

Correct and Efficient Synchronization of Java Threads



Correct and Efficient Synchronization of Java™ Technology-based Threads

Doug Lea and William Pugh
<http://gee.cs.oswego.edu>
<http://www.cs.umd.edu/~pugh>

TS-754, Correct and Efficient Synchronization of Java Threads

Audience

- Assume you are familiar with basics of Java™ technology-based threads ("Java threads")
 - Creating, starting and joining threads
 - Synchronization
 - **wait** and **notifyAll**
- Will talk about things that surprised a lot of experts
 - Including us, James Gosling, Guy Steele, ... (others discovered many of these)



TS-754, Correct and Efficient Synchronization of Java Threads

Java Thread Specification

- Chapter 17 of the Java Language Spec
 - Chapter 8 of the Virtual Machine Spec
- Very, very hard to understand
 - Not even the authors understood it
 - Has subtle implications
 - That forbid standard compiler optimizations
 - All existing JVMs violate the specification
 - Some parts should be violated



TS-754, Correct and Efficient Synchronization of Java Threads

Safety Issues in Multithreaded Systems

- Many intuitive assumptions do not hold
- Some widely used idioms are not safe
 - Double-check idiom
 - Checking non-volatile flag for thread termination
- Can't use testing to check for errors
 - Some anomalies will occur only on some platforms
 - e.g., multiprocessors



TS-754, Correct and Efficient Synchronization of Java Threads

Revising the Thread Spec

- Work is underway to consider revising the Java Thread Spec
 - <http://www.cs.umd.edu/~pugh/java/memoryModel>
- Goals
 - Clear and easy to understand
 - Foster reliable multithreaded code
 - Allow for high performance JVMs
- Will affect JVMs
 - And badly written existing code
 - Including parts of Sun's JDK



TS-754, Correct and Efficient Synchronization of Java Threads

Three Aspects of Synchronization

- Atomicity
 - Locking to obtain mutual exclusion
- Visibility
 - Ensuring that changes to object fields made in one thread are seen in other threads
- Ordering
 - Ensuring that you aren't surprised by the order in which statements are executed



TS-754, Correct and Efficient Synchronization of Java Threads

Correct and Efficient Synchronization of Java Threads

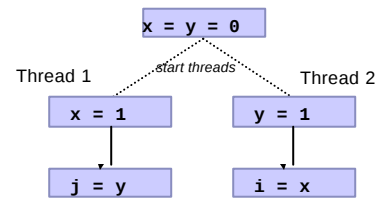
Don't Be Too Clever

- People worry about the cost of synchronization
 - They try to devise schemes to communicate between threads
 - Without using synchronization
- Very difficult to do correctly
 - Inter-thread communication without synchronization is not intuitive



TS-754, Correct and Efficient Synchronization of Java Threads

Quiz Time

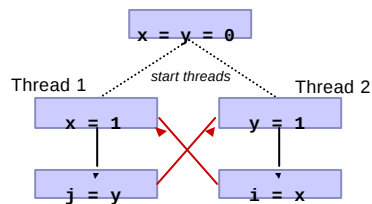


Can this result in $i = 0$ and $j = 0$?



TS-754, Correct and Efficient Synchronization of Java Threads

Answer: Yes!



How can $i = 0$ and $j = 0$?



TS-754, Correct and Efficient Synchronization of Java Threads

How Can This Happen?

- Compiler can reorder statements
 - Or keep values in registers
- Processor can reorder them
- On multi-processor, values not synchronized in global memory
- Must use synchronization to enforce visibility and ordering
 - As well as mutual exclusion



TS-754, Correct and Efficient Synchronization of Java Threads

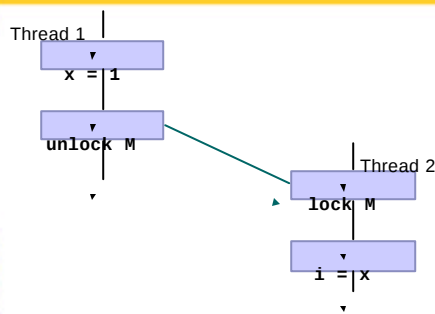
Synchronization Actions (approximately)

```
// block until obtain lock
synchronized(anObject) {
    // get main memory value of field1
    // and field2
    int x = anObject.field1;
    int y = anotherObject.field2;
    anotherObject.field3 = x+y;
    // commit value of field3 to main
    // memory
}
// release lock
moreCode();
```



TS-754, Correct and Efficient Synchronization of Java Threads

When Are Actions Visible to Other Threads?



TS-754, Correct and Efficient Synchronization of Java Threads

Correct and Efficient Synchronization of Java Threads

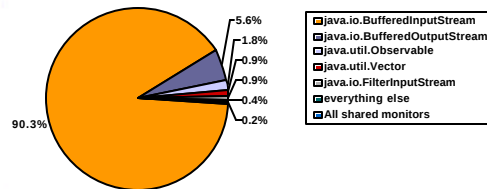
Cost of Synchronization

- Few good public multithreaded benchmarks
 - See us if you want to help
- Volano Benchmark
 - Most widely used server benchmark
 - Multithreaded chat room server
 - Client performs 4.8M synchronizations
 - 8K useful (0.2%)
 - Server 43M synchronizations
 - 1.7M useful (4%)



TS-754, Correct and Efficient Synchronization of Java Threads

Synchronization in VolanoMark Client



7,684 synchronizations on shared monitors
4,828,130 thread local synchronizations



TS-754, Correct and Efficient Synchronization of Java Threads

Cost of Synchronization in VolanoMark

- Removed synchronization of
 - java.io.BufferedInputStream
 - java.io.BufferedOutputStream
- Performance (2 processor Ultra 60)
 - Larger is better
 - HotSpot (1.3 beta)
 - Original: 4788
 - Altered: 4923 (+3%)
 - Exact VM (1.2.2)
 - Original: 6649
 - Altered: 6874 (+3%)



TS-754, Correct and Efficient Synchronization of Java Threads

Most Synchronization is on Thread Local Objects

- Synchronization on thread local object
 - Is useless
 - Current spec says it isn't quite a no-op
 - But very hard to use usefully
 - Revised spec will likely make it a no-op
- Largely arises from using synchronized classes
 - In places where not required



TS-754, Correct and Efficient Synchronization of Java Threads

Synchronize when Needed

- Places where threads interact
 - Need synchronization
 - Need careful thought
 - Need documentation
 - Cost of required synchronization not significant
 - For most applications
 - No need to get tricky
- Elsewhere, using a synchronized class can be expensive



TS-754, Correct and Efficient Synchronization of Java Threads

Synchronized Classes

- Some library classes are synchronized
 - Vector, Hashtable, Stack
 - Most Input/Output Streams
- Contrast with 1.2 Collection classes
 - By default, not synchronized
 - Can request synchronized version
- Using synchronized classes
 - Often doesn't suffice for concurrent interaction



TS-754, Correct and Efficient Synchronization of Java Threads

Correct and Efficient Synchronization of Java Threads

Synchronized Collections Aren't Always Enough

- Transactions (THIS DOESN'T WORK)

```
ID getId(String name) {  
    ID x = (ID) h.get(name);  
    if (x == null) {  
        x = new ID();  
        h.put(name, x);  
    }  
    return x;  
}
```

- Iterators
 - Can't modify collection while another thread is iterating through it



TS-754, Correct and Efficient Synchronization of Java Threads

Concurrent Interactions

- Often need entire transactions to be atomic
 - Reading and updating a Map
 - Writing a record to an OutputStream
- OutputStreams are synchronized
 - Can have multiple threads trying to write to the same OutputStream
 - Output from each thread is non-deterministically interleaved
 - Essentially useless



TS-754, Correct and Efficient Synchronization of Java Threads

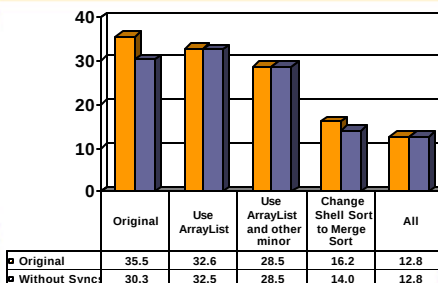
Cost of Synchronization in SpecJVM DB Benchmark

- Program in the SpecJVM benchmark
- Does lots of synchronization
 - > 53,000,000 syncs
 - 99.9% comes from use of Vector
 - Benchmark is single threaded, all of it is useless
- Tried
 - Remove synchronizations
 - Switching to ArrayList
 - Improving the algorithm



TS-754, Correct and Efficient Synchronization of Java Threads

Execution Time of SpecJVM _209_db, Hotspot Server



TS-754, Correct and Efficient Synchronization of Java Threads

Lessons

- Synchronization cost can be substantial
 - 10-20% for DB benchmark
 - Consider replacing all uses of Vector, Hashtable and Stack
- Use profiling
- Use better algorithms!
 - Cost of stupidity higher than cost of synchronization
 - Used built-in merge sort rather than hand-coded shell sort



TS-754, Correct and Efficient Synchronization of Java Threads

Designing Fast Code

- Make it right before you make it fast
- Avoid synchronization
 - Avoid sharing across threads
 - Don't lock already-protected objects
 - Use immutable fields and objects
 - Use volatile (carefully)
- Avoid contention
 - Reduce lock scopes
 - Reduce lock durations



TS-754, Correct and Efficient Synchronization of Java Threads

Correct and Efficient Synchronization of Java Threads

Unsynchronized Reads/Writes of References

- Beware of unsynchronized getX/setX methods that return a reference
 - Same problems as double check
 - Doesn't help to synchronize **only** setX

```
private Color color;
void setColor(int rgb) {
    color = new Color(rgb);
}
Color getColor() {
    return color;
}
```



TS-754, Correct and Efficient Synchronization of Java Threads

Problems

- Don't assume another thread will see your writes
 - Just because you did them
- Calling sleep doesn't guarantee you see changes made while you slept
 - Nothing to force thread that called stop to push change out of registers/cache



TS-754, Correct and Efficient Synchronization of Java Threads

Wrap-up

- Cost of synchronization operations can be significant
 - But cost of *needed* synchronization rarely is
- Thread interaction needs careful thought
 - But not too clever
- Need for synchronization...



TS-754, Correct and Efficient Synchronization of Java Threads

Wrapup - Synchronization

- Communication between threads
 - Requires both threads to synchronize
 - Or communication through volatile fields
- Synchronizing everything
 - Is rarely necessary
 - Can be expensive (3%-20% overhead)
 - May lead to deadlock
 - May not provide enough synchronization
 - e.g., transactions



TS-754, Correct and Efficient Synchronization of Java Threads