

The Ownable series

Use case

- We have kernel objects that Rust should own **uniquely**,
- but which C allocates and frees, so `KBox<T>` does not fit,
- and which are not reference counted, or must not be shared at times, so `ARef<T>` does not fit.

`Owned<T>` is a unique handle that calls `T::release()` on drop: a `KBox<T>` where `T` decides how it is freed.

- `T::release()` end the ownership, but does not have to free the object.
- Consider object caches: `T::release()` recycles rather than frees.

Background: `ARef<T>`

- `ARef<T>` is an owned reference count on an object with an **abstracted** reference count.
- Like `Arc<T>`, but implementer decide where the count lives
 - usually inside the object (intrusive),
 - and alloc/free is out of scope (behind FFI boundary).
- `T: AlwaysRefCounted` supplies `inc_ref(&self)` and `unsafe dec_ref(NonNull<Self>)`.
- `Clone`  `inc_ref()`, `Drop`  `dec_ref()`.
- `&T`  `ARef<T>` upgrade: a `&T` proves the object is alive, so just bump the count.
- Shared access only, never `&mut T`.

Transition `Owned` ↔ `ARef`

- `OwnableRefCounted` converts `Owned` ↔ `ARef` for types that can be both, like `UniqueArc` ↔ `Arc`.
- Not possible for `KBox` and `Arc` due to memory layout difference, hence `UniqueArc`.
- Memory layout not a problem for `Owned` ↔ `ARef` (memory layout out of scope).

Transition **Owned** **ARef**

- `into_shared(this: Owned<T>) -> ARef<T>` : always succeeds, the unique handle becomes the first shared one.
- `try_from_shared(this: ARef<T>) -> Result<Owned<T>, ARef<T>>` : may fail and hand back the `ARef` .
- Sugar: `From<Owned<T>> for ARef<T>` and `TryFrom<ARef<T>> for Owned<T>` .
- Upgrading/downgrading is implementation detail:
 - `block:mq::Request` uses unconventional reference counts to implement state tracking.

Shared reference upgrade

- `Owned` is unique.
- `Owned` implements `Deref`, some times also `DerefMut` (for `T: Unpin`).
- Allowing `&T`  `ARef<T>` upgrade would be unsound under existence of `Owned<T>`.
- Some users of `Into<ARef<T>>` for `&T` feel very strongly about keeping it 🧑
- But these users could change API to take `ARef<T>` instead of `&T`?
- Compromise: `AlwaysRefCounted` is split into `RefCounted` and `AlwaysRefCounted`.
 - `RefCounted` provides `ARef<T>`
 - `AlwaysRefCounted`: `RefCounted` provides `&T`  `ARef<T>` upgrade
 - `unsafe` trait, do not implement with `OwnableRefCounted`.

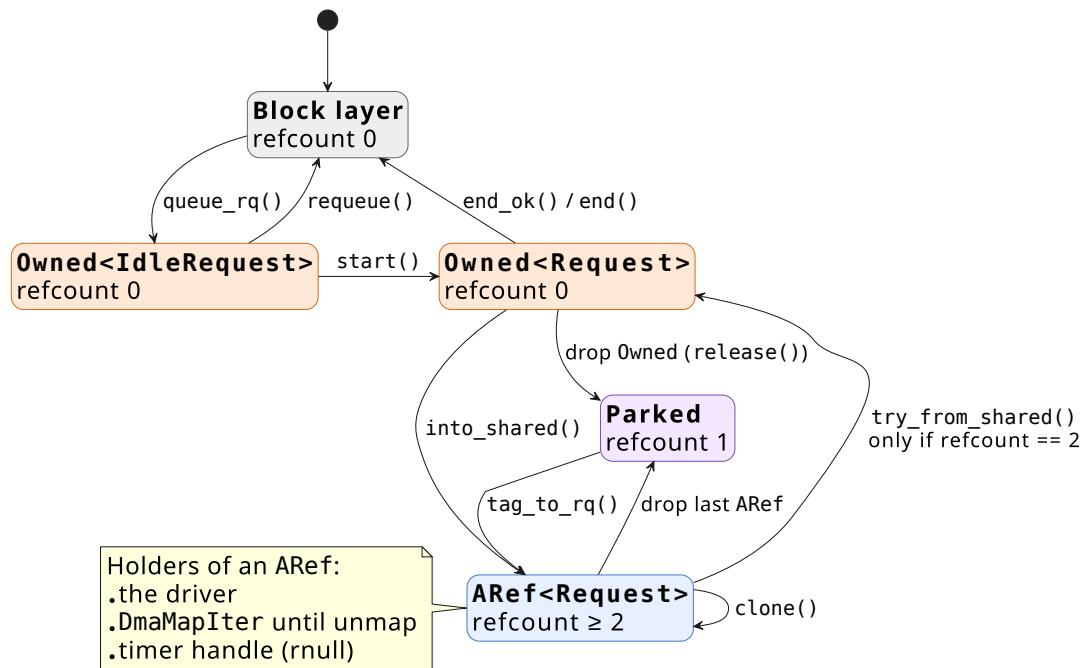
Block layer: **struct request**

- Request descriptors are allocated and initialized **during device setup**.
- Number of request structures == max outstanding IO requests.
- To issue IO, C block layer allocates a **free request** structure and passes it to driver.
- Driver issues and completes commands with device, handing back request when done.
- Request states:
 - Owned by block layer (no touch from driver).
 - Uniquely owned by driver.
 - Shared owned by driver.
 - Parked owned by driver.

Block layer: `struct request`

- `queue_rq(rq: Owned<IdleRequest<T>>)` : the driver owns the request exclusively.
- Ending request requires ownership: `Owned<Request>::end_ok(self)` consumes the reference.
- Drivers need to do DMA mapping of request data ➡ Must keep request alive for the duration of the mapping.
 - Transition to `ARef<Request>` for the duration of the mapping.
 - Fallible transition back when mapping torn down.
- `Ownable::release()` need not free: `Request::release()` hands the request back to the block layer for reuse (**object cache**).

struct request state transitions



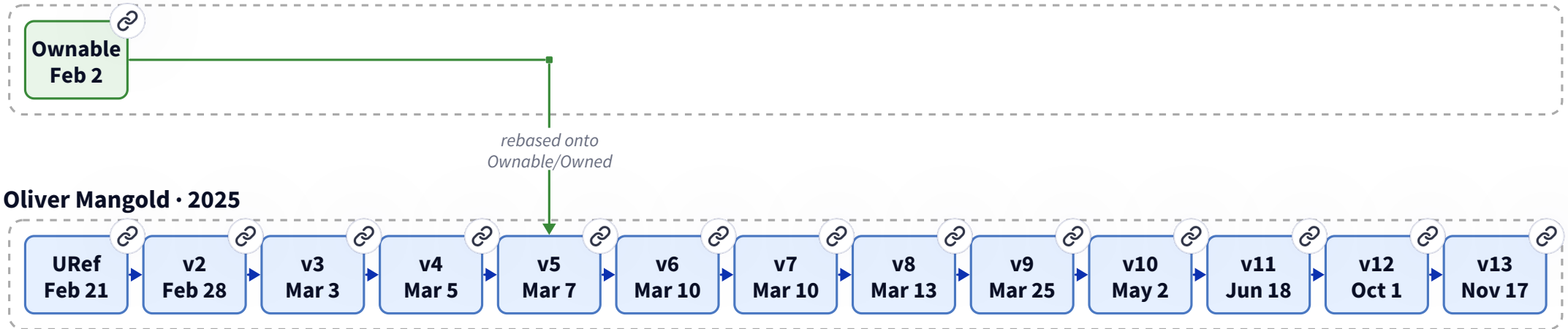
- refcount 0: block layer owns it, or the driver holds it uniquely.
- refcount 1: parked. Rust owns it, nobody references it, `tag_to_rq()` can fetch it.
- refcount ≥ 2 : shared. Number of `ARef` s + 1.
- `rnvme`, `rufs` : `DmaMapIter` holds an `ARef` for the lifetime of the DMA mapping. `try_from_shared()` succeeds only once the mapping is dropped.

Other potential users

- [Asahi Lina](#): IOMMU page tables for the Apple GPU driver, the original motivation.
- [Alice Ryhl](#): Rust Binder pages. Ended up as `ARef<Page>` in v21, since `vm_insert_page()` shares the page.
- [Alice Ryhl](#): DRM GPUVM's `UniqueRefGpuVm`
 - if one `owned` may coexist with `ARef S (ListArc style)` -> out of scope, `owned` is unique.
- Others?

History

Asahi Lina · 2025



Andreas Hindborg · 2026



May 26: "v17 is in preparation for Kangrejos, and we believe the series is approaching a final shape."

Key design changes

When	Change
v1 Lina · Feb 2 2025	<code>Ownable</code> / <code>Owned</code> for <code>Page</code> in IOMMU page tables. <code>OwnableMut</code> marker gates <code>DerefMut</code> .
v1 Mangold · Feb 21 2025	<code>URef</code> / <code>UniqueRefCounted</code> : a unique <code>ARef</code> , as <code>UniqueArc</code> is to <code>Arc</code> . For block-mq <code>Request::end_ok()</code> .
v3 · Mar 3 2025	Unsound <code>&T -> UniqueRef<T></code> conversion removed (Benoît).
v5 · Mar 7 2025	Series merge: <code>UniqueRef</code> dropped for <code>Owned</code> ; <code>OwnableRefCounted</code> converts <code>ARef</code> <code><-></code> <code>Owned</code> . <code>AlwaysRefCounted</code> split into <code>RefCounted</code> + marker trait, as <code>&T -> ARef<T></code> is unsound for <code>ownable</code> types (Alice).
v11 · Jun 18 2025	<code>SimpleOwnableRefCounted</code> blanket impl dropped (Alice).
v12 · Oct 1 2025	<code>OwnableMut</code> dropped; <code>DerefMut</code> requires <code>Unpin</code> , else <code>as_pin_mut()</code> . Code moves to <code>kernel::owned</code> .
v14 · Feb 4 2026	Andreas takes over. <code>Page</code> converted to <code>Ownable</code> ; <code>ForeignOwnable</code> for <code>Owned</code> .
v21 · Sep 10 2026	<code>Page</code> back to <code>AlwaysRefCounted</code> (<code>ARef<Page></code>). <code>ExclusivePage</code> is the <code>Owned</code> user, without <code>Deref<Target = Page></code> to keep aliasing sound (Alice).

Owned<Page> + ARef<Page>

- Page would be `Ownable`, `!AlwaysRefCounted`.
- Binder currently pass `&Page` to `vm_insert_page()`.
- `vm_insert_page()` inserts page to user space address space, taking reference count.
 -  illegal under existence of `Owned<Page>`.

Owned<Page> + ARef<Page>

- With v21 `Page: AlwaysRefCounted` patch, binder keeps `ARef<Page>` instead of `Page`.
- What if we make `Page: RefCounted + Ownable`?
 - Change `vm_insert_page()` to take `ARef<Page>` instead of `&Page`?
 - Make `owned<Page>` provide mutable access to page contents, eliminating `ExclusivePage`.

By type invariant `Owned<Page>` cannot exist in any DMA mapping or VMA ➡ data access is sound.

Final remarks

- As of v21 `RefCounted` provides two way transition `Ownable` ↔ `ARef` .
 - Should we split this trait to seletively enable transition?
- Any other objections or **questions** we should discuss?
- Please review 🙏