

# CAS原理分析

原创2014年02月26日 14:03:079476

## 一、锁机制

常用的锁机制有两种：

- 1、悲观锁：假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作。悲观锁的实现，往往依靠底层提供的锁机制；悲观锁会导致其它所有需要锁的线程挂起，等待持有锁的线程释放锁。
- 2、乐观锁：假设不会发生并发冲突，每次不加锁而是假设没有冲突而去完成某项操作，只在提交操作时检查是否违反数据完整性。如果因为冲突失败就重试，直到成功为止。乐观锁大多是基于数据版本记录机制实现。为数据增加一个版本标识，比如在基于数据库表的版本解决方案中，一般是通过为数据库表增加一个“version”字段来实现。读取出数据时，将此版本号一同读出，之后更新时，对此版本号加一。此时，将提交数据的版本数据与数据库表对应记录的当前版本信息进行比对，如果提交的数据版本号大于数据库表当前版本号，则予以更新，否则认为是过期数据。

乐观锁的缺点是不能解决脏读的问题。

在实际生产环境里边,如果并发量不大且不允许脏读,可以使用悲观锁解决并发问题；但如果系统的并发非常大的话,悲观锁定会带来非常大的性能问题,所以我们就要选择乐观锁定的方法。

锁机制存在以下问题：

- (1) 在多线程竞争下，加锁、释放锁会导致比较多的上下文切换和调度延时，引起性能问题。
- (2) 一个线程持有锁会导致其它所有需要此锁的线程挂起。
- (3) 如果一个优先级高的线程等待一个优先级低的线程释放锁会导致优先级倒置，引起性能风险。

## 二、CAS 操作

JDK 5之前Java语言是靠synchronized关键字保证同步的，这是一种独占锁，也是是悲观锁。java.util.concurrent(J.U.C)种提供的atomic包中的类，使用的是乐观锁，用到的机制就是CAS，CAS ( Compare and Swap ) 有3个操作数，内存值V，旧的预期值A，要修改的新值B。当且仅当预期值A和内存值V相同时，将内存值V修改为B，否则什么都不做。

现代的CPU提供了特殊的指令，允许算法执行读-修改-写操作，而无需害怕其他线程同时修改变量，因为如果其他线程修改变量，那么CAS会检测它（并失败），算法可以对该操作重新计算。而 compareAndSet() 就用这些代替了锁定。

以AtomicInteger为例，研究在没有锁的情况下是如何做到数据正确性的。

```
[java]
1. public class AtomicInteger extends Number implements java.io.Serializable {
2.
3.     private volatile int value;
4.
5.
6.
7.     public final int get() {
8.         return value;
9.     }
10.
11.     public final int getAndIncrement() {
12.         for (;;) {
13.             int current = get();
14.             int next = current + 1;
15.             if (compareAndSet(current, next))
16.                 return current;
17.         }
18.     }
19.
20.     public final boolean compareAndSet(int expect, int update) {
21.         return unsafe.compareAndSwapInt(this, valueOffset, expect, update);
22.     }
```

字段value需要借助volatile原语，保证线程间的数据是可见的（共享的）。这样在获取变量的值的时候才能直接读取。然后来看看++i是怎么做到的。getAndIncrement采用了CAS操作，每次从内存中读取数据然后将此数据和+1后的结果进行CAS操作，如果成功就返回结果，否则重试直到成功为止。而compareAndSet利用JNI



原创	粉丝	喜欢	评论
90	494	277	179

恒

等级： 博客 7 访问量：162万+

积分：1万+ 排名：1978

- 博主最新文章 更多文章
- Cloud Native和微服务
- Actor模型
- Serverless和FaaS
- Spring Cloud Bus
- 背压

文章分类	
C/C++	38篇
Hadoop	35篇
Java	65篇
Java并发编程	16篇
Linux	51篇
NoSQL	3篇
展开	

文章存档	
2018年3月	5篇
2018年2月	2篇
2018年1月	3篇
2017年12月	1篇
2017年7月	2篇
2017年5月	2篇
展开	

- 博主热门文章
- Spring@Autowired注解与自动装配 452102
- Java动态代理的两种实现方法 94668
- 消息队列的两种模式 49715
- Ubuntu下编译的第一个内核模块 46118
- C++中int和char[]之间的转换 40882
- ZooKeeper 节点类型 30852
- socket 的通信过程 30107
- Spring中Bean初始化的三种方法 27530
- Java实现定时调度的三种方法

```
1. public final boolean compareAndSet(int expect, int update) {
2.     return unsafe.compareAndSwapInt(this, valueOffset, expect, update);
3. }
```

整体的过程就是这样子的，利用CPU的CAS指令，同时借助JNI来完成Java的非阻塞算法。其它原子操作都是利用类似的特性完成的。

而整个J.U.C都是建立在CAS之上的，因此对于synchronized阻塞算法，J.U.C在性能上有了很大的提升。

CAS第一个问题是会导致“ABA问题”。

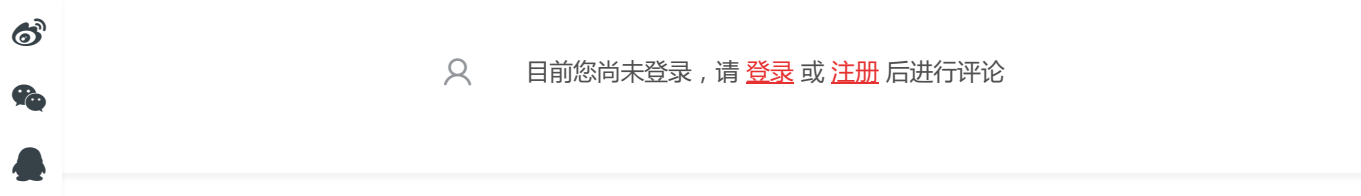
aba实际上是乐观锁无法解决脏数据读取的一种体现。CAS算法实现一个重要前提需要取出内存中某时刻的数值，而在下时刻比较并替换，那么在这个时间差类会导致数据的变化。比如说一个线程one从内存位置V中取出A，这时候另一个线程two也从内存中取出A，并且two进行了一些操作变成了B，然后two又将V位置的数据变成A，这时候线程one进行CAS操作发现内存中仍然是A，然后one操作成功。尽管线程one的CAS操作成功，但是不代表这个过程就是没有问题的。如果链表的头在变化了两次后恢复了原值，但是不代表链表就没有变化。因此AtomicStampedReference/AtomicMarkableReference就很有用了。

AtomicMarkableReference 类描述的一个<Object,Boolean>的对，可以原子的修改Object或者Boolean的值，这种数据结构在一些缓存或者状态描述中比较有用。这种结构在单个或者同时修改Object/Boolean的时候能够有效的提高吞吐量。

AtomicStampedReference 类维护带有整数“标志”的对象引用，可以用原子方式对其进行更新。对比AtomicMarkableReference 类的<Object,Boolean>，AtomicStampedReference 维护的是一种类似<Object,int>的数据结构，其实就是对对象（引用）的一个并发计数（标记版本戳stamp）。但是与AtomicInteger不同的是，此数据结构可以携带一个对象引用（Object），并且能够对此对象和计数同时进行原子操作。

```
1. [java]
```


版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/HEYUTAO007/article/details/19975665>




目前您尚未登录，请 [登录](#) 或 [注册](#) 后参与评论

**JAVA CAS原理深度分析**  Hsuxu 2013年07月25日 13:07 123659


看了一堆文章，终于把JAVA CAS的原理深入分析清楚了。 感谢GOOGLE强大的搜索，借此挖苦下百度，依靠百度什么都学习不到！ 参考文档：<http://www.blogjava.net...>

**java中CAS算法保证原子性 无锁编程**  glory1234work2115 2016年03月07日 23:23 1785


悲观锁和乐观锁 悲观锁会导致其它所有需要锁的线程挂起，等待持有锁的线程释放锁。乐观锁每次不加锁而是假设没有冲突而去完成某项操作，如果因为冲突失败就重试，直到成功为止。 synchronized...

**原子变量和CAS算法**  xiangwanpeng 2017年02月10日 00:46 651


先运行下面一段程序:package concurrent;class AtomicDemo implements Runnable { private int serialNumber = 0;...

**无锁算法——CAS原理**  Roy\_70 2017年04月09日 09:32 2281

一、无锁算法CAS（比较与交换，Compare and swap）是一种有名的无锁算法。无锁编程，即不使用锁的情况下实现多线程之间的变量同步，也就是在没有线程被阻塞的情况下实现变量的同步，所以也叫非...

**CAS原理分析**  ya\_1249463314 2017年01月06日 15:50 1583

在JDK 5之前Java语言是靠synchronized关键字保证同步的，这会导致有锁（后面的章节还会谈到锁）。 锁机制存在以下问题：（1）在多线程竞争下，加锁、释放锁会导致比较多的上下文切换和调度...

**对cas算法的理解**  xiongxianze 2017年04月19日 01:09 486

cas算法主要关心3个值：内存值V，预期值A，要更新的新值B 如下图所示：注：t1，t2线程是同时更新同一变量56的值因为t1和t2线程都同时去访问同一变量56，所以他们会把主内存的值完全拷贝一份到...

24314

联系我们



请扫描二维码联系客服

 webmaster@csdn.net

 400-660-0108

 QQ客服  客服论坛

关于 招聘 广告服务  百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

CAS算法(一)

u014231889

2017年07月09日 13:43

228

CAS算法主要关心3个值：内存值V，预期值A，要更新的新值B。注：t1，t2线程是同时更新同一变量56的值因为t1和t2线程都同时去访问同一变量56，所以他们会把住内存的值完全拷贝一份到自己的工作内存...

👍5

JAVA非阻塞同步算法与CAS无锁算法

qq\_27602093

2017年04月04日 20:44

463

JDK1.5之前靠synchronized关键字保持同步，采用独占方式访问变量 乐观锁与悲观锁 独占锁是悲观锁，然而synchronized是独占锁(悲观的)，它会导致其他需要锁的线程挂起，等待那...

💬

AS(compareAndSet)算法及简单应用AtomicInteger

xiaoxiaosunzhao

2011年05月19日 21:20

8568

参照链接:http://www.blogjava.net/syniii/archive/2010/11/18/338387.html?opt=admincas算法应用的场合：AsyncTask中为Th...

JAVA线程并发性之CAS算法，模拟实现代码

crpxnmmafq


2017年07月04日 12:43

226

在了解算法之前，我们先对回顾基本概念：原子性：具有不可分割性。比如 a=0；（a非long和double类型）这个操作是不可分割的，那么我们说这个操作时原子操作。再比如：a++；这个操作实际是a ...

50万码农评论：英语对于程序员有多重要？

不背单词和语法，一个公式学好英语



【JAVA笔记——道】并发编程CAS算法

wang135139

2015年12月26日 19:41

1744

CAS (Compare And Swap) 原子操作实现多线程同步 需要传入1.目标指针reg；2.被比较值oldval；3.更新值newval 执行过程如下Created with Raph...

理解CAS算法在JAVA中的作用

bluetjs

2016年08月20日 14:58

802

http://www.cnblogs.com/onlywujun/articles/3529572.html 在JDK 5之前Java语言是靠synchronized关键字保证同步的，这会导致有锁...

什么是 CAS 机制？

zhangjq520

2018年01月02日 15:18

410

CAS机制的简单介绍

CAS SSO 工作机制（每一步请求详述）

liu251890347

2014年07月02日 19:58

979

下面分析下CAS SSO(以3.4.5为例)的基本工作原理。先假定一个应用场景，如图，有两个Web应用分别是webapp1、webapp2(Spring Security 3应用)。一个认证服务器C...

利用CAS机制实现多进程，多线程下的无锁并发控制

tenfyguo

2012年12月31日 18:37

4501

CAS机制：CAS，又称Compare-and-Swap，代表一种原子操作，为每一个Node在Set的时候分配一个cas值，（本质是版本号，返回的Node和存储Node的cas值一样，...

Java的CAS机制

yyd19921214

2016年01月28日 15:57

256

http://blog.csdn.net/hsuxu/article/details/9467651

浅谈CAS机制

nakiri\_arisu

2018年01月29日 12:31

211

这里只是浅谈一下CAS机制，有机会的话后续会深入 CAS 背景 机制 为什么具有原子性 缺点 ABA问题 ABA问题的解决方案 CAS 背景 尽管J...

CAS原理

yuan1013922969

2016年06月13日 10:28

1685

企业的信息化过程是一个循序渐进的过程，在企业各个业务网站逐步建设的过程中，根据各种业务信息水平的需要构建了相应的应用系统，由于这些应用系统一般是 在不同的时期开发完成的，各应用系统由于功能侧重、设计方...

CAS机制

wanglei0622

2016年04月20日 15:43

284

CAS，Compare and Swap，翻译成比较并交换：有3个操作数，内存值V，旧的预期值A，要修改的新值B。当且仅当预期值A和内存值V相等时，将内存值V的值修改为B，否则什么都不做。