Java内存模型的历史变迁

发表于2015-05-20 11:00| 1970次阅读| 来源CSDN| 2 条评论| 作者程晓明

[特别策划](http://www.csdn.net/tag/%E7%89%B9%E5%88%AB%E7%AD%96%E5%88%92/news)[Java](http://www.csdn.net/tag/java/news)[Java20周年](http://www.csdn.net/tag/java20%E5%91%A8%E5%B9%B4/news)[《程序员》电子刊](http://www.csdn.net/tag/%E3%80%8A%E7%A8%8B%E5%BA%8F%E5%91%98%E3%80%8B%E7%94%B5%E5%AD%90%E5%88%8A/news)

**摘要：**本文通过介绍Java的新/旧内存模型，来展示Java技术的历史变迁。

本文通过介绍Java的新/旧内存模型，来展示Java技术的历史变迁。

**旧的Java内存模型**

Java使用的是共享内存的并发模型，在线程之间共享变量。Java语言定义了线程模型规范，通过内存模型控制线程与变量的交互，从而实现Java线程之间的通信。在JDK5之前，Java一直使用的是旧内存模型。如图1所示。变量保存在由所有线程共享的主内存中，主内存中的变量称为mastingcopy。每个线程都有一个工作内存，它保存变量的workingcopy。旧的内存模型定义了若干规则，通过这些规则来保证线程何时将主内存中的mastingcopy传送到线程的工作内存中；以及线程何时将工作内存中的workingcopy传送回主内存。旧内存模型使用8个操作来定义线程可以执行的动作。

* read（读）操作：主内存把mastingcopy传送到线程的工作内存，以供后面的load操作使用。
* load（装载）操作：线程将由read操作从主内存传送过来的值，放入工作内存中。
* use（使用）操作：线程将变量的workingcopy传送到线程执行引擎。
* assign（赋值）操作：线程将变量值从线程执行引擎传送到线程的工作内存中。
* store（存储）操作：线程将变量的workingcopy传送到主内存，供后面的write操作使用。
* write（写）操作：主内存将由store操作传送过来的值，放入主内存中。
* lock（锁定）操作：线程获得指定对象的锁。
* unlock（解锁）操作：线程释放指定对象的锁。

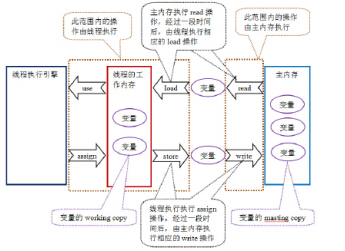


图1 Java旧内存模型

这里的关键是，由于read操作是由主内存执行，而对应的load是由线程执行，read操作和load操作之间是松散耦合的。也就是说，主内存和线程工作内存之间的变量传递是松散耦合的。同样，由于store操作是由线程执行，而对应的write是由主内存执行，store操作和write操作之间是松散耦合的。也就是说，线程工作内存和主内存之间的变量传递是松散耦合的。旧Java内存模型对Java实现如何执行变量的读/写，加锁/解锁，以及volatile变量的读/写，定义了非常严格的规则。这些规则非常复杂，具体详情请参考《JVM规范》，这里就不赘述了。旧Java内存模型通过这些复杂的规则，来保证多线程程序的线程之间，可以可靠地传递共享变量，从而保证多线程程序的正确性。

**新的Java内存模型**

从JDK5开始，Java使用新的内存模型，新内存模型完全抛弃了旧内存模型的主内存和工作内存的概念，也抛弃了旧内存模型的8个内存操作。也就是说，新内存模型完全是重新设计的。

新内存模型引入了一个新的概念，叫happens-before。happens-before的概念最初由LeslieLamport在其一篇影响深远的论文（《Time,ClocksandtheOrderingofEventsinaDistributedSystem》）中提出。LeslieLamport使用happens-before来定义分布式系统中，事件之间的一个偏序关系(partialordering)。LeslieLamport在这篇论文中给出了一个分布式算法，该算法可以将该偏序关系扩展为某种全序关系。

JSR-133使用happens-before的概念来指定两个操作（这里的操作是指程序中对变量的读/写，对锁的加锁和解锁）之间的执行顺序。新内存模型定义了如下的happens-before规则。

* 程序顺序规则：一个线程中的每个操作，happensbefore于该线程中的任意后续操作。
* 监视器锁规则：对一个锁的解锁，happens-before于随后对这个锁的加锁。
* volatile变量规则：对一个volatile域的写，happensbefore于任意后续对这个volatile域的读。
* 传递性：如果Ahappens-beforeB，且BhappensbeforeC，那么Ahappens-beforeC。
* start()规则：如果线程A执行操作ThreadB.start()（启动线程B），那么A线程的ThreadB.start()操作happensbefore于线程B中的任意操作。
* join()规则：如果线程A执行操作ThreadB.join()并成功返回，那么线程B中的任意操作happens-before于线程A从ThreadB.join()操作成功返回。

由于两个操作可以在一个线程之内，也可以是在不同线程之间。因此JMM可以通过happens-before关系向程序员提供跨线程的内存可见性保证（如果A线程的写操作a与B线程的读操作b之间存在happens-before关系，尽管a操作和b操作在不同的线程中执行，但JMM

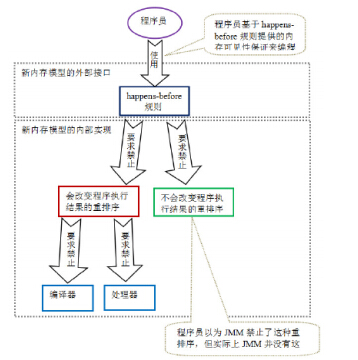


图2新内存模型的设计示意图

向程序员保证a操作将对b操作可见）。在新内存模型向程序员提供happens-before规则，程序员只需要与happens-before打交道即可，因此Java程序员的学习负担大大降低。同时，新内存模型允许不会改变程序结果的重排序，这可以最大限度地放松对编译器和处理器的束缚，新内存模型的执行性能比旧内存模型要好。

**作者简介：**程晓明Java软件工程师，关注并发与NIO，《深入理解Java内存模型》的作者。现就职于南京富士通南大软件技术有限公司，从事富士通Interstage应用服务器的开发。