Contribution ID: **404** Type: **not specified**

type-based slab allocation: kmalloc_obj family

Thursday 11 December 2025 17:00 (15 minutes)

Right now the generic interface to the slab allocator is strictly size based, but most of the allocations done via slab are actually instantiating specific objects, and their type information is much more useful to expose to the allocator than their size. (Though size is still important, give dynamically sized objects via flexible arrays.)

Type information is needed to make better choices about:
- Alignment. Right now slab falls back to worst-case alignment for compatibility reasons; see commit ad59baa31695 ("slab, rust: extend kmalloc() alignment guarantees to remove Rust padding")
- Partitioning (e.g. to resist use-after-free style type confusion attacks). Other partitioning proposals are either totally type agnostic (CONFIG_RANDOM_KMALLOC_CACHES), per call-site (https://lore.kernel.org/lkml/20240809072532.work.266-kees@kernel.org/), or associate types with allocations only on a best-effort basis (https://lore.kernel.org/lkml/20250825154505.1558444-1-elver@google.com/)

Additionally, type-based allocation means that return values are explicitly typed instead of returning void *, which can catch more mistakes at compile time.

Latest version of the proposed kmalloc_obj() interface was v4:
https://lore.kernel.org/lkml/20250315025852.it.568-kees@kernel.org/

Can the we converge on the best API, and adapt it to cover things like devm_kmalloc as well?

Primary author: COOK, Kees (Google)

Presenter: COOK, Kees (Google)

Session Classification: Kernel Memory Management MC

Track Classification: Kernel Memory Management MC