

# 并发编程网 - ifeve.com

让天下没有难学的技术

SEARCH



MAR 06 2013

101,083 人阅读

Simon-SZ

并发译文

★★★★★  
(46 votes, average: 4.63 out of 5)

18 comments

## Java并发性和多线程介绍

作者：Jakob Jenkov 译者：[Simon-SZ](#) 校对：方腾飞

<http://tutorials.jenkov.com/java-concurrency/index.html>

在过去单CPU时代，单任务在一个时间点只能执行单一程序。之后发展到多任务阶段，计算机能在同一时间点并行执行多任务或多进程。虽然并不是真正意义上的“同一时间点”，而是多个任务或进程共享一个CPU，并交由操作系统来完成多任务间对CPU的运行切换，以使得每个任务都有机会获得一定的时间片运行。

随着多任务对软件开发者带来的新挑战，程序不在能假设独占所有的CPU时间、所有的内存和其他计算机资源。一个好的程序榜样是在其不再使用这些资源时对其进行释放，以使得其他程序能有机会使用这些资源。

再后来发展到多线程技术，使得在一个程序内部能拥有多个线程并行执行。一个线程的执行可以被认为是一个CPU在执行该程序。当一个程序运行在多线程下，就好像有多个CPU在同时执行该程序。

多线程比多任务更加有挑战。多线程是在同一个程序内部并行执行，因此会对相同的内存空间进行并发读写操作。这可能是在单线程程序中从来不会遇到的问题。其中的一些错误也未必会在单CPU机器上出现，因为两个线程从来不会得到真正的并行执行。然而，更现代的计算机伴随着多核CPU的出现，也就意味着不同的线程能被不同的CPU核得到真正意义的并行执行。

如果一个线程在读一个内存时，另一个线程正向该内存进行写操作，那进行读操作的那个线程将获得什么结果呢？是写操作之前旧的值？还是写操作成功之后的新值？或是一半新一半旧的值？或者，如果是两个线程同时写同一个内存，在操作完成后将会是什么结果呢？是第一个线程写入的值？还是第二个线程写入的值？还是两个线程写入的一个混合值？因此如果没有合适的预防措施，任何结果都是可能的。而且这种行为的发生甚至不能预测，所以结果也是不确定性的。

## Java的多线程和并发性

Java是最先支持多线程的开发的语言之一，Java从一开始就支持了多线程能力，因此Java开发者能常遇到上面描述的问题场景。这也是我想为

### 热门文章

[Google Guava官方教程（中文版）](#) 581,153 人阅读

[Java NIO系列教程（一）Java NIO 概述](#) 403,315 人阅读

[Java并发性和多线程介绍目录](#) 281,404 人阅读

[Java NIO 系列教程](#) 267,320 人阅读

[Java NIO系列教程（十二）Java NIO与IO](#) 226,295 人阅读

[Java8初体验（二）Stream语法详解](#) 207,621 人阅读

[Java NIO系列教程（六）Selector](#) 201,200 人阅读

[Java NIO系列教程（三）Buffer](#) 197,246 人阅读

[Java NIO系列教程（二）Channel](#) 195,425 人阅读

[《Storm入门》中文版](#) 177,558 人阅读

[69道Spring面试题和答案](#) 167,383 人阅读

[Netty 5用户指南](#) 161,711 人阅读

[面试题](#) 145,544 人阅读

[并发框架Disruptor译文](#) 144,531 人阅读

[Java 7 并发编程指南中文版](#) 138,651 人阅读

[Java NIO系列教程（八）SocketChannel](#) 128,285 人阅读

[\[Google Guava\] 3-缓存](#) 122,685 人阅读

[\[Google Guava\] 2.3-强大的集合工具类：ja...](#) 121,994 人阅读

[\[Google Guava\] 1.1-使用和避免null](#) 112,341 人阅读

[Java NIO系列教程（七）FileChannel](#) 110,873 人阅读

Java并发技术而写这篇系列的原因。作为对自己的笔记，和对其他Java开发的追随者都可获益的。

该系列主要关注Java多线程，但有些在多线程中出现的问题会和多任务以及分布式系统中出现的存在类似，因此该系列会将多任务和分布式系统方面作为参考，所以叫法上称为“并发性”，而不是“多线程”。

原创文章，转载请注明： 转载自[并发编程网 – ifeve.com](#) 本文链接地址: [Java并发性和多线程介绍](#)

并发编程网


让天下没有难学的技术



长按，识别二维码，加关注

微信号: ifeves

AboutLatest Posts



Simon-SZ

@Tencent Cloud @UTStarcom IPTV. Interested in Cloud computing, IPTV, Programming.

★[添加本文到我的收藏](#)

Related Posts:

- [Java并发性和多线程介绍目录](#)
- [为什么开发人员从Java转到Go](#)
- [看动画学并发编程](#)
- [多线程的代价](#)
- [Cloud Card能否干掉App](#)
- [CPU缓存刷新的误解](#)
- [Java NIO系列教程（一）Java NIO 概述](#)
- [通过Axon和Disruptor处理1M tps](#)
- [基本线程同步（七）修改Lock的公平性](#)
- [线程执行者（三）创建一个大小固定的线程执行者](#)
- [Java NIO系列教程（四）Scatter/Gather](#)
- [线程池](#)
- [Java NIO系列教程（二）Channel](#)
- [Java NIO系列教程（十一）Pipe](#)
- [多线程的优点](#)

RECENT POSTS

- [Dubbo-从入门到深入](#)
- [Leader-Follower线程模型概述](#)
- [《Apache Thrift官方文档》简介](#)
- [《RabbitMQ官方指南》安装指南](#)
- [在Windows上安装RabbitMQ](#)
- [动手实现一个 LRU cache](#)
- [《Thrift官方文档》Thrift支持的语言](#)
- [《Thrift官方文档》– docker构建说明](#)
- [浅尝一致性Hash原理](#)
- [Dubbo剖析-线程模型](#)
- [分布式理论：CAP是三选二吗？](#)
- [Jarslink1.6.1版本特性](#)
- [《深入分布式缓存》之“缓存为王”](#)
- [《Thrift官方文档》翻译邀请](#)
- [《Apache RocketMQ用户指南》之定时消息示例](#)
- [使用Spring框架实现远程服务暴露与调用](#)
- [Dubbo剖析-服务消费方Invoker到客户端接口的转换](#)
- [Dubbo剖析-服务消费方远程服务到Invoker的转换](#)
- [Linux零拷贝原理](#)
- [阿里再开源！模块化开发框架JarsLink](#)
- [Dubbo剖析-服务提供方Invoker到Exporter的转换](#)
- [Dubbo剖析-服务提供方实现类到Invoker的转换](#)
- [Dubbo剖析-增强SPI中扩展点自动包装的实现](#)
- [Dubbo剖析-服务消费端异步调用](#)
- [Dubbo剖析-服务直连](#)
- [Dubbo剖析-服务分组与服务版本号](#)
- [Dubbo剖析-监控平台的搭建与使用](#)
- [Dubbo剖析-增强SPI的实现](#)
- [Dubbo剖析-整体架构分析](#)
- [《Linkerd官方文档》在ECS中运行Linkerd](#)

CATEGORIES

- [Android](#) (3)
- [C++](#) (12)
- [CPU](#) (2)
- [Framework](#) (72)
  - [akka](#) (20)
- [GO](#) (6)

Write comment

Comments RSS

Trackback are closed

Comments (18)



winner  
03/31. 2013 10:58am

[Log in to Reply.](#) | [QUOTE](#)

是否可以形象的理解为两个小孩打架的时候，互相拽着对方的头发不松手。



winner  
03/31. 2013 10:59am

[Log in to Reply.](#) | [QUOTE](#)

不好意思，前面那条评论我应该是发到死锁的那一篇文章的。



Arrow  
04/18. 2013 1:18pm

[Log in to Reply.](#) | [QUOTE](#)

从这篇开始学习多线程



[Ulric Qin](#)  
05/18. 2013 7:58pm

[Log in to Reply.](#) | [QUOTE](#)

这个系列的文章不错，真的应该好好充充电了



付政委  
07/25. 2013 4:38pm

[Log in to Reply.](#) | [QUOTE](#)

喜欢ticmy的文章，喜欢他不博客。呵呵 我都快算是个粉丝了



方 腾飞  
07/25. 2013 11:18pm

[Log in to Reply.](#) | [QUOTE](#)

喜欢就好，那我们的推荐才有价值。



伽达默尔  
08/07. 2013 9:56am

[Log in to Reply.](#) | [QUOTE](#)

博主，我想请教一个问题。java并发编程实践3.5.1之前有一个不安全发布的例子，某个类有一个属性非volatile的Holder类型holder，initialize方法中就一行代码holder = new Holder(42)。书上说这是一个不安全的发布。我不太明白，可否解释一下。是不是因为这个赋值的结果其他线程可能看不到？



方 腾飞  
08/07. 2013 11:26am

[Log in to Reply.](#) | [QUOTE](#)

能把完整的代码贴出来吗？截图也行



伽达默尔  
08/08. 2013 8:54am

[Log in to Reply.](#) | [QUOTE](#)

```
class ClassA{  
  
    //不安全发布  
  
    public Holder holder;  
  
  
    public void initialize(){  
  
        holder = new Holer(42);  
  
    }  
  
}
```

代码是这样的，原文中没有写ClassA，只有holder和initialize。文中说这是一个不安全的发布。“由于可见性的问

[groovy](#) (6)

[guava](#) (23)

[JAVA](#) (824)

[JVM](#) (40)

[linux](#) (9)

[microservices](#) (1)

[Netty](#) (31)

[react](#) (6)

[redis](#) (23)

[Scala](#) (11)

[spark](#) (19)

[Spring](#) (23)

[storm](#) (44)

[thinking](#) (3)

[Velocity](#) (10)

[Web](#) (18)

[zookeeper](#) (1)

[公告](#) (5)

[大数据](#) (33)

[好文推荐](#) (31)

[并发书籍](#) (97)

[并发译文](#) (410)

[感悟](#) (3)

[技术问答](#) (12)

[敏捷管理](#) (6)

[本站原创](#) (87)

[架构](#) (32)

[活动](#) (6)

[网络](#) (7)

TAGS

[actor](#) [Basic](#) [classes](#) [collections](#)

[concurrency](#) [Concurrent](#) [concurrent](#)

[data](#) [structure](#) [Customizing](#) [Executor](#)

[Executor](#) [framework](#) [False](#) [Sharing](#) [faq](#) [fork](#)

[Fork/Join](#) [fork](#) [join](#) [Framework](#) [Functional](#)

[Programming](#) [Guava](#) [IO](#) [JAVA](#) [java8](#)

[jmm](#) [join](#) [JVM](#) [lock](#) [Memory](#) [Barriers](#) [Netty](#)

[NIO](#) [OAuth](#) [2.0](#) [pattern-matching](#) [RingBuffer](#) [Scala](#)

[service](#) [mesh](#) [slf4j](#) [spark](#) [spark](#) [官方文档](#) [stm](#)

[Storm](#) [synchronization](#) [Synchronized](#)

[thread](#) [tomcat](#) [volatile](#) [多线程](#) [并发译](#)

[文](#) [Java](#) [Maven](#)

题，容器还是会在其他线程中被设置为一个不一致的状态，即使它的不变约束已经在构造函数中得以正确的创建。这种不正确的发布导致其他线程可以观察到'局部创建对象 ( partially constructed object ) ' 。”



方 腾飞      [Log in to Reply.](#) | [QUOTE](#)  
08/09. 2013 3:45am

我是这么理解不安全发布的，是指在并发场景下不一定能拿到自己想要的结果。

比如A线程执行initialize方法holder = new  
Holer(42);，如果B线程也在执行holder=new  
Holer(20);  
那holder到底等于什么，这个结果就不一定了。

PS：原文翻译得羞涩难懂。



伽达默尔      [QUOTE](#)  
08/09. 2013 12:29pm

你说的对。还有，翻译确实有点晦涩。多谢解答。



hengzheCP      [Log in to Reply.](#) | [QUOTE](#)  
11/23. 2014 12:28pm

像这种field未使用final、或者volatile修饰的，其它线程访问它的时候，由于重排序，可能导致其它线程在访问该变量的时候，它的值还没有被赋上，因此该线程读到了一个“错误”的变量。以下面的代码为例，holder = new Holer(42);其实是两个操作：

1、初始化Holder实例 2、将Holder实例赋值给引用变量holder。由于重排序，代码的实际执行顺序可能是先执行2,再执行1。此时，其它线程在访问holder的时候（由于它没有采用volatile、也没有用final修饰）而未加锁的话，就可能访问到一个未初始化完毕的Holder实例（ 仅仅是指向一块内存地址的引用而已，并无实质性对象 ）。

```
class ClassA{  
  
//不安全发布  
  
public Holder holder;  
  
public void initialize(){  
  
holder = new Holer(42);  
  
}  
  
}
```



it\_power      [Log in to Reply.](#) | [QUOTE](#)  
04/13. 2015 5:41pm

这个不安全引用是因为，holder 是public 的，如果一个线程A，在初始化的时候，线程B获取hodler的值，就可能造成结果不一致，所以不安全。



bad man      [Log in to Reply.](#) | [QUOTE](#)  
04/02. 2014 3:51pm

给力哦 开始理解多线程



yehong542983647  
12/02. 2015 4:00pm

[Log in to Reply](#) | [QUOTE](#)

看完一篇，就需要顶一下，谢谢你们的劳动



丁国航  
03/10. 2016 3:12pm

[Log in to Reply](#) | [QUOTE](#)

路过点赞



波波  
04/07. 2016 9:42am

[Log in to Reply](#) | [QUOTE](#)

最近准备研究多线程呢，由于英文有限看java并发编程实战的译文有点难懂。  
手里正好买了一本方老师的。来到这个网站，希望和书一起学习，也能得到大神的指点



帝释天老邪  
08/09. 2016 10:00am

[Log in to Reply](#) | [QUOTE](#)

衷心感谢翻译人员，你们的辛勤劳动，可以使我们这些新人踩在你们的肩膀  
上更好的前进

You must be [logged in](#) to post a comment.

死锁

Java中的读/写锁