Recitation 5 — Eraser

Relation to lecture

- Threads! You've seen the problems that occur with concurrency in lecture; this paper gives you a better idea of how tricky concurrent code is to debug.

Difficulties with threads

- Timing dependencies make concurrency bugs difficult to reproduce
- Bugs can appear long after the execution of the code that caused the bug
- Bugs are often caused by modules interacting; we may have written one of the modules, but not the other

Prior approaches (to Eraser)

- Some didn't detect races on dynamically allocated memory
- Others (e.g., "happens-before") have high overhead and were dependent on the scheduler.
- **Goal for Eraser:** low overhead, less dependent on scheduler

Lockset Algorithm

- Dynamically constructs the set of locks that can be associated with each accessed memory location. If a lockset becomes empty, that indicates a problem.
- Figure 3 in the paper gives a good example of this algorithm
- Extensions to handle common situations:
    - Allow initialization without synchronization, and read-only data after initialization, by keeping some additional state (Figure 4)
    - Augment the algorithm to handle reader/writer locks, which are a different type of lock that allows for multiple readers at once (but only one writer)

Implementation/Evaluation

- Implementation shows that binary rewriting is possible, the data structures involved are reasonable, Eraser runs in practice, etc.
- Eraser slows down applications
- Tested on a fairly wide variety of systems