

返回课程

直播时间: 3月8日 20:00--22:00

## J.U.C包内的原子操作封装类

AtomicBoolean: 原子更新布尔类型

AtomicInteger: 原子更新整型

AtomicLong: 原子更新长整型

AtomicIntegerArray: 原子更新整型数组里的元素。

AtomicLongArray: 原子更新长整型数组里的元素。

AtomicReferenceArray: 原子更新引用类型数组里的元素。

AtomicIntegerFieldUpdater: 原子更新整型的字段的更新器。

AtomicLongFieldUpdater: 原子更新长整型字段的更新器。

AtomicReferenceFieldUpdater: 原子更新引用类型里的字段。

AtomicReference: 原子更新引用类型。

AtomicStampedReference: 原子更新带有版本号的引用类型

AtomicMarkableReference: 原子更新带有标记位的引用类型。

1.8更新

计数器增强版, 高并发下性能更好

更新器: DoubleAccumulator、LongAccumulator

计数器: DoubleAdder、LongAdder

原理: 分成多个操作单元, 不同线程更新不同的单元

只有需要汇总的时候才计算所有单元的操作

场景: 高并发频繁更新、不太频繁地读取

公告 暂无公告

42人正在观看

随风飘散ykt45048627997... 19:59

1

新飞kt471731349523291 19:59

1

张瑞印000 19:59

1

王瑞浩ykt1079089124108... 19:59

1

孙文耀 19:59

1

曹瑞浩ykt1079089124108... 20:00

1

曹瑞浩ykt1079089124108... 20:00

1

曹瑞浩ykt1079089124108... 20:01

1

我要发言

发送

返回课程

直播时间: 3月8日 20:00--22:00

## CAS的三个问题

1. 循环+CAS, 自旋的实现让所有线程都处于高频运行, 争抢CPU执行时间的状态。如果操作长时间不成功, 会带来很大的CPU资源消耗。

2. 仅针对单个变量的操作, 不能用于多个变量来实现原子操作。

3. ABA问题。

公告 暂无公告

53人正在观看

krto6876 20:34

2

ykt1488060783487 20:34

2

曹瑞浩ykt20017029704... 20:34

2

周国文 20:34

2

NullShow database 20:34

2

范树树 20:34

2

曹瑞浩ykt1079089124108... 20:35

引用

随风飘散ykt45048627997... 20:35

引用的地址

我要发言

发送

返回课程

直播时间: 3月8日 20:00--22:00

## ABA问题

thread 1

thread 2

堆

read i

read i

i=0

V=0

V=1

V=2

CAS(0, 1)

CAS(1, 0)

CAS(0, 1)

预期失败

> thread1、thread2同时读取到i=0后,

> thread1、thread2都要执行CAS(0, 1)操作,

> 假设thread2操作稍之后与thread1, 则thread1执行成功

> thread1紧接着执行了CAS(1, 0), 将i的值改回0

公告 暂无公告

53人正在观看

地址变了 20:45

版本

大鱼大鱼你快飞 20:45

应该要连版本一起比较

cosongja 20:45

1

Xiaol1111 20:45

0

ykt3309215743746154 20:45

1

范树树 20:45

1

曹瑞浩ykt1079089124108... 20:45

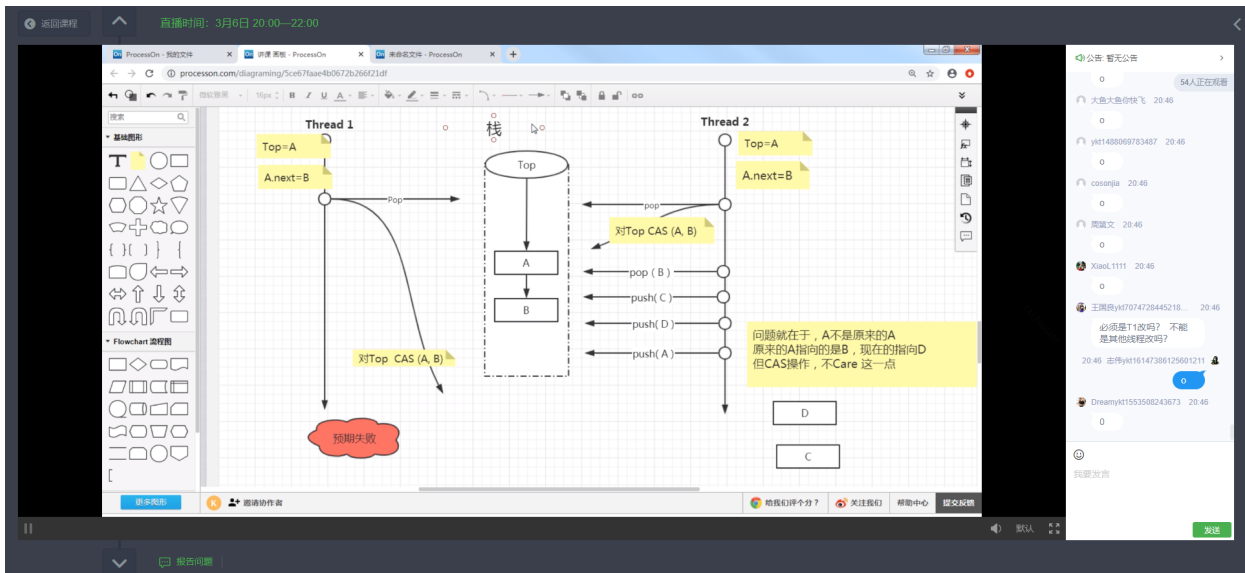
会造成什么影响呢

曹瑞浩 20:45

0

我要发言

发送



## Synchronized使用到原理

### JAVA中锁的概念

**自旋锁：**是指当一个线程在获取锁的时候，如果锁已经被其它线程获取，那么该线程将循环等待，然后不断的判断锁是否能够被成功获取，直到获取到锁才会退出循环。

**乐观锁：**假定没有冲突，在修改数据时如果发现数据和之前获取的不一致，则读最新数据，修改后重试修改

**悲观锁：**假定会发生并发冲突，同步所有对数据的相关操作，从读数据就开始上锁。

```

synchronized(this) {
    { read i .
      +1
    }
}

```

### 同步关键字synchronized

- 1、用于实例方法、静态方法时，隐式指定锁对象
- 2、用于代码块时，显示指定锁对象
- 3、锁的作用域：对象锁、类锁、分布式锁
- 4、引申：如果是多个进程，怎么办？

**特性：**可重入、独占、悲观锁

**锁优化：**锁消除（开启锁消除的参数：-XX:+DoEscapeAnalysis -XX:-EliminateLocks）

思考一下??

## 锁住 this 对象

```
synchronized(this) {
    i++;
}
```

## 加锁的状态如何记录？

公告: 暂无公告

1 52人正在观看

kelvin105 21:40

1

随风飘散ykt450486827997... 21:40

1

yk16938920899545409 21:40

1

yk1488069783487 21:40

1

遥遥海 21:41

加了volatile 也能被粗化吗

遼遼海 21:41

不是不能被重排吗

yk3309215743746154 21:43

有钛细化吗

● ၆:၂၁ ၂၁:၄၃



我要发言

发送