Unsafe Cell async

help

ewan15 November 30, 2022, 10:20pm 1

When using async libraries if you spawn a new task you will often have to deal with concurrency issues surrounding this (wrapping in Arc<Mutex<>>). If I was to only have a single threaded application (e.g tokio max_threads=1) could I wrap my variables shared across threads in UnsafeCell instead? If so why do I still have to wrap in UnsafeCell?

Cerber-Ursi December 1, 2022, 2:11am 2

ewan15:

> If I was to only have a single threaded application (e.g tokio max_threads=1) could I wrap my variables shared across threads in UnsafeCell instead?

You don't have to reach for `unsafe` - it's entirely possible to use `Rc<RefCell<_>>`, if you use single-threaded type of scheduler (e.g. `tokio` with `feature = "rt"` and not `feature = "rt-multi-thread"`), which doesn't require spawned futures to be `Send`.

ewan15:

> why do I still have to wrap in UnsafeCell?

`*Cell` is necessary anywhere you have some shared mutable state. This is not strictly question of parallel execution - the iconic example of single-threaded shared mutability problem is iterator invalidation. See here for a good explanation.

afetisov December 1, 2022, 3:04am 3

`UnsafeCell` on its own is not a synchronization mechanism, and doesn't guarantee correctness even for single-threaded applications. It's a very complicated low-level building block, which is used only in unsafe code implementing more useful higher-level abstractions, like `Mutex` itself.

I'm not sure what exactly is your issue with `Arc<Mutex<T>>`. Is it a worry about performance? An uncontended single-threaded mutex is basically free. Your tokio-based web server will definitely spend orders of magnitude more time on I/O. Is it the verbosity of declaration? I suggest to get used to it, that's standard Rust. Of course, you can always declare a type alias if you use a certain type a lot. Is it the locking API? That's basically a requirement even in single-threaded code for shared data, `Rc<RefCell<T>>` uses basically the same mechanism.

Note that tokio *requires* your shared data to be `Send`, even in single-threaded mode. In short, that's part of its design, and having different requirements for single-threaded code would basically require a separate library anyway. Even if you run all your async tasks on a single executor thread, you still need extra blocking threads to perform non-blocking IO. The basic OS IO interfaces used by tokio are blocking. This means that whenever you want to do a blocking operation, the relevant data and the request must be offloaded to a separate blocking thread. Since your shared data may be involved in those requests, it must also always be `Send`.

It's possible to have a truly single-threaded runtime with no `Send` requirements (see e.g. glommio), but that's just not what tokio is, and it comes with its own limitations.

Cerber-Ursi December 1, 2022, 5:34am 4

afetisov:

> Note that tokio *requires* your shared data to be `Send`, even in single-threaded mode.

In single-threaded mode it should be possible to use `LocalSet`, if I understand correctly?

afetisov December 1, 2022, 3:30pm 5

LocalSet must be used within the `Runtime::block_on` call. You can't call it outside of a usual runtime, or inside `tokio::spawn`. This makes it quite inconvenient. You could use it in principle, but I don't think readymade web frameworks do it. Also, if you use `tokio::spawn`, then the future will need to be `Send` anyway, and if you use `block_in_place`, then you will, well, block the current thread running the LocalSet.

1 Like

system Closed March 1, 2023, 3:31pm 6

This topic was automatically closed 90 days after the last reply. We invite you to open a new topic if you have further questions or comments.

Related topics

Topic		Replies	Views	Activity
Sending Rc objects to another thread help	11	384	September 23, 2025	

Why must local variables in async functions satisfy `Send`? help	12	1887	July 13, 2021
Alternative to Mutexs, When code is sound without them	4	875	February 20, 2023
Shared resources: generic bound for `Rc<RefCell<T>>` and `Arc<Mutex<T>>`,	3	332	February 16, 2024
Questions about tokio/futures and RwLock help	19	4741	January 12, 2023