
COMP 322: Fundamentals of Parallel Programming

<https://wiki.rice.edu/confluence/display/PARPROG/COMP322>

Lecture 30: Advanced locking in Java

Vivek Sarkar
Department of Computer Science
Rice University
vsarkar@rice.edu



Reading vs. writing

- Recall that the use of synchronization is to protect interfering accesses
 - Multiple concurrent reads of same memory: *Not a problem*
 - Multiple concurrent writes of same memory: *Problem*
 - Multiple concurrent read & write of same memory: *Problem*

So far:

- If concurrent write/write or read/write might occur, use synchronization to ensure one-thread-at-a-time

But:

- This is unnecessarily conservative: we could still allow multiple simultaneous readers

Consider a hashtable with one coarse-grained lock

- So only one thread can perform operations at a time

But suppose:

- There are many simultaneous lookup operations
- insert operations are very rare



java.util.concurrent.locks.ReadWriteLock interface

```
interface ReadWriteLock {  
    Lock readLock();  
    Lock writeLock();  
}
```

- Even though the interface appears to just define a pair of locks, the semantics of the pair of locks is coupled as follows
 - Case 1: a thread has successfully acquired writeLock().lock()
 - No other thread can acquire readLock() or writeLock()
 - Case 2: no thread has acquired writeLock().lock()
 - Multiple threads can acquire readLock()
 - No other thread can acquire writeLock()
- java.util.concurrent.locks.ReadWriteLock interface is implemented by java.util.concurrent.locks.ReadWriteReentrantLock class



Example code

```
class Hashtable<K,V> {  
    ...  
    // coarse-grained, one lock for table  
    ReadWriteLock lk = new ReentrantReadWriteLock();  
    V lookup(K key) {  
        int bucket = hasher(key);  
        lk.readLock().lock(); // only blocks writers  
        ... read array[bucket] ...  
        lk.readLock().unlock();  
    }  
    void insert(K key, V val) {  
        int bucket = hasher(key);  
        lk.writeLock().lock(); // blocks readers and writers  
        ... write array[bucket] ...  
        lk.writeLock().unlock();  
    }  
}
```

