Java并发编程之CAS

原文地址：作者： Jakob Jenkov 译者：张坤

CAS（Compare and swap）比较和替换是设计并发算法时用到的一种技术。简单来说，比较和替换是使用一个期望值和一个变量的当前值进行比较，如果当前变量的值与我们期望的值相等，就使用一个新值替换当前变量的值。这听起来可能有一点复杂但是实际上你理解之后发现很简单，接下来，让我们跟深入的了解一下这项技术。

CAS的使用场景

在程序和算法中一个经常出现的模式就是“check and act”模式。先检查后操作模式发生在代码中首先检查一个变量的值，然后再基于这个值做一些操作。下面是一个简单的示例：

```
01 class MyLock {
02
03     private boolean locked = false;
04
05     public boolean lock() {
06         if(!locked) {
07             locked = true;
08             return true;
09         }
10         return false;
11     }
12 }
```

上面这段代码，如果用在多线程的程序会出现很多错误，不过现在请忘掉它。

如你所见，lock()方法首先检查locked>成员变量是否等于false，如果等于，就将locked设为true。

如果同个线程访问同一个MyLock实例，上面的lock()将不能保证正常工作。如果一个线程检查locked的值，然后将其设置为false，与此同时，一个线程B也在检查locked的值，又或者，在线程A将locked的值设为false之前。因此，线程A和线程B可能都看到locked的值为false，然后两者都基于这个信息做一些操作。

为了在一个多线程程序中良好的工作，”check then act” 操作必须是原子的。原子就是说”check“操作和”act“被当做一个原子代码块执行。不存在多个线程同时执行原子块。

下面是一个代码示例，把之前的lock()方法用synchronized关键字重构成一个原子块。

```
01 class MyLock {
02
03     private boolean locked = false;
04
```

```
    public synchronized boolean lock() {
        if(!locked) {
            locked = true;
            return true;
        }
        return false;
    }
}
```

现在lock()方法是同步的，所以，在某一时刻只能有一个线程在同一个MyLock实例上执行它。

原子的lock方法实际上是一个”compare and swap“的例子。

CAS用作原子操作

现在CPU内部已经执行原子的CAS操作。Java5以来，你可以使用java.util.concurrent.atomic包中的一些原子类来使用CPU中的这些功能。

下面是一个使用AtomicBoolean类实现lock()方法的例子：

```
public static class MyLock {
    private AtomicBoolean locked = new AtomicBoolean(false);

    public boolean lock() {
        return locked.compareAndSet(false, true);
    }

}
```

locked变量不再是boolean类型而是AtomicBoolean。这个类中有一个compareAndSet()方法，它使用一个期望值和AtomicBoolean实例的值比较，和两者相等，则使用一个新值替换原来的值。在这个例子中，它比较locked的值和false，如果locked的值为false，则把修改为true。

如果值被替换了，compareAndSet()返回true，否则，返回false。

使用Java5+提供的CAS特性而不是使用自己实现的的好处是Java5+中内置的CAS特性可以让你利用底层的你的程序所运行机器的CPU的CAS特性。这会使还有CAS的代码运行更快。

方 腾飞
2015/04/14 11:10下午

翻译的不错，这个系列还有一些未翻译完成的，有兴趣可以翻译下。

http://ifeve.com/java-concurrency-thread-directory/

DragonKing

2015/12/22 3:04下午

这篇翻译的没有之前几篇翻译的好，语句不通顺，错别字地方有点多。希望再接再厉

赵玉春
2017/01/20 5:10下午

简直写的一塌糊涂

tinylcy
2017/02/15 3:29下午

有写的不好的请指出。

cenyol
2017/03/16 12:54下午

如果同个线程访问同一个MyLock实例，上面的lock()将不能保证正常工作。如果一个线程检查locked的值，然后将其设置为false，与此同时，一个线程B也在检查locked的值，又或者，在线程A将locked的值设为false之前。因此，线程A和线程B可能都看到locked的值为false，然后两者都基于这个信息做一些操作。

这里第一句中的同个线程，是不是改为多个线程更好呢？