When or why should I use a Mutex over an RwLock?

Asked 4 years, 5 months ago Modified 2 months ago Viewed 19k times

69

When I read the documentations of `Mutex` and `RwLock`, the difference I see is the following:

- `Mutex` can have only one reader or writer at a time,
- `RwLock` can have one writer or multiple readers at a time.

When you put it that way, `RwLock` seems always better (less limited) than `Mutex`, why would I use it, then?

multithreading rust mutex readwritelock

Share Edit Follow

edited Sep 17 at 14:50

Nikita Fedyashev
17.5k 11 45 80

asked Jun 5, 2018 at 15:52

Boiethios
34.3k 14 123 173

2 Answers

Sorted by: Highest score (default)

58

Sometimes it is better to use a `Mutex` over an `RwLock` in Rust:

`RwLock<T>` needs more bounds for `T` to be thread-safe:

- `Mutex` requires `T: Send` to be `Sync`,
- `RwLock` requires `T` to be `Send` **and** `Sync` to be itself `Sync`.

In other words, `Mutex` is the only wrapper that can make a `T` syncable. I found a good and intuitive explanation in reddit:

> Because of those bounds, RwLock requires its contents to be Sync, i.e. it's safe for two threads to have a &ptr to that type at the same time. Mutex only requires the data to be Send, because conceptually you can think of it like when you lock the Mutex it sends the data to your thread, and when you unlock it the data gets sent to another thread.

Use `Mutex` when your `T` is only `Send` and not `Sync`.

Preventing writer starvation

`RwLock` does not have a specified implementation because it uses the implementation of the system. Some read-write locks can be subject to writer starvation while `Mutex` cannot have this kind of issue.

`Mutex` should be used when you have possibly too many readers to let the writers have the lock.

Share Edit Follow

edited Jun 6, 2018 at 7:45

answered Jun 5, 2018 at 15:52
Boiethios

34.3k 14 123 173

4 Only a naive implementation of a reader/writer lock would allow writers to starve. If I was responsible for an application that would benefit from a reader/writer lock, and if I had to deploy it on some platform where the "standard" reader/writer lock was crap, then I would write my own---tailored to the needs of my application---rather than fall back to a simple mutex. – Solomon Slow Jun 5, 2018 at 20:14

5 I would touch on *performance* and *intent* here. That is (1) RW Lock implementations are generally more complex, and thus slower and (2) if your algorithm only needs a Mutex, using a RW Lock is just confusing your readers for no benefits. Could even add correctness: if you paper around locking issues by just upgrading Mutex to RW Lock at random, you're toast. Multi-threading and Distributed programming require clear models of the data-flow; just throwing locks around randomly does not correct the flows magically. – Matthieu M. Jun 6, 2018 at 6:23

10 Although the question is under 'rust' tag, I found the documentation on ReadWriteLock interface in Java to have quite a comprehensive discussion on various things the *user* must consider when choosing between a mutex and a read-write lock, and the possible implementation trade-offs. – Dmitry Timofeev May 22, 2019 at 17:32

1 @DmitryTimofeev Thanks for your link! I must definitely take the time to enhance this answer. – Boiethios May 23, 2019 at 7:54

6

Mutex is a simple method of locking to control access to shared resources.

- At the same time, only one thread can master a mutex, and threads with locked status can access shared resources.
- If another thread wants to lock a resource that has been mutexed, the thread hangs until the locked thread releases the mutex.

Read write locks are more complex than mutex locks.

- Threads using mutex lack read concurrency.
- When there are more read operations and fewer write operations, read-write locks can be used to improve thread read concurrency.

Let me summarize for myself:

- The implementation of read-write lock is more complex than that of mutual exclusion lock, and the performance is poor.
- The read-write lock supports simultaneous reading by multiple threads. The mutex lock does not support simultaneous reading by multiple threads, so the read-write lock has high concurrency.

Share Edit Follow

edited Aug 27, 2021 at 3:44

answered Aug 27, 2021 at 2:23

guoyanzhang
117 1 8