

线程安全的C

2017年01月17日 16:07:19

版权声明：本文为博主原创文章

证明CopyOnWriteArray

先写一段代码证明CopyO

ReadThread.java

```
1 import java.util.List;
2
3 public class ReadThread implements Runnable {
4     private List<Integer> list;
5
6     public ReadThread(List<Integer> list) {
7         this.list = list;
8     }
9
10    @Override
11    public void run() {
12        for (Integer ele : list) {
13            System.out.println("ReadThread:"+ele);
14        }
15    }
16 }
```

WriteThread.java



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 网站地图

©2018 CSDN版权所有 京ICP证09002463号

🐾 百度提供搜索支持

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

in1/article/details/54581787

```

1  import java.util.
2
3  public class Wr
4      private Lis
5
6      public Writ
7          this.li
8  }
9
10 @Override
11 public void
12     this.li
13 }
14 }

```

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](#) [网站地图](#)

©2018 CSDN版权所有 京ICP证09002463号

🐶 百度提供搜索支持

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

18

6

TestCopyOnWriteArrayList.java

```

1  import java.util.Arrays;
2  import java.util.List;
3  import java.util.concurrent.CopyOnWriteArrayList;
4  import java.util.concurrent.ExecutorService;
5  import java.util.concurrent.Executors;
6
7  public class TestCopyOnWriteArrayList {
8
9      private void test() {
10         //1、初始化CopyOnWriteArrayList
11         List<Integer> tempList = Arrays.asList(new Integer [] {1,2});
12         CopyOnWriteArrayList<Integer> copyList = new CopyOnWriteArrayList<>(tempList);
13
14
15         //2、模拟多线程对list进行读和写
16         ExecutorService executorService = Executors.newFixedThreadPool(10);
17         executorService.execute(new ReadThread(copyList));
18         executorService.execute(new WriteThread(copyList));
19         executorService.execute(new WriteThread(copyList));

```

```

20      executio
21      executio
22      executio
23      executio
24      executio
25
26      System.
27  }
28
29
30  public stat
31      new Tes
32  }
33  }

```

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](#) [网站地图](#)

©2018 CSDN版权所有 京ICP证09002463号

🔍 百度提供搜索支持

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

```

st));
t));
st));
t));
st));

ize());

```

18

6

运行上面的代码,没有报出

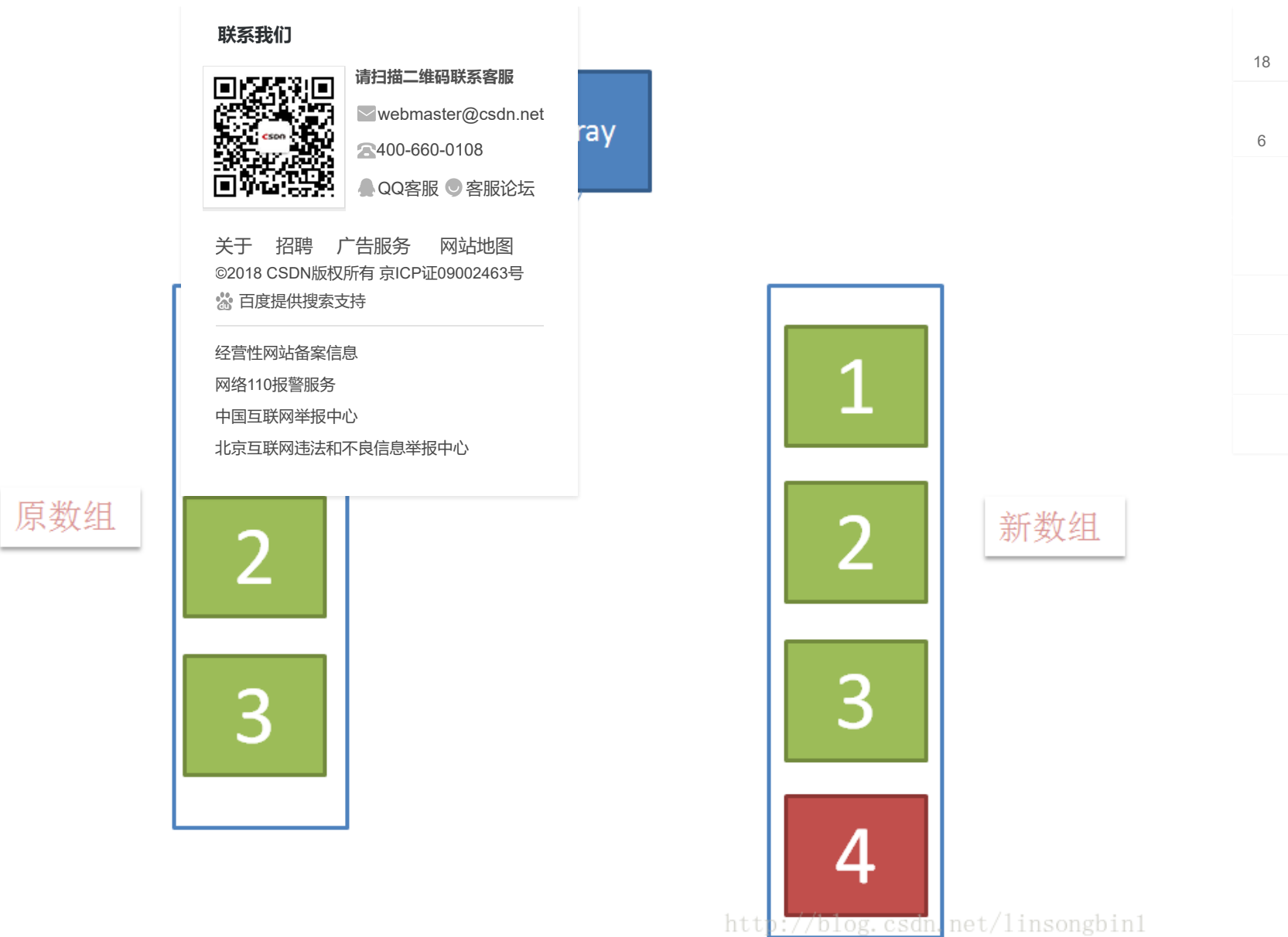
```
1 java.util.ConcurrentModificationException
```

说明了CopyOnWriteArrayList并发多线程的环境下, 仍然能很好的工作。

CopyOnWriteArrayList如何做到线程安全的

CopyOnWriteArrayList使用了一种叫**写时复制**的方法, 当有新元素添加到CopyOnWriteArrayList时, 先从原有的数组中拷贝一份出来, 然后在新的数组做写操作, 写完之后, 再将原来的数组引用指向到新数组。

当有新元素加入的时候, 如下图, 创建新数组, 并往新数组中加入一个新元素,这个时候, array这个引用仍然是指向原数组的。



<http://blog.csdn.net/linsongbin1>

当元素在新数组添加成功后，将array这个引用指向新数组。



<http://blog.csdn.net/linsongbin1>

CopyOnWriteArrayList的
这样做是为了避免在多线程

CopyOnWriteArrayList的

```
1 public boolean  
2 //1、先加锁  
3 final ReentrantLock  
4 lock.lock()  
5 try {  
6     Object[]  
7     int len  
8     //2、拷  
9     Object[]  
10    //3、将  
11    newElement  
12    //4、将array中的旧数据  
13    setArray(newElements);  
14    return true;  
15 } finally {  
16     //5、解锁  
17     lock.unlock();  
18 }  
19 }
```

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](#) [网站地图](#)

©2018 CSDN版权所有 京ICP证09002463号

🔍 百度提供搜索支持

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

把数据搞乱了，导致最终的数组数据不是我们期望的。

len + 1);

由于所有的写操作都是在新数组进行的，这个时候如果有线程并发的写，则通过锁来控制，如果有线程并发的读，则分几种情况：

- 1、如果写操作未完成，那么直接读取原数组的数据；
- 2、如果写操作完成，但是引用还未指向新数组，那么也是读取原数组数据；
- 3、如果写操作完成，并且引用已经指向了新的数组，那么直接从新数组中读取数据。

可见，CopyOnWriteArrayList的读操作是可以不用加锁的。

CopyOnWriteArrayList的使用场景

18

6

通过上面的分析，CopyOnWriteArrayList

- 1、由于写操作的时候，不能用于实时读的场是还是没法满足实时性要求
- 2、不能用于实时读的场是还是没法满足实时性要求

CopyOnWriteArrayList 合因为谁也没法保证CopyOnWriteArrayList中，这种操作分分钟引起

CopyOnWriteArrayList

如上面的分析CopyOnWriteArrayList

- 1、读写分离，读和写分开
- 2、最终一致性
- 3、使用另外开辟空间的思路，来解决并发冲突

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 网站地图
©2018 CSDN版权所有 京ICP证09002463号
🔍 百度提供搜索支持

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

内容比较多的情况下，可能导致 young gc 或者 full gc
所以调用一个 set 操作后，读取到数据可能还是旧的,虽然 CopyOnWriteArrayList 能做到最终一致性,但
数据稍微有点多，每次add/set都要重新复制数组，这个代价实在太高
在高性能的互联网应用

18

6

参考的文章

- [JAVA中的COPYONWRITE容器](#)

想对作者说点什么

chunqiuwei: 感谢分享，get到了 (06-28 09:42 #3楼) [查看回复\(1\)](#)

Melody712: 条理清晰，谢谢分享 (04-11 19:50 #2楼) [查看回复\(1\)](#)

小乌贼007: 读写分离，可以减少读和写的并发冲突，感觉这是它存在的理由。 (06-26 19:59 #1楼) [查看回复\(1\)](#)

<div><div>CopyOnWriteArrayL</div><div>CopyOnWriteArrayList:CopyOnWriteArrayList</div><div>并发容器之CopyOnW</div><div>一、 CopyOnWriteArrayLis</div><div>并发编程6: CopyOn</div><div>首先提个问题: - 线程安全</div><div>Java 中 Vector 、 St</div><div>Vector、Stack、CopyOnW</div><div>深入Java集合系列之:</div><div>CopyOnWriteArrayList简介CopyOnWriteArrayList容器是Collections.synchronizedList(List list)的替代方案, CopyOnWrit...</div><div>Java中 CopyOnWriteArrayList 的使用</div><div>java中, List在遍历的时候, 如果被修改了会抛出java.util.ConcurrentModificationException错误。 看如下代码: import java.util.Array...</div><div>CopyOnWriteArrayList源码解析——JDK1.8</div><div>参考: http://www.cnblogs.com/skywang12345/p/3498483.html1、 CopyOnWriteArrayList介绍它相当于线程安全的*ArrayList。和Ar...</div><div>Java并发编程: 并发容器之CopyOnWriteArrayList (转载)</div><div>跨进程编程时, 服务端可能会接受多个客户端的请求进行并发性操作, Aidl文件只支持List中的ArrayList,但在服务端使用CopyOnWriteA...</div><div>Vector与ArrayList与CopyOnWriteArrayList区别</div><div>1. Vector & amp; ArrayList 1) Vector的方法都是同步的(Synchronized),是线程安全的(thread-safe), 而ArrayList的方法不是, 由...</div><div>CopyOnWriteArrayList详解</div><div>CopyOnWriteArrayList详解 1. CopyOnWriteArrayList (写数组的拷贝) 是ArrayList的一个线程安全的变体, CopyOnWriteArrayList和...</div></div>	<div><div>联系我们</div><div><div></div><div>请扫描二维码联系客服</div><div>✉ webmaster@csdn.net</div><div>☎ 400-660-0108</div><div>🗣 QQ客服 🗣 客服论坛</div></div><div><div>关于 招聘 广告服务 网站地图</div><div>©2018 CSDN版权所有 京ICP证09002463号</div><div>🔍 百度提供搜索支持</div></div><div><div>经营性网站备案信息</div><div>网络110报警服务</div><div>中国互联网举报中心</div><div>北京互联网违法和不良信息举报中心</div></div></div>	<div><div> 👁 1万</div><div>的变体, 其原理大概可以通俗的理解为:初始化的时候只有...</div><div> 👁 203</div><div>序设计中的优化策略。其基本思路是, 从一开始大家都在...</div><div> 👁 4655</div><div>特点以及使用场景? 如果这个问题你答不上来, 那这篇文...</div><div><div>见解析</div><div> 👁 800</div></div><div> 👁 4317</div><div> 👁 2.2万</div><div> 👁 356</div><div> 👁 1071</div><div> 👁 153</div><div> 👁 1.3万</div></div>	<div>18</div> <div>6</div> <div></div> <div></div> <div></div> <div></div> <div></div>
--	--	---	--

相关热词for线程安全

个人资料

Sam哥哥

博客专家

原创163

粉丝649

喜欢553

等级：博客6

积分：6951

勋章：

访问：

排名：4737

归档

2018年8月2篇

2018年6月7篇

2018年5月8篇

2018年4月6篇

2018年3月13篇

展开

博主专栏

DOJO ClassD

阅读量：36095篇

JVM

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

关于招聘广告服务网站地图

©2018 CSDN版权所有 京ICP证09002463号

百度提供搜索支持

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

程安全

18

6



阅读量：21176



Eclipse

阅读量：38998

JAVA_JDK_API_

阅读量：16995



nettv

[展开](#)

最新文章

Spring Cloud工程模块划分
重构购物车的过程

Python基础学习-异常的检测和处理

Python基础学习-文件操作

Python基础学习-字典以及字典推导

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](#) [网站地图](#)

©2018 CSDN版权所有 京ICP证09002463号

 百度提供搜索支持

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

18
6