Go语言高并发与微服务实战

Go语言高并发与微服务实战_下载链接1_

著者:朱荣鑫,黄迪璇,张天

出版者:中国铁道出版社有限公司

出版时间:2020-4

装帧:平装

isbn:9787113266622

近年来云原生技术发展迅猛，帮助开发者在云上快速和频繁地构建、发布和部署应用，以提高开发效率和快速定位故障。
微服务作为开展云原生技术落地的核心，它将复杂的单体应用按照业务划分并进行有效地拆分，每个微服务都可以进行独立部署和开发，大大提升了应用开发效率。Go语言作为新生代的编译型编程语言，具备语法简单、高并发性能良好和编译速度快等特点，是微服务架构落地实践的绝妙利器。

作者介绍:

朱荣鑫

软件工程硕士，微服务早期实践者，微服务方面技术专家，对高并发、分布式有多年深入的实践经验。掘金优秀作者，CSDN博客专家。公众号“aoho求索”的作者。

黄迪璇

毕业于南京大学，目前就职于国内一线互联网公司，曾就职于字节跳动、腾讯，具有多年服务端开发经验，技术极客，热衷于新技术的研究和实践。

张天

服务端技术专家，精耕于微服务、分布式、数据库和性能调优等后端开发领域。著有《Spring Cloud微服务架构进阶》，CSDN博客专家。公众号“程序员历小冰”的作者。

目录: 第一篇 云原生与微服务
云原生与微服务分别是什么，它们之间有什么关系呢？本部分围绕云原生与微服务的概念展开介绍，我们透过云计算的历史和系统架构的演进，具体了解这两个概念的意义及其背 后的技术发展。
第 1 章 云原生架构
1.1 云计算的历史 1
1.1.1 云计算的基础：虚拟化技术 1
1.1.2 基于虚拟机的云计算 3
1.1.3 容器的横空出世和容器编排大战 5
1.1.4 云计算演进总结 6
1.2 云原生是什么 7
1.2.1 云原生出现的背景 7
1.2.2 云原生的定义 8
1.2.3 云原生与 12 因素 9
1.3 云原生的基础架构 11
1.3.1 微服务 11
1.3.2 容器 12
1.3.3 服务网格 13
1.3.4 DevOps 14
1.4 小结 15
第 2 章 微服务概述
2.1 系统架构的演进 16
2.1.1 单体架构 16
2.1.2 垂直分层架构 17
2.1.3 SOA 面向服务架构 17
2.1.4 微服务架构 19
2.1.5 云原生架构 21
2.2 常见的微服务框架 22
2.2.1 Java 中的 Spring Cloud 与 Dubbo 框架 22

2.2.2 Go 语言中的 Go Kit 与 Go Micro 框架 24
2.3 微服务设计的六大原则 27
1．高内聚，低耦合 27
2．高度自治 27
3．以业务为中心 28
4．弹性设计 28
5．日志与监控 28
6．自动化 28

2.4 领域驱动设计 28
2.4.1 设计微服务的困境 28
2.4.2 解困之法：领域驱动设计（DDD） 29
2.4.3 DDD 的应用领域 30
2.4.4 DDD 领域划分 31
2.4.5 微服务架构中的团队组织和管理 33
2.5 小结 34
第二篇 Go 语法基础与特性功能
在正式进入微服务组件的学习之前，我们要巩固一下 Go 语言的基础，包括容器、原生数据类型、函数与接口、结构体和方法等常用的语法基础；其次是 Go 语言的特性功能：反 射与并发模型，介绍 Go 语言协程、通道、多路复用和同步的具体实践；最后是 Golang Web 的相关介绍，一起构建一个完整的 Go Web 服务器。
第 3 章 Go 语言基础
3.1 Go 语言介绍 35
3.2 环境安装 36
3.2.1 Go 开发包安装 36
3.2.2 第一个 Go 语言程序 38
3.2.3 编译工具 40
3.3 基本语法 41
3.3.1 变量的声明与初始化 41
3.3.2 原生数据类型 43
【实例 3-1】分别以 byte 和 rune 的方式遍历字符串 44
3.3.3 指针 45
【实例 3-2】使用 flag 从命令行中读取参数 47
3.3.4 常量与类型别名 48
3.3.5 分支与循环控制 49
3.4 Go 中常用的容器 50

3.4.1 数组 50
3.4.2 切片 51
【实例 3-3】切片的动态扩容 53
3.4.3 列表与字典 54
3.4.4 容器遍历 57
【实例 3-4】对给出的数组 nums、切片 slis 和字典 tmpMap 分别进行遍历 57
3.5 函数与接口 58
3.5.1 函数声明和参数传递 58
3.5.2 匿名函数和闭包 59
【实例 3-5】使用回调函数处理字符串 59
【实例 3-6】用闭包的特性实现一个简单的计数器 60
3.5.3 接口声明和嵌套 61
3.5.4 函数体实现接口 62
3.6 结构体和方法 62
3.6.1 结构体的定义 63
3.6.2 结构体的实例化和初始化 63
3.6.3 方法与接收器 64
【实例 3-7】为 Person 结构体添加修改姓名和输出个人信息两个方法 65
3.6.4 结构体实现接口 66
【实例 3-8】使用一个结构体同时实现 Cat 和 Dog 接口 66
3.6.5 内嵌和组合 67
【实例 3-9】内嵌不同结构体表现不同行为 68
3.7 小结 69
第 4 章 进阶——Go 语言高级特性
4.1 依赖管理 70

4.1.1 包管理 70
4.1.2 GOPATH 72
4.1.3 Go Modules 73
4.2 反射基础 73
4.2.1 reflect.Type 类型对象 74
4.2.2 类型对象 reflect.StructField 和 reflect.Method 76
4.2.3 reflect.Value 反射值对象 78
【实例 4-1】使用反射调用接口方法 80
4.3 并发模型 82
4.3.1 并发与并行 82
4.3.2 CSP 并发模型 82
4.3.3 常见的线程模型 83
4.3.4 MPG 线程模型概述 85

4.4 并发实践 87
4.4.1 协程 goroutine 87
4.4.2 通道 channel 89
【实例 4-2】协程使用 channel 发送和接收数据 90
【实例 4-3】使用带缓冲区的 channel 91
【实例 4-4】使用 switch 从多个 channel 中读取数据 92
4.4.3 sync 同步包 94
【实例 4-5】使用 sync.Mutex 控制多 goroutine 串行执行 94
【实例 4-6】sync.RWMutex 允许多读和单写 95
【实例 4-7】sync.WaitGroup 阻塞主 goroutine 直到其他 goroutine 执行结束 97
【实例 4-8】使用 sync.Map 并发添加数据 98
4.5 小结 99
第 5 章 构建 Go Web 服务器
5.1 Web 的工作原理 100
5.1.1 HTTP 协议详解 100
5.1.2 访问 Web 站点的过程 103
5.2 使用 Go 语言构建服务器 104
【实例 5-1】快速搭建一个 Go Web 服务器 104
5.3 接收和处理请求 105
5.3.1 Web 工作的几个概念 106
5.3.2 处理器处理请求 107
5.3.3 解析请求体 109
【实例 5-2】Go Web 请求体解析 109
5.3.4 返回响应体 111
【实例 5-3】返回响应体实践 112
5.4 实践案例：Golang Web 框架 Gin 实践 113
5.5 服务端数据存储 116
5.5.1 内存存储 116
【实例 5-4】服务端基于内存的存储方式实践 116
5.5.2 database/sql 接口 118
5.5.3 关系数据库存储（MySQL） 118
【实例 5-5】服务端基于 MySQL 的存储方式实践 119
5.5.4 Nosql 数据库存储（MongoDB） 120
【实例 5-6】服务端基于 MongoDB 的存储方式实践 121
5.6 Golang ORM 框架 beego 实践 122
5.7 小结 125

第三篇 微服务核心组件
本部分是全书的核心，介绍微服务中各个核心组件的原理和实践应用，包括分布式配置中心、服务注册与发现、微服务网关、微服务的容错、微服务中的通信与负载均衡、统

一认
证与授权、微服务中的链路追踪。通过组件原理的介绍、组件的选型对比以及组件的实践应 用，吃透每一个微服务组件。
第 6 章 服务注册与发现
6.1 服务注册与发现的基本原理 126
6.1.1 服务注册与发现中心的职责 126
6.1.2 服务实例注册服务信息 127
6.1.3 CAP 原理 127
6.2 常用的服务注册与发现框架 128
6.2.1 基于 Raft 算法的开箱即用服务发现组件 Consul 128
6.2.2 基于 HTTP 协议的分布式 key/Value 存储组件 Etcd 130
6.2.3 重量级一致性服务组件 Zookeeper 131
6.2.4 服务注册与发现组件的对比与选型 132
6.3 Consul 安装和接口定义 133
6.3.1 Consul 的安装与启动 133
6.3.2 Go-kit 项目结构 134
6.3.3 服务注册与发现接口 135
6.3.4 项目的总体结构 135
6.4 实践案例：直接使用 HTTP 的方式和 Consul 交互 140
6.4.1 服务注册与健康检查 142
6.4.2 服务注销 144
6.4.3 服务发现 146
6.5 实践案例：借助 Go-kit 服务注册与发现包和 Consul 交互 147
6.5.1 服务注册与健康检查 148
6.5.2 服务注销 149
6.5.3 服务发现 150
6.5.4 服务实例信息缓存 150
6.5.5 MyDiscoverClient 和 KitDiscoverClient 的比较 153
6.6 实践案例：基于服务注册与发现的 string-service 153
6.6.1 项目结构 153
6.6.2 各层构建 154
6.7 小结 162

第 7 章 远程过程调用 RPC
7.1 RPC 机制和实现过程 164
7.1.1 RPC 机制 164
7.1.2 传递参数 167
7.1.3 通信协议制定 168
7.1.4 出错和超时处理 170
7.1.5 通用 RPC 接口 171
7.2 简易的 Go 语言原生 RPC 172
7.2.1 实践案例：Go 语言 RPC 过程调用实践 172
7.2.2 服务端注册实现原理分析 175
7.2.3 服务端处理 RPC 请求原理分析 178
7.2.4 客户端发送 RPC 请求原理分析 182
7.2.5 资源重用 187
7.3 高性能的 gRPC 188
7.3.1 gRPC 的安装 189
7.3.2 实践案例：gRPC 过程调用实践 190
7.3.3 流式编程 193
【实例 7-1】gRPC 流式请求 193
7.4 便捷的 Go-kit RPC 196
7.4.1 Go-kit 简介 196
7.4.2 实践案例：Go-kit 过程调用实践 197

7.5 小结 202
第 8 章 分布式配置中心
8.1 如何管理分布式应用的配置 203
8.2 常见分布式配置中心开源组件 204
8.2.1 Spring Cloud Config. 204
8.2.2 Apollo 205
8.2.3 Disconf 208
8.2.4 分布式配置中心的对比 210
8.3 应用 Spring Cloud Config 统一管理配置 210
8.3.1 搭建 Spring Cloud Config Server 210
8.3.2 Viper 介绍 213
【实例 8-1】Viper 实现读取本地配置信息 214
8.3.3 实战案例：动手实现 Spring Cloud Config 的 Go 语言客户端 216
8.4 实践案例：实现配置的热更新 219
8.4.1 如何实现热更新 219
8.4.2 Go 语言客户端改进 220

8.4.3 结果验证 223
8.5 配置信息的加密解密 224
8.5.1 JCE 环境安装 225
8.5.2 对称加密与解密 225
8.5.3 非对称加密与解密 226
8.6 小结 227
第 9 章 微服务网关
9.1 微服务网关介绍与功能特性 228
9.2 实践案例：自己动手实现一个网关 231
9.2.1 实现思路 231
9.2.2 编写反向代理方法 232
9.2.3 编写入口方法 233
9.2.4 运行 235
9.2.5 测试 235
9.3 API 网关选型 235
9.3.1 标配组件：Nginx 网关 236
9.3.2 Java 前置网关服务最佳选型：Netflix Zuul 237
9.3.3 高可用服务网关：Mashape Kong 239
9.3.4 三种常用 API 网关组件的指标对比 240
9.4 Kong 接入 240
9.4.1 为什么使用 Kong 240
9.4.2 Kong 安装实践 241
【实例 9-1】Docker 方式安装 Kong 242
9.4.3 创建服务 244
9.4.4 创建路由 245
9.5 安装 Kong 插件 246
9.5.1 跨域身份验证：JWT 认证插件 246
9.5.2 系统监控报警：Prometheus 可视化监控插件 248
9.5.3 实时链路数据追踪：Zipkin 插件 250
9.5.4 进阶应用：自定义 Kong 插件 252
【实例 9-2】自定义鉴权插件 token-auth 252
9.6 小结 257
第 10 章 微服务的容错处理与负载均衡
10.1 服务熔断 258
10.1.1 分布式系统中的服务雪崩 258
10.1.2 服务熔断保障系统可用性 260
10.1.3 断路器 261

10.2 负载均衡 262
10.2.1 负载均衡类型 262
10.2.2 负载均衡算法 262
10.3 实践案例：服务熔断和负载均衡使用 263
10.3.1 负载均衡器 263
10.3.2 服务编写 264
10.3.3 使用 Go-kit Hystrix 中间件 270
【实例 10-1】使用 Go-kit Hystrix 中间件修饰 Endpoint 270
10.4 Hystrix 详解 271
10.4.1 Hystrix 基本使用 272
10.4.2 运行流程 273
10.4.3 常用参数配置 274
10.5 Hystrix 监控面板 275
10.5.1 获取 Hystrix 命令调用信息 275
10.5.2 使用 Hystrix Dashboard 可视化面板 277
10.6 实践案例：在网关中添加 Hystrix 熔断和负载均衡 279
10.7 小结 282
第 11 章 统一认证与授权
11.1 微服务安全的挑战和现状 283
11.2 常见的认证与授权方案 283
11.2.1 当前行业授权标准 OAuth2 283
11.2.2 数据共享的分布式 Session 287
11.2.3 安全传输对象 JWT 288
11.3 实践案例：基于 OAuth2 协议和 JWT 实现一套简单 的认证和授权系统 290
11.3.1 系统整体架构 290
11.3.2 授权服务器 291
1．用户服务和客户端服务 292
2．TokenGrant 令牌生成器 294
3．TokenService 令牌服务 296
4．TokenStore 令牌存储器 300
5．/oauth/token 和/oauth/check_token 303
6．请求访问令牌和刷新令牌 306
11.3.3 资源服务器 311
1．令牌认证 311
2．鉴权 312
3．访问受限资源 313
11.4 小结 317

第 12 章 分布式链路追踪
12.1 诊断分布式系统的问题 318
12.1.1 为什么需要分布式链路追踪 318
12.1.2 什么是分布式链路追踪 319
12.1.3 分布式链路追踪规范：OpenTracing 320
12.1.4 分布式链路追踪的基础概念 321
12.2 几种流行的分布式链路追踪组件 323
12.2.1 简单易上手的 Twitter Zipkin 323
12.2.2 云原生链路监控组件 Uber Jaeger 324
12.2.3 探针性能低损耗的 SkyWalking 326
12.2.4 链路统计详细的 Pinpoint 327
12.2.5 4 种分布式链路追踪组件的指标对比 328
12.3 实践案例：应用 Zipkin 追踪 Go 微服务 329
12.3.1 微服务中集成 zipkin-go 330
12.3.2 Go-kit 微服务框架集成 Zipkin 实现链路追踪 337

1．HTTP 调用方式的链路追踪 338
2．gRPC 调用方式的链路追踪 342
12.4 小结 346
第四篇 综合实战
本部分是商品秒杀系统的实战项目，综合难度相对较高，我们通过分析业务系统的领域设计，将系统划分成具体的微服务，整合各个微服务组件，最终实现一个高并发的商品秒杀 系统。
第 13 章 综合实战：秒杀系统的设计与实现
13.1 秒杀系统简介 347
13.2 项目架构简介 350
13.2.1 项目简述 350
13.2.2 架构信息 350
13.2.3 流程简介 352
13.3 整合升级：各个微服务脚手架的组装 353
13.3.1 服务注册和发现 353
13.3.2 负载均衡策略 357
13.3.3 RPC 客户端装饰器 360
13.3.4 限流 362
13.3.5 Go 语言 Redis 使用简介 364

13.3.6 Zookeeper 集成 366
13.3.7 Go-kit 开发利器 Truss 367
13.4 秒杀核心逻辑 368
13.4.1 秒杀业务系统 370
13.4.2 秒杀核心系统 380
13.4.3 秒杀管理系统 384
13.5 性能压测 386
13.5.1 查看服务的配置文件 386
13.5.2 压测实验 387
13.6 小结 390
・・・・・・(收起)

Go语言高并发与微服务实战_下载链接1_

标签

微服务

go

golang

编程

docker

评论

参考性一般

最近在转go开发，参考做了项目，原理方面还可以更加深入。

泛泛而谈，没把握好定位

内容还可以，不过基于的是 Go-kit讲解微服务的组件，如果是Go-Micro就好了。

Go语言高并发与微服务实战_下载链接1_

书评

背景介绍：我是有go使用经验的，希望在微服务架构上能有进一步了解，因此买了《Go语言高并发与微服务实战》和《微服务架构设计模式》这两本书，我简单分享一下读完《Go语言高并发与微服务实战》这本书后的感受。
整本书不到400页，跳着看完了。总的来说，这本书一般般，看完后...

Go语言高并发与微服务实战_下载链接1_