Stanford CS Department 2020 OS Quals

Reading list

- Alexander Matveev, Nir Shavit, Pascal Felber, and Patrick Marlier. Read-Log-Update. In *Proceedings of the 25th ACM SOSP*. 2015.

Assignment

Implement and benchmark a user-level RLU library in C++14 (or later) using C++ atomics. You should support the following functions described in the paper:

- `rlu_reader_lock`
- `rlu_reader_unlock`
- `rlu_dereference`
- `rlu_try_lock`
- `rlu_cmp_objs`
- `rlu_assign_ptr`
- `rlu_abort`
- `rlu_malloc`
- `rlu_free`

Note, however, that you don't need to have functions by these names or keep the exact same parameter lists, because you may be able to expose the same functionality implicitly using templates and operator overloading. Email two things to David Mazières by 5pm Friday May 22, 2020:

1. A tarfile of your implementation, and
2. A write-up of no more than 3 pages with no smaller than 11pt font and 1-inch margins. Your write-up must contain at least one graph with benchmark results.

If you wish, you can assume that your compiler properly implements `memory_order_consume`, though this is now deprecated. Alternatively, you can include some non-portable code, so long as you explain why and keep the non-portable code to a minimum. You can borrow code from the linux kernel (or other open-source operating systems) for the non-portable parts. Your implementation only needs to work on one architecture with one compiler.