链滴

volidate大集合

作者：oldcaptain

原文链接：https://ld246.com/article/1413023123897

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

我们知道,在 Java 中设置变量值的操作,除了 long 和 double 类型的变量外都是原子操作也就是说,对于变量值的简单读写操作没有必要进行同步。
这在 JVM 1.2 之前, Java 的内存模实现总是从主存读取变量,是不需要进行特别的注意的。而随着 JVM 的成熟和优化,现在在多线程环境 volatile 关键字的使用变得非常重要。在当前的 Java 内存模型下,线程可以把变量保存在本地内存(比机器的寄存器)中,而不是直接在主存中进行读写。这就可能造成一个线程在主存中修改了一个变量的值而另外一个线程还继续使用它在寄存器中的变量值的拷贝,造成数据的不一致。要解决这个问题,只需像在本程序中的这样,把该变量声明为 volatile (不稳定的)即可,这就指示 JVM ,这个变量是不稳定的,每使用它都到主存中进行读取。一般说来,多任务环境下各任务间共享的标志都应该加 volatile 修饰。 &bsp;
Volatile 修饰的成员变量在每次被线程访问时,都强迫从共享内存中重读该成员变量的值。且,当成员变量发生变化时,强迫线程将变化值回写到共享内存。这样在任何时刻,两个不同的线程总是到某个成员变量的同一个值。
Java 语言规范中指出:为了获得最佳速度,允许线程保存享成员变量的私有拷贝,而且只当线程进入或者离开同
步代码块时才与共享成员变量的原始值比。
这样当多个线程同时与某个对象交互时,就必须要注意到要让线程及时的得到共享员变量的变化。
而 volatile 关键字就是提示 VM :对于这个成员变量不能保存它的私拷贝,而应直接与共享成员变量交互。
使用建议:在两个或者更多的线程访问的成员变上使用 volatile 。当要访问的变量已在 synchronized 代码块中,或者为常量时,不必用。
由于使用 volatile 屏蔽掉了 VM 中必要的代码优化,所以在效率上比较低,因此一定在必时才使用此关键字。

Volatile 变量
Volatile 变量具有 synchronized 的可见性特性,但不具备原子特性。这就是说线程能够自动发现 volatile 变量的最新值。Volatile 变量可用于提供线程全,但是只能应用于非常有限的一组用例:多个变量之间或者某个变量的当前值与修改后值之间没有约 。因此,单独使用 volatile 还不足以实现计数器、互斥锁或任何具有与多个变量相关的不变式(Invariant)的类(例如 “start<=end”)。
出于简易性或可伸缩性的考虑,您可能倾向于使用 volatile 变而不是锁。当使用 volatile 变量而非锁时,某些习惯用法(idiom)更加易于编码和阅读。此外,volatile 量不会像锁那样造成线程阻塞,因此也很少造成可伸缩性问题。
在某些情况下,如果读操作远远于写操作,volatile 变量还可以提供优于锁的性能优势。
正确使用 volatile 变量的条件您只能在限的一些情形下使用 volatile 变量替代锁。

要使 volatile 变量提供理想的线程安全,必须同时满足下面两个条件:
对变量的写操作不依赖于当前值。
该变量没有包含在具有其他变量的不变式中。

实际上,这些条件表明,可以被写入 volatile 变量这些有效值独立于任何程序的状态,包括变量的当前状态。
第一个条件的限制使 volatile 变量不能用作线程安全计数器。虽然增量操作(x++)看上去类似一单独操作,实际上它是一个由读取-修改-写入操作序列组成的组合操作,必须以原子方式执行,而 volatile 不能提供必须的原子特性。实现正确的操作需要使 x 的值在操作期间保持不变,而 volatle 变量无法实现这点。(然而,如果将值调整为只从单个线程写入,那么可以忽略第一个条件。)
大多数编程情形都会与这两个条件的其中之一冲突,使得 volatil 变量不能像 synchronized 那样普遍适用于实现线程安全。