Contribution ID: **332** Type: **not specified**

External locking for internally synchronized data structures

Some C kernel data structures exposed to Rust code apply internal synchronization (XArray). Depending on the type of lock, such data structures need to unlock locks when allocating memory. Sometimes it is beneficial to use a single external lock to protect multiple such data structures.

In Rust this creates a problem that is not present in C. This is because that mutably borrowing through a lock guard takes out a mutable borrow on the lock guard, thus making the guard unavailable, and thus the lock cannot be released momentarily while allocating memory.

In Kangrejos (the Rust for Linux annual workshop), we presented a Rust API that allows the use of an external lock while still allowing the data structure to momentarily drop locks for the purpose of allocating memory.

In this session we address concerns that were raised at Kangrejos; how we handle races that can occur while the lock is dropped for memory allocation purposes, and the effect of the locking pattern on data structures applying the Entry pattern. We also aim to present benchmark results collected from real kernel code, rather than user space toy examples.

We aim to use the session to collect input from the community to iron out any potential pain points of the external locking API. We hope that the session will spark discussion and awareness in the community, such that the basic shape of the API is widely accepted when patches hit the list.

Primary author: Mr HINDBORG, Andreas (Samsung)

Presenter: Mr HINDBORG, Andreas (Samsung)

Session Classification: Rust MC

Track Classification: Rust MC