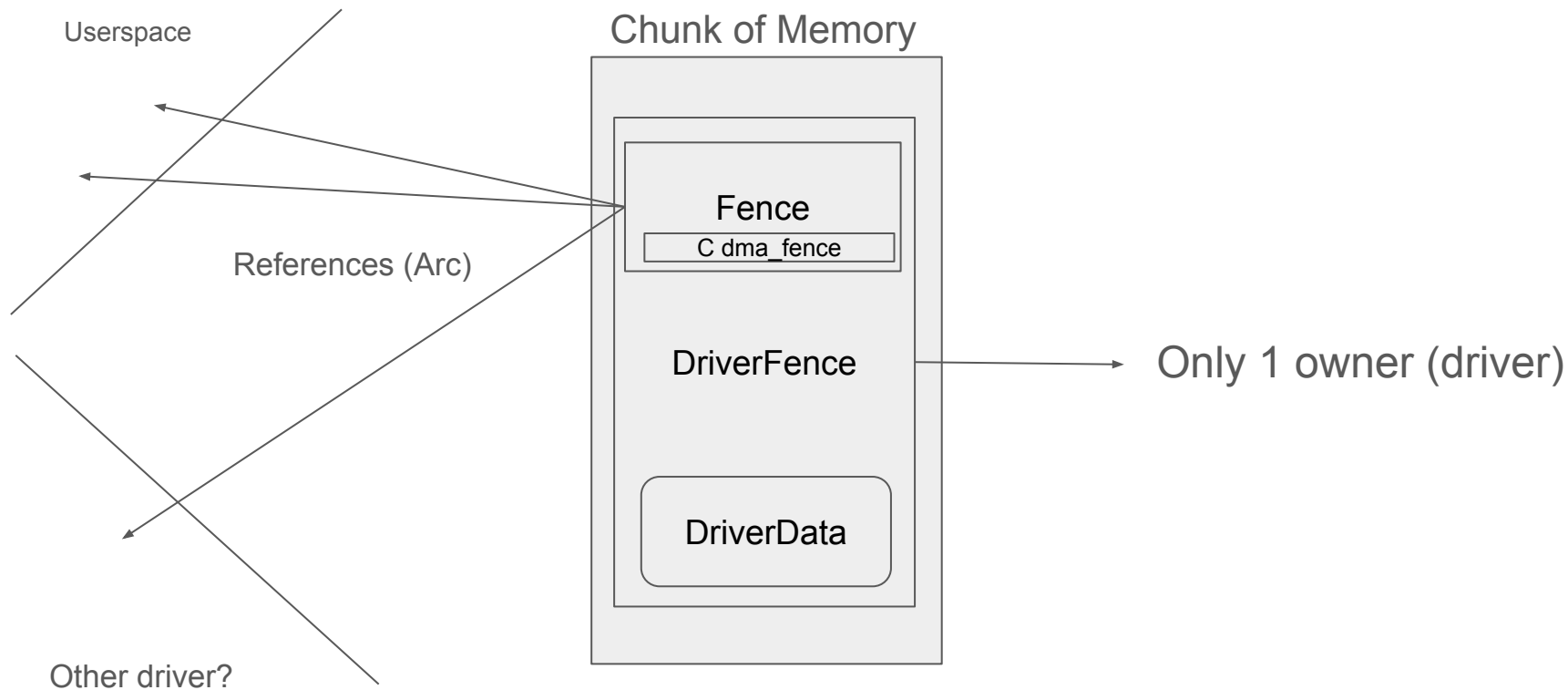
Fence Operation: Interrupt-driven Signaling



Fence Operation: Fence Callbacks



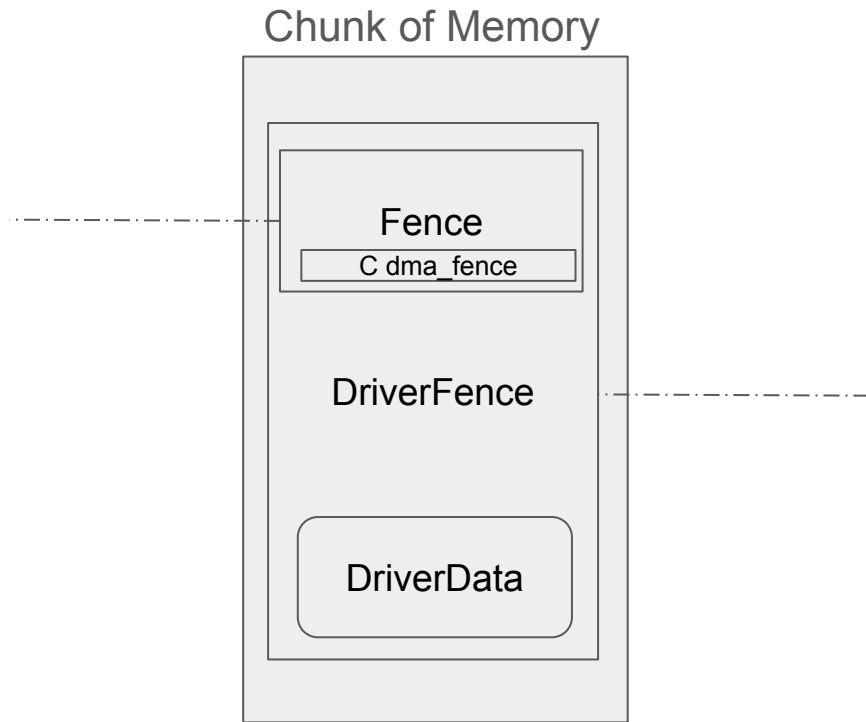
Our Design in Rust



Our Design in Rust

Fence:

- Arbitrary number of Arc's
- You can register callbacks on it (i.e., “wake up userspace”)
- You can check the fence status

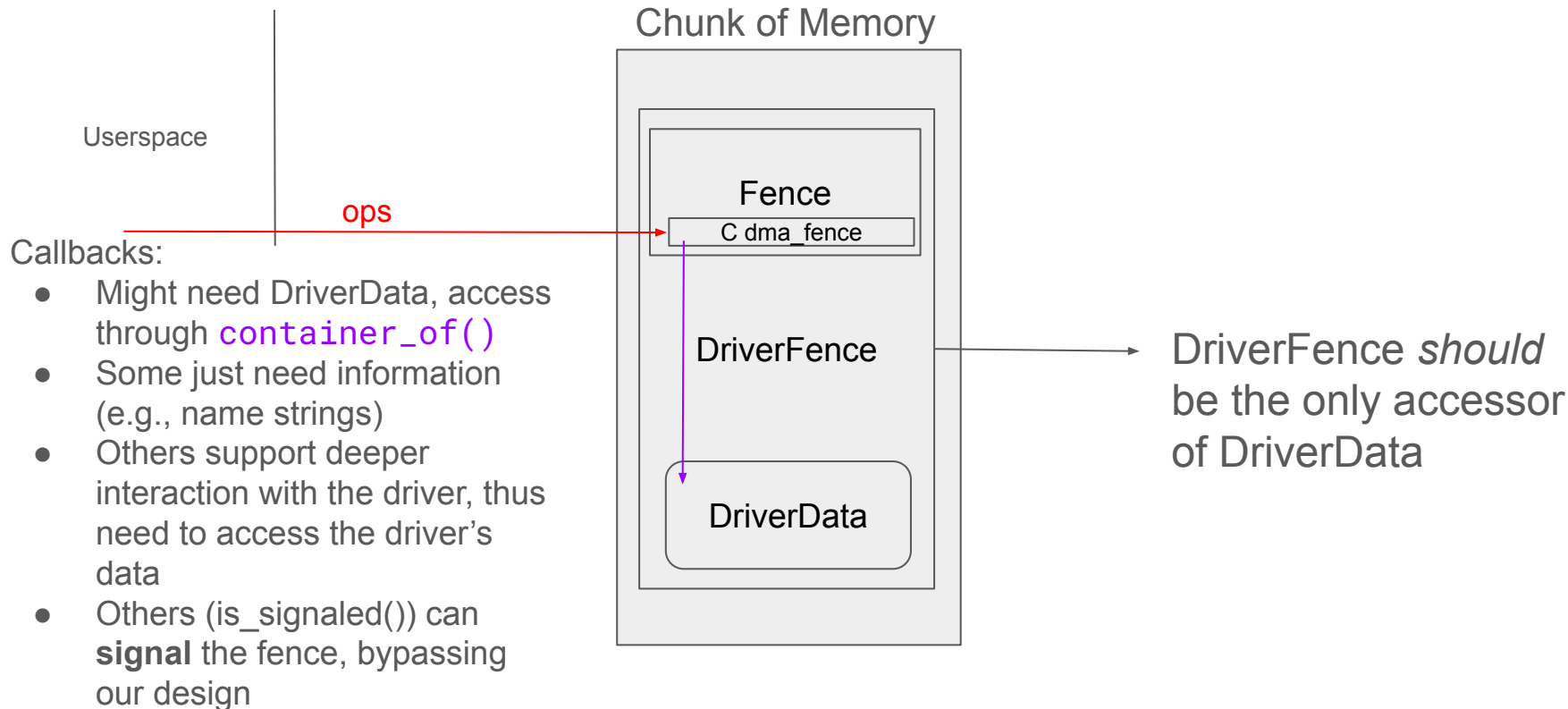


DriverFence:

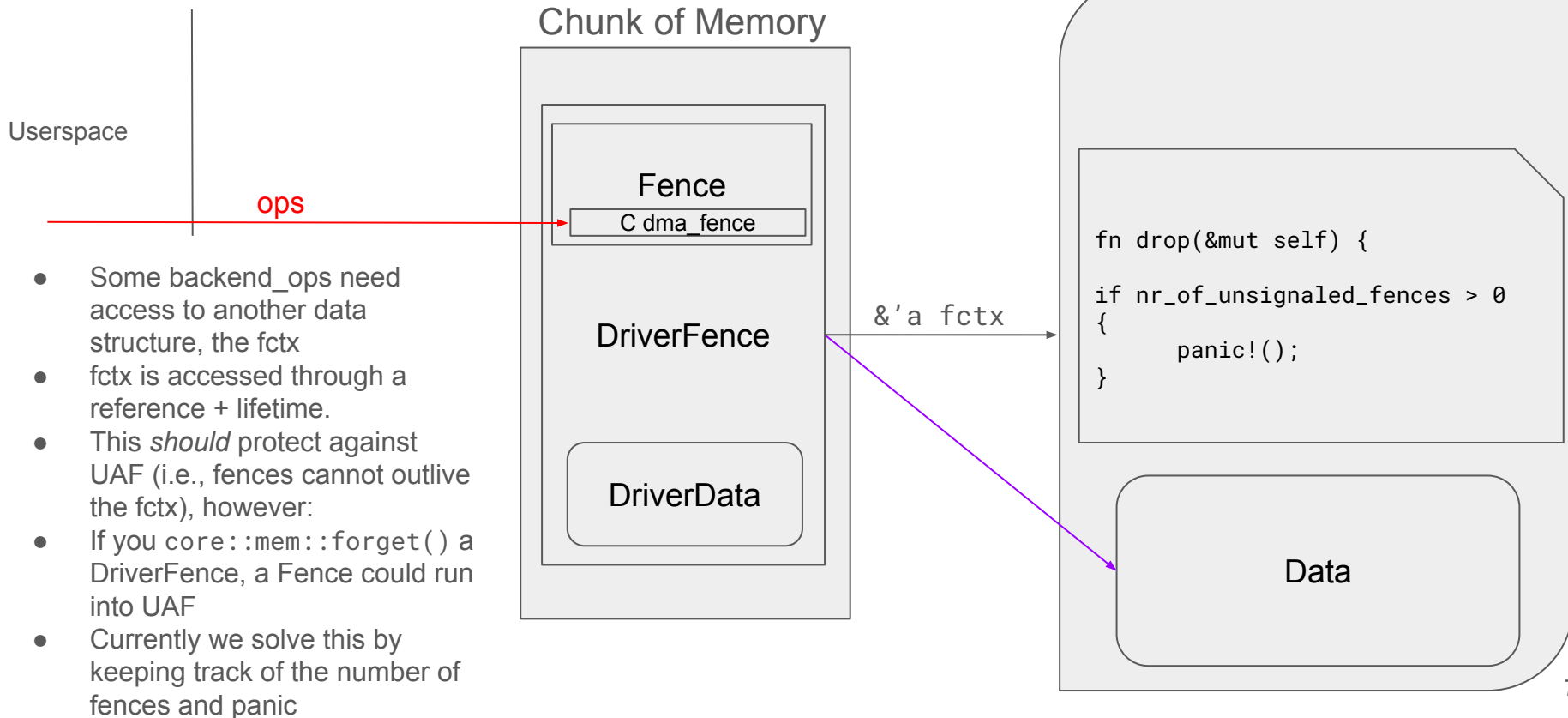
- Only one owner
- Only object you can signal fences with
- Signals itself with an error / warning on drop
- *User-specified DriverData drops in atomic context*

All refcount C's dma_fence together. Last kref_put() frees the memory

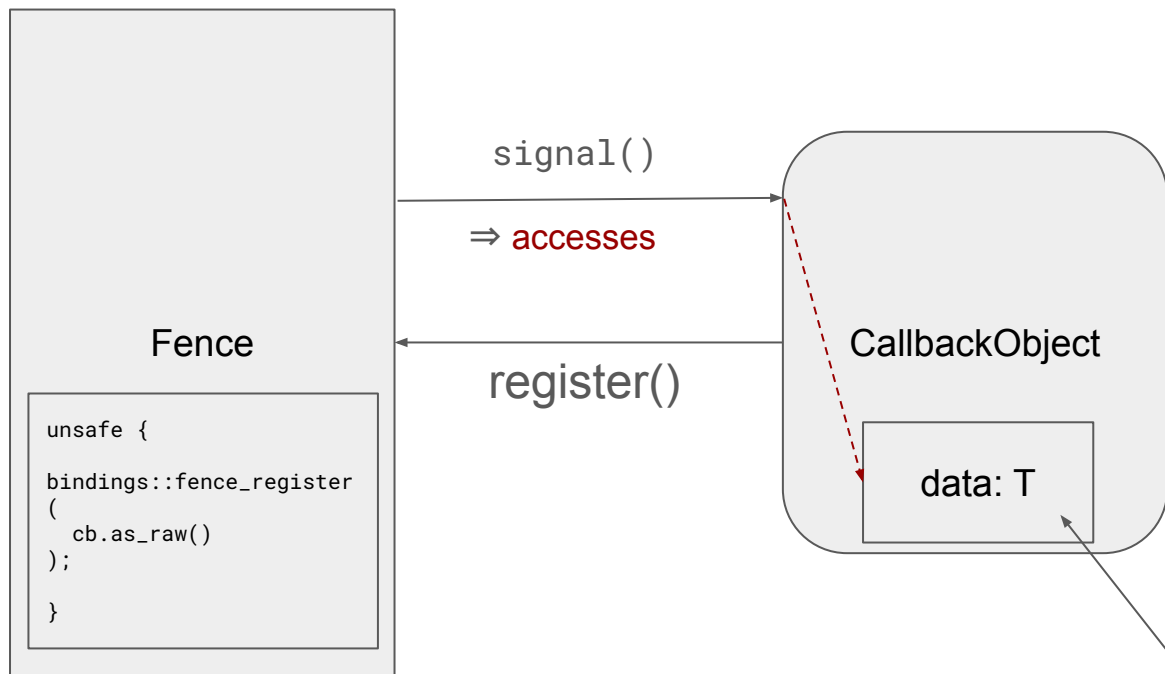
backend_ops – A Future Achilles Heel?



core::mem::forget() - Problem 1



core::mem::forget() - Problem 2



1.
`core::mem::forget(`
 `CallbackObject`
`);`

2. Fence signals

Result:
Access on data,
which might belong
to unloaded driver

T: 'static ??
data: Arc<T> ??

How to defeat `core::mem::forget()` ?

Conceived solutions:

1. 'static lifetime $\Rightarrow$ bubbles up through all users (uncool)
2. `Arc<T>` $\Rightarrow$ T can outlive its driver
3. Keep track in your parent object and panic if stuff is forgotten $\Rightarrow$ You shouldn't panic the kernel (and probably only works for problem №1 anyways)

What should we do?

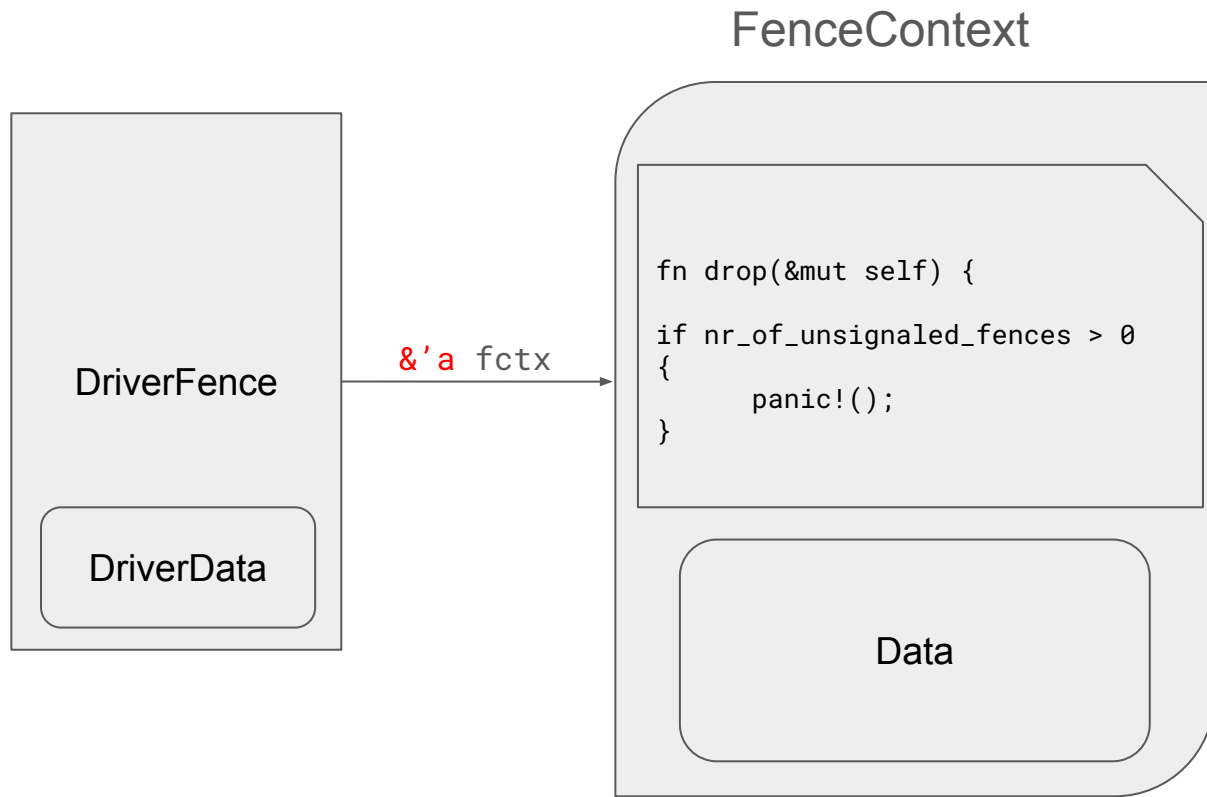
Acknowledgements

A large number of people co-designed DmaFence together:

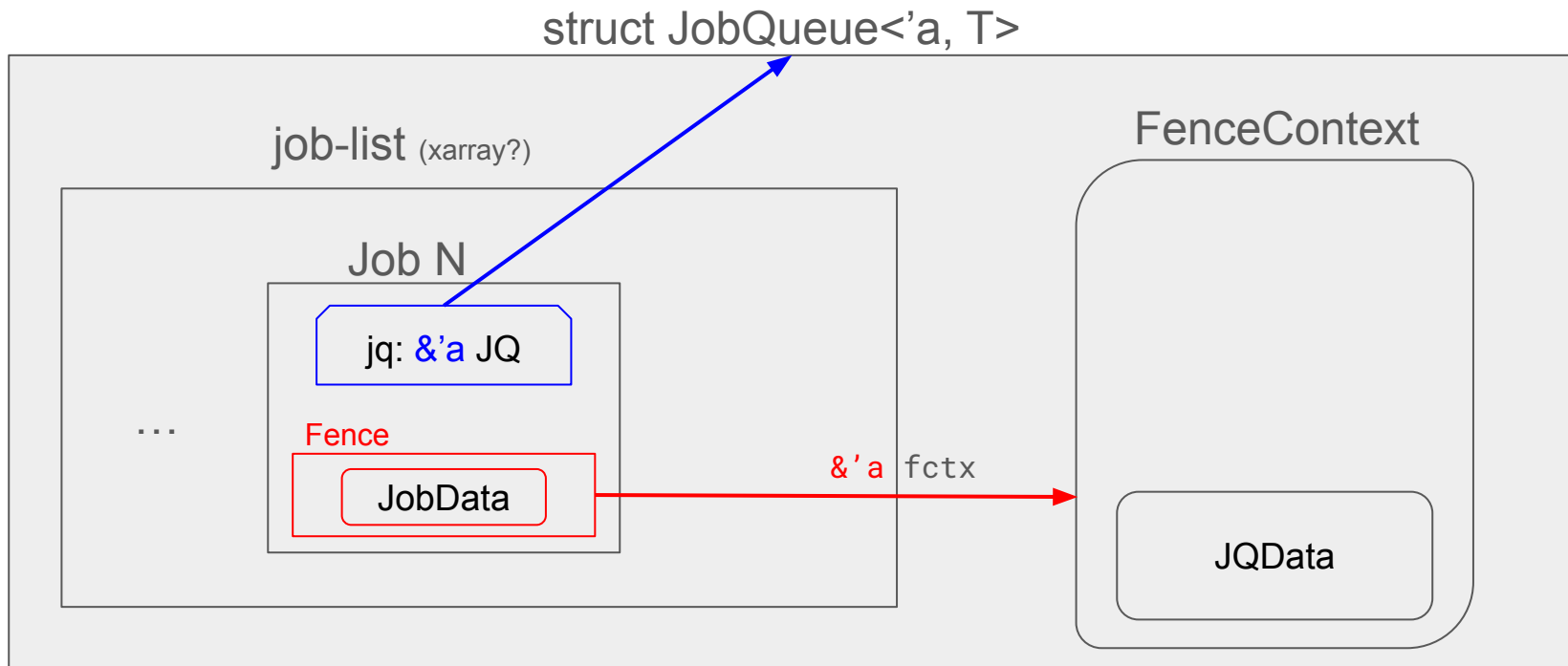
- Alice Ryhl (Google)
- Boris Brezillon (Collabora)
- Daniel Almeida (Collabora)
- Danilo Krummrich (Red Hat)
- Gary Guo (Red Hat)

Appendix - JobQueue Problems

Remember the Lifetime from DmaFence?



Plumbing: Self-Referential Lifetime



Jobs contain **ref to JQ**. Jobs' fences contain **ref** to FenceContext
⇒ Currently does not compile (with `pin_init`). Gary Guo working on