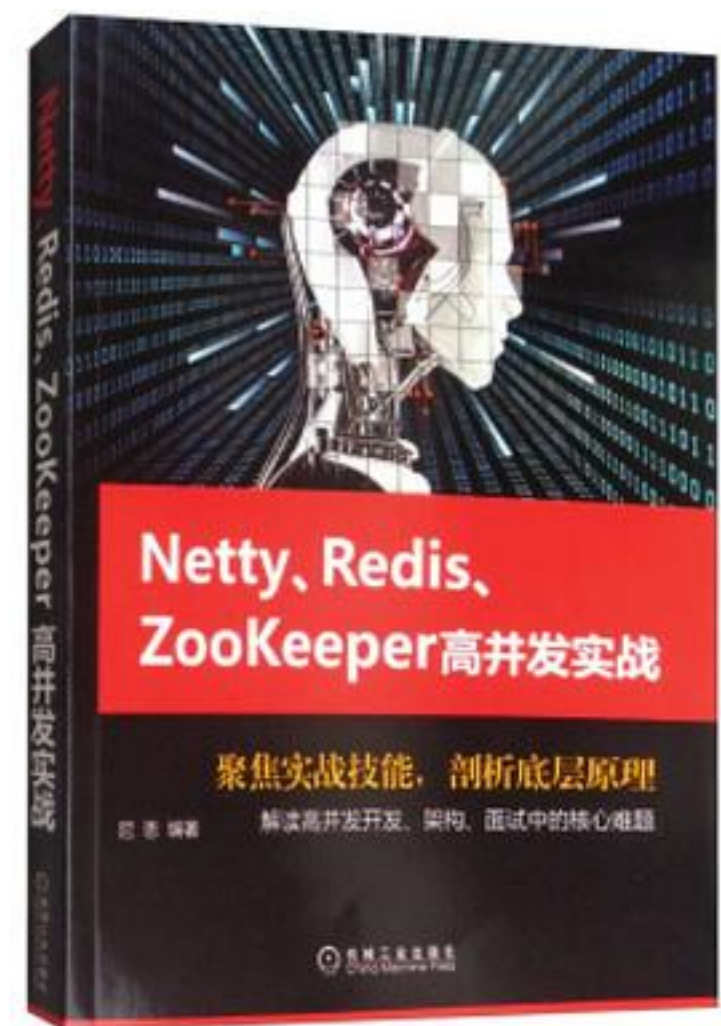


# Netty、Redis、Zookeeper高并发实战



[Netty、Redis、Zookeeper高并发实战\\_下载链接1](#)

著者:尼恩

出版者:机械工业出版社

出版时间:2019-8

装帧:平装

isbn:9787111632900

本书为了让读者扎稳高性能基础，浅显易懂地剖析高并发IO的底层原理，细致细腻地解

析Reactor高性能模式，图文并茂地介绍Java异步回调模式。掌握这些基础原理，能够帮助读者解决Java后台开发的一些实际问题。

本书共12章，主要介绍高性能通信框架Netty，并详尽介绍Netty的EventLoop、Handler、Pipeline、ByteBuf、Decoder、Encoder等重要组件，然后介绍单体IM的实战设计和模块实现。本书对ZooKeeper、Curator API、Redis、Jedis API的使用也进行详尽的介绍，让读者具备高并发、可扩展系统的设计和开发能力。

作者介绍:

目录: 前言

第1章 高并发时代的必备技能 1

1.1 Netty为何这么火 1

1.1.1 Netty火热的程度 1

1.1.2 Netty是面试的必杀器 2

1.2 高并发利器Redis 2

1.2.1 什么是Redis 2

1.2.2 Redis成为缓存事实标准的原因 3

1.3 分布式利器ZooKeeper 3

1.3.1 什么是ZooKeeper 3

1.3.2 ZooKeeper的优势 4

1.4 高并发IM的综合实践 4

1.4.1 高并发IM的学习价值 4

1.4.2 庞大的应用场景 5

1.5 Netty、Redis、ZooKeeper实践计划 5

1.5.1 第1天：Java NIO实践 5

1.5.2 第2天：Reactor反应器模式实践 6

1.5.3 第3天：异步回调模式实践 7

1.5.4 第4天：Netty基础实践 8

1.5.5 第5天：解码器（Decoder）与编码器（Encoder）实践 9

1.5.6 第6天：JSON和ProtoBuf序列化实践 11

1.5.7 第7~10天：基于Netty的单聊实战 12

1.5.8 第11天：ZooKeeper实践计划 14

1.5.9 第12天：Redis实践计划 14

1.6 本章小结 16

第2章 高并发IO的底层原理 17

2.1 IO读写的基础原理 17

2.1.1 内核缓冲区与进程缓冲区 18

2.1.2 详解典型的系统调用流程 18

2.2 四种主要的IO模型 19

2.2.1 同步阻塞IO（Blocking IO） 20

2.2.2 同步非阻塞NIO（None Blocking IO） 21

2.2.3 IO多路复用模型（IO Multiplexing） 22

2.2.4 异步IO模型（Asynchronous IO） 23

2.3 通过合理配置来支持百万级并发连接 24

2.4 本章小结 26

第3章 Java NIO通信基础详解 27

3.1 Java NIO简介 27

3.1.1 NIO和OIO的对比 28

3.1.2 通道（Channel） 28

3.1.3 Selector 选择器 28

3.1.4 缓冲区（Buffer） 29

- 3.2 详解NIO Buffer类及其属性 29
  - 3.2.1 Buffer类 29
  - 3.2.2 Buffer类的重要属性 29
  - 3.2.3 4个属性的小结 31
- 3.3 详解NIO Buffer类的重要方法 31
  - 3.3.1 allocate()创建缓冲区 31
  - 3.3.2 put()写入到缓冲区 32
  - 3.3.3 flip()翻转 33
  - 3.3.4 get()从缓冲区读取 34
  - 3.3.5 rewind()倒带 35
  - 3.3.6 mark()和reset() 37
  - 3.3.7 clear()清空缓冲区 38
  - 3.3.8 使用Buffer类的基本步骤 38
- 3.4 详解NIO Channel（通道）类 38
  - 3.4.1 Channel（通道）的主要类型 39
  - 3.4.2 FileChannel文件通道 39
  - 3.4.3 使用FileChannel完成文件复制的实践案例 41
  - 3.4.4 SocketChannel套接字通道 42
  - 3.4.5 使用SocketChannel发送文件的实践案例 44
  - 3.4.6 DatagramChannel数据报通道 46
  - 3.4.7 使用DatagramChannel数据包通道发送数据的实践案例 47
- 3.5 详解NIO Selector选择器 49
  - 3.5.1 选择器以及注册 49
  - 3.5.2 SelectableChannel可选择通道 50
  - 3.5.3 SelectionKey选择键 50
  - 3.5.4 选择器使用流程 50
  - 3.5.5 使用NIO实现Discard服务器的实践案例 52
  - 3.5.6 使用SocketChannel在服务器端接收文件的实践案例 54
- 3.6 本章小结 57
- 第4章 鼎鼎大名的Reactor反应器模式 59
  - 4.1 Reactor反应器模式为何如此重要 59
    - 4.1.1 为什么首先学习Reactor反应器模式 59
    - 4.1.2 Reactor反应器模式简介 60
    - 4.1.3 多线程OIO的致命缺陷 60
  - 4.2 单线程Reactor反应器模式 62
    - 4.2.1 什么是单线程Reactor反应器 62
    - 4.2.2 单线程Reactor反应器的参考代码 63
    - 4.2.3 一个Reactor反应器版本的EchoServer实践案例 65
    - 4.2.4 单线程Reactor反应器模式的缺点 67
  - 4.3 多线程的Reactor反应器模式 68
    - 4.3.1 多线程池Reactor反应器演进 68
    - 4.3.2 多线程Reactor反应器的实践案例 68
    - 4.3.3 多线程Handler处理器的实践案例 70
  - 4.4 Reactor反应器模式小结 72
  - 4.5 本章小结 73
- 第5章 并发基础中的Future异步回调模式 74
  - 5.1 从泡茶的案例说起 74
  - 5.2 join异步阻塞 75
    - 5.2.1 线程的join合并流程 75
    - 5.2.2 使用join实现异步泡茶喝的实践案例 75
    - 5.2.3 详解join合并方法 77
  - 5.3 FutureTask异步回调之重武器 77
    - 5.3.1 Callable接口 77
    - 5.3.2 初探FutureTask类 78

5.3.3 Future接口	79
5.3.4 再探FutureTask类	79
5.3.5 使用FutureTask类实现异步泡茶喝的实践案例	80
5.4 Guava的异步回调	82
5.4.1 详解FutureCallback	82
5.4.2 详解ListenableFuture	83
5.4.3 ListenableFuture异步任务	84
5.4.4 使用Guava实现泡茶喝的实践案例	84
5.5 Netty的异步回调模式	87
5.5.1 详解GenericFutureListener接口	87
5.5.2 详解Netty的Future接口	88
5.5.3 ChannelFuture的使用	88
5.5.4 Netty的出站和入站异步回调	89
5.6 本章小结	90
第6章 Netty原理与基础	91
6.1 第一个Netty的实践案例DiscardServer	91
6.1.1 创建第一个Netty项目	91
6.1.2 第一个Netty服务器端程序	92
6.1.3 业务处理器NettyDiscardHandler	93
6.1.4 运行NettyDiscardServer	94
6.2 解密Netty中的Reactor反应器模式	95
6.2.1 回顾Reactor反应器模式中IO事件的处理流程	95
6.2.2 Netty中的Channel通道组件	96
6.2.3 Netty中的Reactor 反应器	96
6.2.4 Netty中的Handler处理器	97
6.2.5 Netty的流水线 (Pipeline)	98
6.3 详解Bootstrap启动器类	100
6.3.1 父子通道	100
6.3.2 EventLoopGroup线程组	101
6.3.3 Bootstrap的启动流程	101
6.3.4 ChannelOption通道选项	104
6.4 详解Channel通道	105
6.4.1 Channel通道的主要成员和方法	105
6.4.2 EmbeddedChannel嵌入式通道	107
6.5 详解Handler业务处理器	108
6.5.1 ChannelInboundHandler通道入站处理器	109
6.5.2 ChannelOutboundHandler通道出站处理器	110
6.5.3 ChannelInitializer通道初始化处理器	111
6.5.4 ChannelInboundHandler的生命周期的实践案例	112
6.6 详解Pipeline流水线	115
6.6.1 Pipeline入站处理流程	115
6.6.2 Pipeline出站处理流程	116
6.6.3 ChannelHandlerContext上下文	118
6.6.4 截断流水线的处理	118
6.6.5 Handler业务处理器的热拔插	120
6.7 详解ByteBuf缓冲区	122
6.7.1 ByteBuf的优势	122
6.7.2 ByteBuf的逻辑部分	123
6.7.3 ByteBuf的重要属性	123
6.7.4 ByteBuf的三组方法	124
6.7.5 ByteBuf基本使用的实践案例	125
6.7.6 ByteBuf的引用计数	127
6.7.7 ByteBuf的Allocator分配器	128
6.7.8 ByteBuf缓冲区的类型	130

- 6.7.9 三类ByteBuf使用的实践案例 131
- 6.7.10 ByteBuf的自动释放 133
- 6.8 ByteBuf浅层复制的高级使用方式 136
  - 6.8.1 slice切片浅层复制 136
  - 6.8.2 duplicate整体浅层复制 137
  - 6.8.3 浅层复制的问题 138
- 6.9 EchoServer回显服务器的实践案例 138
  - 6.9.1 NettyEchoServer回显服务器的服务器端 138
  - 6.9.2 共享NettyEchoServerHandler处理器 139
  - 6.9.3 NettyEchoClient客户端代码 140
  - 6.9.4 NettyEchoClientHandler处理器 142
- 6.10 本章小结 143
- 第7章 Decoder与Encoder重要组件 144
  - 7.1 Decoder原理与实践 144
    - 7.1.1 ByteToMessageDecoder解码器 145
    - 7.1.2 自定义Byte2IntegerDecoder整数解码器的实践案例 146
    - 7.1.3 ReplayingDecoder解码器 148
    - 7.1.4 整数的分包解码器的实践案例 149
    - 7.1.5 字符串的分包解码器的实践案例 152
    - 7.1.6 MessageToMessageDecoder解码器 156
  - 7.2 开箱即用的Netty内置Decoder 157
    - 7.2.1 LineBasedFrameDecoder解码器 157
    - 7.2.2 DelimiterBasedFrameDecoder解码器 158
    - 7.2.3 LengthFieldBasedFrameDecoder解码器 159
    - 7.2.4 多字段Head-Content协议数据帧解析的实践案例 162
  - 7.3 Encoder原理与实践 164
    - 7.3.1 MessageToByteEncoder编码器 165
    - 7.3.2 MessageToMessageEncoder编码器 166
  - 7.4 解码器和编码器的结合 167
    - 7.4.1 ByteToMessageCodec编解码器 168
    - 7.4.2 CombinedChannelDuplexHandler组合器 169
  - 7.5 本章小结 169
- 第8章 JSON和ProtoBuf序列化 171
  - 8.1 详解粘包和拆包 172
    - 8.1.1 半包问题的实践案例 172
    - 8.1.2 什么是半包问题 174
    - 8.1.3 半包现象的原理 174
  - 8.2 JSON协议通信 175
    - 8.2.1 JSON序列化的通用类 175
    - 8.2.2 JSON序列化与反序列化的实践案例 176
    - 8.2.3 JSON传输的编码器和解码器之原理 178
    - 8.2.4 JSON传输之服务器端的实践案例 179
    - 8.2.5 JSON传输之客户端的实践案例 180
  - 8.3 Protobuf协议通信 182
    - 8.3.1 一个简单的proto文件的实践案例 182
    - 8.3.2 控制台命令生成POJO和Builder 183
    - 8.3.3 Maven插件生成POJO和Builder 183
    - 8.3.4 消息POJO和Builder的使用之实践案例 184
  - 8.4 Protobuf编解码的实践案例 187
    - 8.4.1 Protobuf编码器和解码器的原理 187
    - 8.4.2 Protobuf传输之服务器端的实践案例 188
    - 8.4.3 Protobuf传输之客户端的实践案例 189
  - 8.5 详解Protobuf协议语法 191
    - 8.5.1 proto的头部声明 191

- 8.5.2 消息结构体与消息字段 192
- 8.5.3 字段的数据类型 193
- 8.5.4 其他的语法规则 194
- 8.6 本章小结 195
- 第9章 基于Netty的单体IM系统的开发实践 196
  - 9.1 自定义ProtoBuf编解码器 196
    - 9.1.1 自定义Protobuf编码器 197
    - 9.1.2 自定义Protobuf解码器 198
    - 9.1.3 IM系统中Protobuf消息格式的设计 199
  - 9.2 概述IM的登录流程 202
    - 9.2.1 图解登录/响应流程的9个环节 203
    - 9.2.2 客户端涉及的主要模块 203
    - 9.2.3 服务器端涉及的主要模块 204
  - 9.3 客户端的登录处理的实践案例 204
    - 9.3.1 LoginConsoleCommand和User POJO 205
    - 9.3.2 LoginSender发送器 207
    - 9.3.3 ClientSession客户端会话 209
    - 9.3.4 LoginResponseHandler登录响应处理器 211
    - 9.3.5 客户端流水线的装配 212
  - 9.4 服务器端的登录响应的实践案例 213
    - 9.4.1 服务器流水线的装配 214
    - 9.4.2 LoginRequestHandler登录请求处理器 215
    - 9.4.3 LoginProcessor用户验证逻辑 216
    - 9.4.4 EventLoop线程和业务线程相互隔离 217
  - 9.5 详解ServerSession服务器会话 218
    - 9.5.1 通道的容器属性 219
    - 9.5.2 ServerSession服务器端会话类 220
    - 9.5.3 SessionMap会话管理器 222
  - 9.6 点对点单聊的实践案例 223
    - 9.6.1 简述单聊的端到端流程 223
    - 9.6.2 客户端的ChatConsoleCommand收集聊天内容 224
    - 9.6.3 客户端的CommandController发送POJO 224
    - 9.6.4 服务器端的ChatRedirectHandler消息转发 225
    - 9.6.5 服务器端的ChatRedirectProcessor异步处理 226
    - 9.6.6 客户端的ChatMsgHandler接收POJO 227
  - 9.7 详解心跳检测 228
    - 9.7.1 网络连接的假死现象 228
    - 9.7.2 服务器端的空闲检测 229
    - 9.7.3 客户端的心跳报文 230
  - 9.8 本章小结 232
- 第10章 ZooKeeper分布式协调 233
  - 10.1 ZooKeeper伪集群安装和配置 233
    - 10.1.1 创建数据目录和日志目录： 234
    - 10.1.2 创建myid文件 234
    - 10.1.3 创建和修改配置文件 235
    - 10.1.4 配置文件示例 237
    - 10.1.5 启动ZooKeeper伪集群 238
  - 10.2 使用ZooKeeper进行分布式存储 239
    - 10.2.1 详解ZooKeeper存储模型 239
    - 10.2.2 zkCli客户端命令清单 240
  - 10.3 ZooKeeper应用开发的实践 241
    - 10.3.1 ZkClient开源客户端介绍 242
    - 10.3.2 Curator开源客户端介绍 242
    - 10.3.3 Curator开发的环境准备 243

- 10.3.4 Curator客户端实例的创建 244
- 10.3.5 通过Curator创建ZNode节点 245
- 10.3.6 在Curator中读取节点 247
- 10.3.7 在Curator中更新节点 248
- 10.3.8 在Curator中删除节点 249
- 10.4 分布式命名服务的实践 251
  - 10.4.1 ID生成器 252
  - 10.4.2 ZooKeeper分布式ID生成器的实践案例 253
  - 10.4.3 集群节点的命名服务之实践案例 254
  - 10.4.4 使用ZK实现SnowflakeID算法的实践案例 256
- 10.5 分布式事件监听的重点 261
  - 10.5.1 Watcher标准的事件处理器 261
  - 10.5.2 NodeCache节点缓存的监听 265
  - 10.5.3 PathChildrenCache子节点监听 267
  - 10.5.4 Tree Cache节点树缓存 272
- 10.6 分布式锁的原理与实践 276
  - 10.6.1 公平锁和可重入锁的原理 276
  - 10.6.2 ZooKeeper分布式锁的原理 277
  - 10.6.3 分布式锁的基本流程 279
  - 10.6.4 加锁的实现 280
  - 10.6.5 释放锁的实现 285
  - 10.6.6 分布式锁的使用 287
  - 10.6.7 Curator的InterProcessMutex可重入锁 288
- 10.7 本章小结 289
- 第11章 分布式缓存Redis 290
  - 11.1 Redis入门 290
    - 11.1.1 Redis安装和配置 290
    - 11.1.2 Redis客户端命令 292
    - 11.1.3 Redis Key的命名规范 294
  - 11.2 Redis数据类型 295
    - 11.2.1 String字符串 295
    - 11.2.2 List列表 296
    - 11.2.3 Hash哈希表 297
    - 11.2.4 Set集合 298
    - 11.2.5 Zset有序集合 299
  - 11.3 Jedis基础编程的实践案例 300
    - 11.3.1 Jedis操作String字符串 301
    - 11.3.2 Jedis操作List列表 303
    - 11.3.3 Jedis操作Hash哈希表 304
    - 11.3.4 Jedis操作Set集合 305
    - 11.3.5 Jedis操作Zset有序集合 306
  - 11.4 JedisPool连接池的实践案例 308
    - 11.4.1 JedisPool的配置 308
    - 11.4.2 JedisPool创建和预热 310
    - 11.4.3 JedisPool的使用 312
  - 11.5 使用spring-data-redis完成 CRUD的实践案例 313
    - 11.5.1 CRUD中应用缓存的场景 313
    - 11.5.2 配置spring-redis.xml 315
    - 11.5.3 使用RedisTemplate模板API 316
    - 11.5.4 使用RedisTemplate模板API完成CRUD的实践案例 321
    - 11.5.5 使用RedisCallback回调完成CRUD的实践案例 323
  - 11.6 Spring的Redis缓存注解 325
    - 11.6.1 使用Spring缓存注解完成CRUD的实践案例 325
    - 11.6.2 spring-redis.xml中配置的调整 327

- 11.6.3 详解@CachePut和 @Cacheable注解 328
- 11.6.4 详解@CacheEvict注解 329
- 11.6.5 详解@Caching组合注解 330
- 11.7 详解SpringEL (SpEL) 331
  - 11.7.1 SpEL运算符 332
  - 11.7.2 缓存注解中的SpringEL表达式 334
- 11.8 本章小结 336
- 第12章 亿级高并发IM架构的开发实践 337
  - 12.1 如何支撑亿级流量的高并发IM架构的理论基础 337
    - 12.1.1 亿级流量的系统架构的开发实践 338
    - 12.1.2 高并发架构的技术选型 338
    - 12.1.3 详解IM消息的序列化协议选型 339
    - 12.1.4 详解长连接和短连接 339
  - 12.2 分布式IM的命名服务的实践案例 340
    - 12.2.1 IM节点的POJO类 341
    - 12.2.2 IM节点的ImWorker类 342
  - 12.3 Worker集群的负载均衡之实践案例 345
    - 12.3.1 ImLoadBalance负载均衡器 346
    - 12.3.2 与WebGate的整合 348
  - 12.4 即时通信消息的路由和转发的实践案例 349
    - 12.4.1 IM路由器WorkerRouter 349
    - 12.4.2 IM转发器WorkerReSender 352
  - 12.5 Feign短连接RESTful调用 354
    - 12.5.1 短连接API的接口准备 355
    - 12.5.2 声明远程接口的本地代理 355
    - 12.5.3 远程API的本地调用 356
  - 12.6 分布式的在线用户统计的实践案例 358
    - 12.6.1 Curator的分布式计数器 358
    - 12.6.2 用户上线和下线的统计 360
  - 12.7 本章小结 361
  - • • • • ([收起](#))

[Netty、Redis、Zookeeper高并发实战\\_下载链接1](#)

标签

并发

Zookeeper

Netty

Java

Redis



计算机

编程

尼恩

## 评论

粗略地看了几个章节，Java NIO 和 Netty 的部分写得不错，概念解释地比较清晰

-----  
排版很差，讲述一般。

-----  
前半本还好，后半本感觉对新手很不友好。

-----  
前两章翻一翻就好了。后面讲zk还有错误。不推荐购买阅读。

-----  
版权归作者所有，任何形式转载请联系作者。 作者：梁生（来自豆瓣）  
来源：<https://www.douban.com/doubanapp/dispatch/review/12464798>

看了八个章节了，说说我的感受吧。

作者应该是一个很会讲故事的人，小孩子应该跟他也很融洽。

这本书对于我这种netty小白来说，非常受用，其实我连网络底层知识都没具备多少，作者讲的还是很明白的。第四章开始，到第八章我都是精读的，有的地方设计得很巧妙，循序渐进，使得其当得讲解得方式让我感觉也不太难理解，配合适当给出的经验之谈，我觉得自己也是牛逼哄哄的。

能读到这本书，算是在2020年2月份，疫病下的唯一安慰吧。

-----  
优点：知识点讲得很浅显，没有什么难度。

缺点：用微信读书看的，代码都粘在一起，类型和变量名中间要自己找到分隔，看得头晕脑胀。序言中的源码地址已经不存在。从gitee找到一个推断是作者本人的源码库，

然而那份代码也不能直接跑：想运行前面章节的代码，却发现依赖到后面章节代码所在模块，而后面章节模块报错（可能还要装别的软件），所以前面章节代码也连带着不能跑。总之，整个项目结构有问题，不该耦合的耦合在一起，这要是工作中的代码，维护起来肯定头疼。从代码足以看出作者水平，所以这本书可看可不看。要看就看看基本知识点，要学怎么写代码就去看看别的书吧。

-----  
这本书真的垃圾，主要是作者太不负责了，编辑也是，不会好好审一下嘛？书里的代码全是粘着的，还一堆错别字，代码还是残缺的，这让人怎么看。。。

-----  
好好的一本书，就不能好好排版？看代码都能累死个人，作者和编辑都太不负责了吧

-----  
[Netty、Redis、Zookeeper高并发实战\\_下载链接1](#)

## 书评

看了八个章节了，说说我的感受吧。  
作者应该是一个很会讲故事的人，小孩子应该跟他也很融洽。  
这本书对于我这种netty小白来说，非常受用，其实我连网络底层知识都没具备多少，作者讲的还是很明白的。第四章开始，到第八章我都是精读的，有的地方设计得很巧妙，循序渐进，使得其...

-----  
[Netty、Redis、Zookeeper高并发实战\\_下载链接1](#)