Ｊａｖａ并发编程深度解析与实战

Ｊａｖａ并发编程深度解析与实战书籍信息

书名：Ｊａｖａ并发编程深度解析与实战

ＩＳＢＮ：９７８７１２１４２１３６５

作者：谭锋&ｎｂｓｐ；

出版社：电子工业出版社

出版时间：２０２１－１０

页数：５０８

价格：１１８

纸张：

装帧：

开本：

语言：未知

丛书：咕泡学院Ｊａｖａ架构师成长丛书

ＴＡＧ：

豆瓣评分：

信息来源：Ｊａｖａ并发编程深度解析与实战　电子书网盘下载　２０２４　ｐｄｆ　ｍｏｂ

Ｊａｖａ并发编程深度解析与实战

Ｊａｖａ并发编程深度解析与实战书籍简介

本书涵盖Ｊａｖａ并发编程体系的核心库和核心类的使用及原理分析，具体包括线程、ｓｙｎ
重入锁和读写锁、并发中的条件等待机制、Ｊ．Ｕ．Ｃ并发工具集、深度探索并发编程不得不
工具、阻塞队列、并发安全集合、线程池、异步编程特性等。书中针对每一个技术点，
纵向分析与其相关的所有内容，并且对相关知识点进行了非常详细的说明，同时从架构
实践的角度来看待并发，通过大量实战案例让读者理解各类技术在实际应用中的使用方
法。

作者花费了４年时间，投入了大量精力对并发编程领域进行了深入的研究，将自己１３年的Ｊ
发及架构经验融入了书中，对各位读者来说，这应该是一本非常值得阅读的图书。

Ｊａｖａ并发编程深度解析与实战目录

第１章　Ｊａｖａ线程的实践及原理揭秘
1
１．１　如何理解系统并发
1
１．２　系统如何支撑高并发
2
１．３　线程的前世今生
3
１．３．１　大白话理解进程和线程
3
１．３．２　线程的核心价值
5
１．３．３　如何理解并发和并行
6
１．４　在Ｊａｖａ中如何使用多线程
6
１．４．１　实现Ｒｕｎｎａｂｌｅ接口创建线程
6
１．４．２　继承Ｔｈｒｅａｄ类创建线程
7
１．４．３　实现Ｃａｌｌａｂｌｅ接口并创建带返回值的线程
7
１．５　多线程如何应用到实际场景
8

1.5.1 ServerSocket
9
1.5.2 SocketThread
10
1.6 多线程的基本原理
11
1.7 线程的运行状态
11
1.7.1 线程运行状态演示
12
1.7.2 线程运行状态流转图
14
1.8 如何正确终止线程
15
1.8.1 关于安全中断线程的思考
17
1.8.2 安全中断线程之interrupt
17
1.8.3 如何中断处于阻塞状态下的线程
18
1.8.4 interrupt()方法的实现原理
21
1.9 理解上下文切换带来的性能影响
24
1.9.1 上下文切换带来的问题
25

1．9．2　什么是上下文切换
27
1．9．3　如何减少上下文切换
29
1．10　揭秘守护线程
30
1．10．1　守护线程的应用场景
32
1．10．2　守护线程使用注意事项
32
1．11　快速定位并解决线程导致的生产问题
33
1．11．1　死锁导致请求无法响应
35
1．11．2　CPU占用率很高，响应很慢
36
1．12　本章小结
38
第2章　深度揭秘synchronized实现原理
39
2．1　揭秘多线程环境下的原子性问题
40
2．1．1　深入分析原子性问题的本质
41
2．1．2　关于原子性问题的解决办法
43

2．2　Ｊａｖａ中的ｓｙｎｃｈｒｏｎｉｚｅｄ同步锁
44
2．2．1　ｓｙｎｃｈｒｏｎｉｚｅｄ的使用方法
44
2．2．2　了解ｓｙｎｃｈｒｏｎｉｚｅｄ同步锁的作用范围
45
2．3　关于ｓｙｎｃｈｒｏｎｉｚｅｄ同步锁的思考
49
2．4　ｓｙｎｃｈｒｏｎｚｉｅｄ同步锁标记存储分析
49
2．4．1　揭秘Ｍａｒｋ　Ｗｏｒｄ的存储结构
50
2．4．2　图解分析对象的实际存储
52
2．4．3　通过ＣｌａｓｓＬａｙｏｕｔ查看对象内存布局
53
2．4．4　Ｈｏｔｓｐｏｔ虚拟机中对象存储的源码
57
2．5　ｓｙｎｃｈｒｏｎｉｚｅｄ的锁类型
59
2．5．1　偏向锁的原理分析
60
2．5．2　轻量级锁的原理分析
64
2．5．3　重量级锁的原理分析
65

2．6　关于ＣＡＳ机制的实现原理分析
68
2．6．1　ＣＡＳ在Ａｔｏｍｉｃ Ｉｎｔｅｇｅｒ中的应用
70
2．6．2　ＣＡＳ实现自旋锁
72
2．6．3　ＣＡＳ在ＪＶＭ中的实现原理分析
73
2．7　锁升级的实现流程
76
2．7．1　偏向锁的实现原理
77
2．7．2　轻量级锁的实现原理
82
2．7．3　重量级锁的实现原理
86
2．8　ｓｙｎｃｈｒｏｎｉｚｅｄ使用不当带来的死锁问题
89
2．8．1　死锁的案例分析
90
2．8．2　死锁产生的必要条件
92
2．8．3　如何解决死锁问题
92
2．9　本章小结
96

第3章　ｖｏｌａｔｉｌｅ为什么能解决可见性和有序性问题
９７
３．１　关于线程的可见性问题分析
９７
３．１．１　思考导致问题的原因
９８
３．１．２　ｖｏｌａｔｉｌｅ关键字解决可见性问题
９９
３．２　深度理解可见性问题的本质
１００
３．２．１　如何最大化提升ＣＰＵ利用率
１００
３．２．２　详述ＣＰＵ高速缓存
１０１
３．２．３　ＣＰＵ缓存一致性问题
１０７
３．２．４　总结可见性问题的本质
１１１
３．３　ｖｏｌａｔｉｌｅ如何解决可见性问题
１１２
３．４　指令重排序导致的可见性问题
１１３
３．４．１　什么是指令重排序
１１４
３．４．２　ａｓ－ｉｆ－ｓｅｒｉａｌ语义
１１６

3．5　从CPU层面深度剖析指令重排序的本质
117
3．5．1　CPU优化—Store　Buffers
117
3．5．2　CPU优化—Store　Forwarding
119
3．5．3　CPU优化—Invalidate　Queues
122
3．6　通过内存屏障解决内存系统重排序问题
125
3．6．1　内存屏障详解
125
3．6．2　通过内存屏障防止重排序
127
3．6．3　不同CPU的重排序规则
128
3．6．4　总结CPU层面的可见性问题
129
3．7　Java　Memory　Mode
129
3．7．1　从JVM和硬件层面理解Java　Memory　Mode
130
3．7．2　JVM提供的内存屏障指令
133
3．8　揭秘volatile实现原理
136

3．9　Happens-Before模型
138
3．9．1　程序顺序规则
138
3．9．2　传递性规则
139
3．9．3　volatile变量规则
139
3．9．4　监视器锁规则
140
3．9．5　start规则
141
3．9．6　join规则
141
3．10　本章小结
142
第4章　深入浅出分析J．U．C中的重入锁和读写锁
143
4．1　J．U．C中与锁相关的API
143
4．1．1　ReentrantLock的基本应用
144
4．1．2　ReentrantReadWriteLock的基本应用
145
4．1．3　StampedLock的基本应用
147

4.2 ReentrantLock的设计猜想
149
4.2.1 锁的互斥，必须要竞争同一个共享变量
150
4.2.2 没有竞争到锁的线程，需要阻塞
151
4.2.3 需要一个容器存储被阻塞的线程
151
4.3 ReentrantLock实现原理分析
151
4.4 AbstractQueuedSynchronizer
152
4.5 ReentrantLock源码分析
154
4.5.1 ReentrantLock.lock()方法
154
4.5.2 AbstractQueuedSynchronizer.acquire(
156
4.5.3 NonfairSync.tryAcquire()方法
156
4.5.4 ReentrantLock.nofairTryAcquire()方法
157
4.5.5 AbstractQueuedSynchronizer.addWaite
158
4.5.6 AQS.acquireQueued()方法
159

4.6 ReentrantLock释放锁源码分析
162
4.6.1 ReentrantLock.tryRelease()方法
163
4.6.2 unparkSuccessor()方法
163
4.6.3 释放锁的线程继续执行
164
4.7 分析ReentrantReadWriteLock类的原理
166
4.7.1 WriteLock锁竞争原理
167
4.7.2 ReadLock锁竞争原理
170
4.7.3 ReentrantReadWriteLock中的锁降级
177
4.8 StampedLock的原理分析
179
4.8.1 核心内部类分析
180
4.8.2 StampedLock原理图解
182
4.8.3 StampedLock锁升级
184
4.9 本章小结
187

第5章　从线程通信来窥探并发中的条件等待机制
188
5.1　wait/notify
189
5.1.1　wait()/notify()方法使用实战
189
5.1.2　图解生产者/消费者
192
5.1.3　wait()/notify()方法的原理
193
5.1.4　wait()/notify()方法为什么要加同步锁
195
5.2　通过Thread.join获取线程执行结果
195
5.2.1　Thread.join()方法的执行流程
196
5.2.2　Thread.join()方法的实现原理
196
5.3　J.U.C中的条件控制Condition
198
5.3.1　Condition的基本应用
199
5.3.2　基于Condition的手写阻塞队列
201
5.4　Condition的设计猜想
203

5.5 Condition的源码分析
203
5.5.1 Condition.await()方法
204
5.5.2 Condition.signal()方法
208
5.5.3 锁竞争成功后的执行流程
210
5.6 本章小结
213
第6章 J.U.C并发工具集实战及原理分析
214
6.1 CountDownLatch简单介绍
214
6.1.1 CountDownLatch的基本使用
215
6.1.2 CountDownLatch运行流程
216
6.1.3 如何落地到实际应用
216
6.1.4 CountDownLatch的其他用法
220
6.2 CountDownLatch底层原理
221
6.2.1 让线程等待的await()方法到底做了什么
222

6．2．2　深入分析ｃｏｕｎｔＤｏｗｎ（）方法源码
224
6．2．3　线程被唤醒后的执行逻辑
228
6．3　Ｓｅｍａｐｈｏｒｅ
230
6．3．1　Ｓｅｍａｐｈｏｒｅ使用案例
231
6．3．2　Ｓｅｍａｐｈｏｒｅ方法及场景说明
232
6．4　Ｓｅｍａｐｈｏｒｅ原理分析
233
6．4．1　Ｓｅｍａｐｈｏｒｅ令牌获取过程分析
233
6．4．2　Ｓｅｍａｐｈｏｒｅ令牌释放过程分析
236
6．5　ＣｙｃｌｉｃＢａｒｒｉｅｒ
237
6．5．1　ＣｙｃｌｉｃＢａｒｒｉｅｒ的基本使用
237
6．5．2　基本原理分析
239
6．6　ＣｙｃｌｉｃＢａｒｒｉｅｒ实现原理及源码
239
6．6．1　ａｗａｉｔ（）方法
241

6.6.2 reset()方法
244
6.7 本章小结
244
第7章 深度探索并发编程不得不知的工具
245
7.1 初步认识ThreadLocal
245
7.2 ThreadLocal的应用场景分析
247
7.3 ThreadLocal解决SimpleDateFormat线程安全问题
249
7.3.1 SimpleDateFormat线程安全问题的原理
250
7.3.2 ThreadLocal实现线程安全性
253
7.4 ThreadLocal实现原理分析
254
7.4.1 set()方法源码分析
255
7.4.2 get()方法源码分析
265
7.4.3 ThreadLocal内存泄漏
266
7.5 任务拆分与聚合Fork/Join
269

7.5.1 Fork/Join的核心API说明
269
7.5.2 Fork/Join的基本使用
270
7.6 Fork/Join的实现原理
272
7.6.1 WorkQueue的原理
274
7.6.2 工作窃取算法
275
7.7 Fork/Join的核心源码分析
275
7.7.1 任务提交过程详解
276
7.7.2 唤醒或者创建工作线程
281
7.7.3 工作线程和工作队列的绑定
283
7.7.4 ForkJoinWorkerThread运行过程
285
7.8 使用Fork/Join解决实际问题
286
7.8.1 项目结构说明
286
7.8.2 ILoadDataProcessor
287

7.8.3 AbstractLoadDataProcessor
288
7.8.4 业务服务类
288
7.8.5 Item聚合任务服务
289
7.8.6 ComplexTradeTaskService
291
7.8.7 测试代码
292
7.9 本章小结
293
第8章 深度剖析阻塞队列的设计原理及实现
294
8.1 什么是阻塞队列
294
8.2 Java中提供的阻塞队列
295
8.3 阻塞队列中提供的方法
296
8.4 阻塞队列的使用
297
8.4.1 生产者/消费者模型代码
297
8.4.2 图解阻塞队列实现原理
299

8．5　阻塞队列应用实战
299
8．5．1　基于阻塞队列的责任链源码
300
8．5．2　阻塞队列实战场景总结
304
8．6　详解J．U．C中阻塞队列的使用
305
8．6．1　基于数组结构的阻塞队列ArrayBlockingQueue
305
8．6．2　基于链表的阻塞队列LinkedBlockingQueue
306
8．6．3　优先级阻塞队列PriorityBlockingQueue
308
8．6．4　延迟阻塞队列DelayQueue
310
8．6．5　无存储结构的阻塞队列SynchronousQueue
314
8．6．6　阻塞队列结合体LinkedTransferQueue
318
8．6．7　双向阻塞队列LinkedBlockingDeque
319
8．7　阻塞队列的实现原理
321
8．7．1　put()方法说明
321

8．7．2　take()方法说明
324
8．8　本章小结
326
第9章　深度解读并发安全集合的原理及源码
327
9．1　并发安全集合ConcurrentHashMap
327
9．2　正确理解ConcurrentHashMap的线程安全性
328
9．2．1　computeIfAbsent()方法详解
330
9．2．2　computeIfPresent()方法详解
331
9．2．3　compute()方法详解
331
9．2．4　merge()方法详解
332
9．3　ConcurrentHashMap的数据结构
332
9．3．1　ConcurrentHashMap数据存储相关定义
333
9．3．2　Node数组初始化过程分析
335
9．3．3　单节点到链表的转化过程分析
339

9．3．4　扩容还是转化为红黑树
341
9．4　深度分析ConcurrentHashMap中的并发扩容机制
346
9．4．1　多线程并发扩容原理图解
347
9．4．2　详解ConcurrentHashMap中的数据迁移
347
9．5　分段锁设计提高统计元素数量的性能
357
9．5．1　size计数的基本原理分析
358
9．5．2　addCount()方法详解
358
9．5．3　fullAddCount()方法分析
360
9．6　详解红黑树的实现原理
365
9．6．1　什么是红黑树
365
9．6．2　红黑树的平衡规则
366
9．6．3　红黑树的平衡场景规则说明
368
9．6．4　红黑树插入元素平衡图解
369

9．6．5　红黑树规则实战解析
373
9．6．6　红黑树中删除元素的平衡规则
376
9．7　ConcurrentHashMap中红黑树的使用
381
9．7．1　TreeBin的基本介绍
383
9．7．2　链表转化成红黑树
384
9．7．3　自平衡
386
9．7．4　ConcurrentHashMap总结
388
9．8　Java中其他并发安全集合
388
9．8．1　ConcurrentLinkedQueue
388
9．8．2　ConcurrentLinkedDeque
390
9．8．3　ConcurrentSkipListMap
391
9．9　深度分析数据结构：跳表
391
9．9．1　什么是跳表
392

９．９．２　跳表的特性
３９２
９．９．３　跳表的基本操作
３９２
９．１０　本章小结
３９４
第１０章　站在架构的角度思考线程池的设计及原理
３９５
１０．１　线程池的优势
３９６
１０．２　Ｊａｖａ中提供的线程池
３９６
１０．２．１　线程池的使用
３９７
１０．２．２　ＴｈｒｅａｄＰｏｏｌＥｘｅｃｕｔｏｒ
３９８
１０．３　Ｅｘｅｃｕｔｏｒ框架详解
４０２
１０．４　线程池的设计猜想
４０４
１０．４．１　线程池的需求分析
４０４
１０．４．２　生产者／消费者模型的设计
４０５
１０．４．３　任务拒绝策略
４０６

10.4.4 非核心线程的回收
408
10.4.5 线程池设计总结
408
10.5 从实现原理了解线程池
408
10.6 线程池核心源码剖析
409
10.6.1 线程状态和数量存储
410
10.6.2 线程池的状态机及变更
412
10.6.3 从execute()方法分析线程池源码
413
10.7 合理设置线程池参数
425
10.7.1 线程池大小的合理设置
426
10.7.2 动态设置线程池参数
427
10.8 线程池的监控
431
10.8.1 线程池监控的基本原理
432
10.8.2 在Spring Boot应用中发布线程池信息
433

10.9 本章小结
442
第11章 Java并发编程中的异步编程特性
443
11.1 了解Future/Callable
443
11.2 Future/Callable的实现原理
445
11.2.1 FutureTask的核心属性
446
11.2.2 FutureTask.run()
447
11.2.3 FutureTask.get()
448
11.2.4 finishCompletion()
452
11.3 Java 8新特性之CompletableFuture
453
11.3.1 CompletableFuture类关系图
454
11.3.2 CompletableFuture方法说明
454
11.3.3 主动获取执行结果
458
11.4 CompletionStage方法及作用说明
459

11.4.1 方法分类概述
460
11.4.2 CompletionStage异常处理方法
466
11.4.3 方法类型总结
470
11.5 CompletableFuture综合实战
470
11.5.1 商品实体对象
470
11.5.2 模拟微服务请求实现类
471
11.5.3 Web请求
472
11.6 CompletableFuture实现原理分析
474
11.6.1 Completion说明
476
11.6.2 图解Completion的栈结构
477
11.7 核心源码分析
480
11.7.1 CompletableFuture静态任务创建
480
11.7.2 Completion Stack构建
482

１１．７．３　简述ＵｎｉＣｏｍｐｌｅｔｉｏｎ
４８４
１１．７．４　任务执行流程
４８６
１１．７．５　获取任务执行结果
４８７
１１．８　本章小结
４９

Ｊａｖａ并发编程深度解析与实战作者简介

谭锋（Ｍｉｃ）

咕泡学院联合创始人，２０１７年开始创业，至今已有４年多时间。拥有１３年Ｊａｖａ开发有４年授课经验，培养了３万多名学员，学员遍布一二线主流互联网企业。

曾就职于中国电信、平安支付、挖财等公司担任业务架构师，在平安支付主导了基于Ｄｕｂｂ服务化架构设计和落地，在挖财推动了基于Ｓｐｒｉｎｇ　Ｂｏｏｔ微服务化架构的改造。因、高并发架构有非常深入的研究，以及丰富的实践经验。

目前担任教学总监一职，负责微服务及高并发领域的课程研发和设计。

Ｊａｖａ并发编程深度解析与实战

Ｊａｖａ并发编程深度解析与实战其它

书籍介绍

本书涵盖Ｊａｖａ并发编程体系的核心库和核心类的使用及原理分析，具体包括线程、ｓｙｎ重入锁和读写锁、并发中的条件等待机制、Ｊ．Ｕ．Ｃ并发工具集、深度探索并发编程不得不工具、阻塞队列、并发安全集合、线程池、异步编程特性等。书中针对每一个技术点，纵向分析与其相关的所有内容，并且对相关知识点进行了非常详细的说明，同时从架构实践的角度来看待并发，通过大量实战案例让读者理解各类技术在实际应用中的使用方法。

作者花费了４年时间，投入了大量精力对并发编程领域进行了深入的研究，将自己１３年的Ｊ发及架构经验融入了书中，对各位读者来说，这应该是一本非常值得阅读的图书。

尾页

版权说明

本站所提供下载的ＰＤＦ图书仅提供预览和简介，请支持正版图书。

更多精彩内容请访问：Ｊａｖａ并发编程深度解析与实战　电子书网盘下载　２０２４　ｐｄ

ＰＤＦ　Ｊａｖａ并发编程深度解析与实战　ｐｄｆ　网盘　电子书　下载　全格式

ＥＰＵＢ　Ｊａｖａ并发编程深度解析与实战　ｅｐｕｂ　网盘　电子书　下载　全格式

ＡＺＷ３　Ｊａｖａ并发编程深度解析与实战　ａｚｗ３　网盘　电子书　下载　全格式

ＭＯＢＩ　Ｊａｖａ并发编程深度解析与实战　ｍｏｂｉ　网盘　电子书　下载　全格式

ＷＯＲＤ　Ｊａｖａ并发编程深度解析与实战　ｗｏｒｄ　网盘　电子书　下载　全格式

ＴＸＴ　Ｊａｖａ并发编程深度解析与实战　ｔｘｔ　网盘　电子书　下载　全格式