Building a Better List-Set

EECS 3/495 “Rust”

Spring 2017
Can we do better?

Coarse-grained synchronization:

- Lock the whole object for each operation
Can we do better?

Coarse-grained synchronization:

- Lock the whole object for each operation
- Easy to reason about :-}
Can we do better?

Coarse-grained synchronization:

- Lock the whole object for each operation
- Easy to reason about :-(
- But sequential bottleneck :-(
Four strategies

1. Fine-grained synchronization
Four strategies

1. Fine-grained synchronization
   
   :-) Can synchronize on different parts of object concurrently
Four strategies

1. Fine-grained synchronization
   :-) Can synchronize on different parts of object concurrently
   :-) But lots of locking/unlocking overhead
Four strategies

1. Fine-grained synchronization
   :-) Can synchronize on different parts of object concurrently
   :-( But lots of locking/unlocking overhead

2. Optimistic synchronization
   :-) No need to lock while traversing
   :-( But need to validate, and may require expensive retries
Four strategies

1. Fine-grained synchronization
   -): Can synchronize on different parts of object concurrently
   -( : But lots of locking/unlocking overhead

2. Optimistic synchronization
   -): No need to lock while traversing
   -( : But need to validate, and may require expensive retries

3. Lazy synchronization
   -): Less work needed than optimistic synchronization
   -( : But contended operations still need to retry
Four strategies

1. Fine-grained synchronization
   :-) Can synchronize on different parts of object concurrently
   :-( But lots of locking/unlocking overhead

2. Optimistic synchronization
   :-) No need to lock while traversing
   :-( But need to validate, and may require expensive retries

3. Lazy synchronization
   :-) Less work needed than optimistic synchronization
   :-( But contended operations still need to retry

4. Lock-free synchronization
   :-) No longer at the mercy of the scheduler
   :-( But complex, and maybe high overhead