The Java Memory Model

March 16, 2011

Sources

- Jeremy Manson and Brian Goetz. JSR 133 (Java Memory Model) FAQ. February 2004.
- Jeremy Manson, William Pugh and Sarita V. Adve. The Java Memory Model. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* , pages 378–391, Long Beach, CA, USA, January 2005. ACM.
- James Gosling, Bill Joy, Guy Steele and Gilad Bracha. *The Java Language Specification.* Chapter 17. Addison-Wesley, Reading, MA, USA, 2nd edition, 2000.

What is a Memory Model?

A memory model defines necessary and sufficient conditions for knowing that writes to memory by other processors are visible to the current processor, and writes by the current processor are visible to other processors.

Why do we Need a Memory Model?

Local memory (caches, registers, and other hardware and compiler optimizations) can improve performance tremendously, but it presents a host of new challenges. What, for example, happens when two processors examine the same memory location at the same time? Under what conditions will they see the same value? A memory model provides answers to these questions.

Synchronized Methods/Blocks

After we exit a synchronized block, we release the monitor, which has the effect of flushing the cache to main memory, so that writes made by this thread can be visible to other threads.

Before we can enter a synchronized block, we acquire the monitor, which has the effect of invalidating the local processor cache so that variables will be reloaded from main memory. We will then be able to see all of the writes made visible by the previous release.

Volatile

Each read or write of a volatile field acts like "half" a synchronization, for purposes of visibility.

Each read of a volatile will see the last write to that volatile by any thread; in effect, they are designated by the programmer as fields for which it is never acceptable to see a "stale" value as a result of caching or reordering.

Volatile

```
class Volatile
{
   private int value;
   private volatile boolean initialized;

   public void write(int value)
   {
      this.value = value;
      this.initialized = true;
   }

   public void use()
   {
      if (this.initialized)
      {
         // write has been invoked
         ...
      }
   }
}
```

What is a Partial Order?

Definition

Let X be a set. A binary relation $\sqsubseteq$ on X is a partial order if for all x, y and $z \in X$,

- $x \sqsubseteq x$,
- if $x \sqsubseteq y$ and $y \sqsubseteq x$ then $x = y$, and
- if $x \sqsubseteq y$ and $y \sqsubseteq z$ then $x \sqsubseteq z$.

Partial Orders

- The standard less-than-or-equal relation $\leq$ on the real numbers.
- The relation $\cdot$divides$\cdot$ on the natural numbers.
- The inclusion relation $\subseteq$ on the powerset of a given set.

What is a Total Order?

Definition

Let X be a set. A binary relation $\sqsubseteq$ on X is a total order if for all x, y and $z \in X$,

- if $x \sqsubseteq y$ and $y \sqsubseteq x$ then $x = y$,
- if $x \sqsubseteq y$ and $y \sqsubseteq z$ then $x \sqsubseteq z$, and
- $x \sqsubseteq y$ or $y \sqsubseteq x$.

Total Orders

- The standard less-than relation $<$ on the real numbers.
- The lexicographic order on words.