高并发高可用的分布式电商平台架构研究

耿晓利,张 芒,尹永宏
(广州大学华软软件学院,广东 广州 510990)

摘 要:为了克服软件架构在巨大 QPS 冲击下性能低、可拓展性差等弊端,解决在经常高并发场景下的程序崩溃、服务宕机以及数据出错等问题,文章针对亟需平稳运行的电商业务,设计并实现了一套基于分布式架构的高可用新型电子商务系统。该系统提供了高可靠的分布式框架,含有服务节点自动注册与发现、配置统一管理以及服务集群容错等功能。在可能出错的服务节点上,运用了安全认证鉴权、分布式缓存锁、消息服务、分布式搜索引擎等手段,使得服务在透明化远程调用中既强化了程序的安全可靠,又保障了数据在传输过程中的精准正确。系统在开发中采用前后分离的 Restful Api 架构风格和容器化部署方法,测试并验证了在高并发场景下的功能可用性和性能可靠性,为百万级流量系统的软件架构技术选型提供了一种可行的服务化治理方案。

关键词:分布式;微服务;事务机制;Docker 容器;高并发
中图分类号:TP391 **文献标识码**:A **文章编号**:1673-629X(2021)02-0111-05
doi:10.3969/j.issn.1673-629X.2021.02.021

Research on Distributed E-commerce Platform Architecture with High Concurrency and High Availability

GENG Xiao-li, ZHANG Mang, YIN Yong-hong
(South China Institute of Software Engineering, Guangzhou 510990, China)

Abstract: In order to overcome the disadvantages of software architecture such as low performance and poor extensibility under the impact of huge QPS and solve the problems of program crash, service failure and data error under frequently high concurrent scenarios, a new high-availability e-commerce system based on distributed architecture is designed and implemented for e-commerce which needs to run smoothly. The system provides a highly reliable distributed framework with the functions of automatic registration and discovery of service nodes, unified configuration management and fault tolerance of service cluster. On the service node which may make mistakes, the security authentication, the distributed cache lock, the message service, the distributed search engine and other means are used, so that the service in transparent remote invocation not only strengthens the security and reliability of the program, but also ensures the accuracy of data in the process of transmission. The system uses the split Restful Api architecture style and container deployment method to test and verify the functional availability and performance reliability in high concurrency scenarios, which provides a feasible service-oriented governance scheme for the selection of software architecture technology of million-class traffic system.

Key words: distributed; microservices; transaction mechanism; Docker containers; high concurrency

0 引 言

微服务[1]具有模块化开发、分布式部署的特性,各个模块独立部署、运行更新,有着灵活的扩展性,适用于海量数据单体应用的解耦[2]。Dubbo[3]是阿里巴巴公司开源的一个高性能和透明化的 PRC 远程调用框架,搭配 Zookeeper[4]作为分布式服务的注册中心,在如今高可用解决方案的技术选型中被广泛使用[5]。

基于分布式的架构技术正在不断发展,各大互联网厂商的分布式框架不断迭代,分布式架构在电商行业应用中蓬勃发展,较多文献对分布式架构下电商平台的构建进行了研究[6-8],同时也向金融、政务、社交等领域逐步渗透。

该文以电商业务为切入点,以 SpringBoot[4]和 MyBatis 作为实现基础服务单元的企业级基础框架,综合集成分布式开源框架 Dubbo、注册中心 Zookeeper、百度开源统一配置中心 Disconf、消息中间

收稿日期:2020-03-21 **修回日期**:2020-07-22
基金项目:2018 年广东省重点平台及科研项目(青年人才项目-自然科学类)(2018KQNCX389)
作者简介:耿晓利(1982-),女,硕士,CCF 会员(B8033M),研究方向为分布式数据库、数据分析与挖掘;张 芒,高级工程师,研究方向为分布式系统。

件 RabbitMq、Spring 安全认证授权框架 Spring Security[9]、缓存数据库 Redis[10] 等,采用前后分离的 Restful Api[11] 架构风格和 Docker[12-13] 容器化部署方法,设计并实现了基于 B2C 的新零售购物平台。系统通过服务化的手段,按照功能维度划分,使各功能服务中心最大限度地解耦,各服务通过注册中心自动完成服务的注册与发现,互相隔离,使用消息中间件技术进行消息订阅、异步处理、流量削峰等。

1 系统架构

本系统有用户中心、商品中心、营销中心、库存中心、物流中心、客服中心以及数据统计中心,各服务中心之间以 RPC(远程调用)的方式进行通信,统一在 Disconf 配置中心下载各自指定的配置文件,同时以服务提供者的角色注册在注册中心 Zookeeper 上。控制器层作为服务消费者角色向注册中心 Zookeeper 订阅其需消费的服务,通过 Nginx 反向代理分发至不同客户端。系统采用 Spring Security 和 Jwt 作为安全认证授权框架,客户端登录到用户中心,用户中心将通过此框架向客户端发布带有加密用户信息的授权令牌(token),并在此后用户访问中拦截其请求,校验令牌信息和验证用户身份是否正确,正确则放行,否则将拒绝此用户访问。

系统架构如图 1 所示。

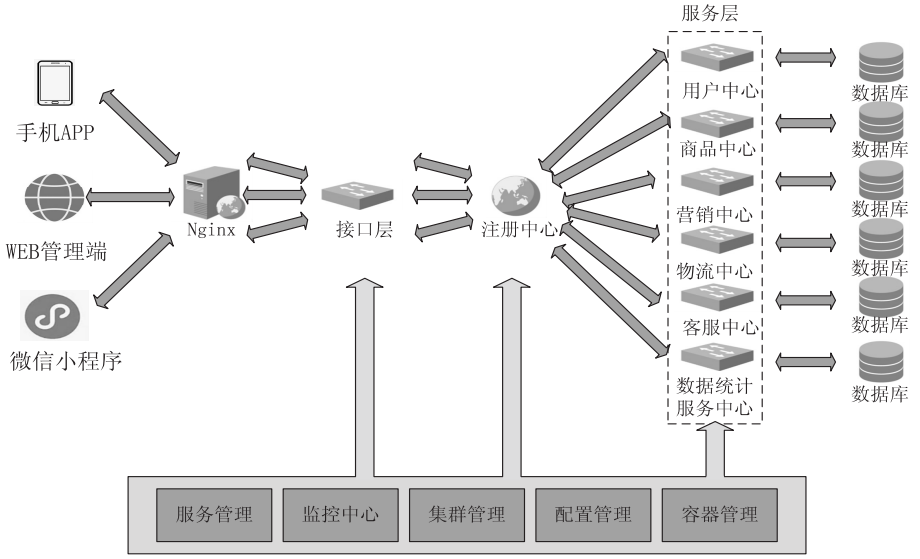


图 1 系统架构

1.1 应用层

系统架构应用层主要有移动端 APP、微信小程序和 WEB 端后台管理系统。各不同客户端调用 Nginx 提供的统一接口访问地址,再通过 Nginx 将请求反向代理至服务层,在应用层中,Nginx 将担负着负载均衡和限流的任务,在系统流量突然爆发的时间段,将流量按照调度算法均衡地分发至不同服务器,同时狙击恶意请求流量和恶意攻击;是洪峰流量进入后台服务的一道重要的安检口,也是保护系统平稳运行的重要关卡之一。在该系统设计中,应用层还有一个重要的功能:在高并发处理中使用 Nginx 降级策略,将牺牲一部分请求来保证系统的稳定,从而保证了系统的高可用性。

1.2 接口层

接口层在系统架构中扮演着服务消费者的角色,是应用层通过 Nginx 将请求分发的目的地。在接口层中,设计了基于 Spring Security 和 Jwt 安全授权访问过滤器,其主要作用是通过解析令牌(token)校验请求的合法性以及是否拥有所访问接口权限,保证接口安全。由于 Jwt 的认证方式是无状态的,可以在多个服务之间共享,加上生命周期的设置,实现了系统的 SSO 单点登录功能。

接口层通过订阅注册中心上的已发布的服务,完成应用层的请求和回应。该层提供 Restful Api 架构风格的调用地址,通过远程调用的方式完成与服务层之间的数据交互。

1.3 服务层

服务隔离是指将系统服务分割开,是为了在系统发生故障时,能限定传播范围和影响范围,即发生故障时不会出现像单体架构一样的滚雪球效应。在该系统架构中,注册中心的一个节点出现故障后,注册中心通过 HTTP 心跳检测会将该节点排除出服务提供者名单,自动切换其他可用节点,从而保证只有出现故障的节点不可用,其他节点仍然是可用的。该系统服务层是系统功能的具体实现,扮演着服务提供者的角色。按照系统功能维度划分不同的服务中心,各服务相互隔离、独立部署,拥有自己独立的数据库,是系统完成特定功能领域的集合;通过服务化的设计,将单体架构

中捆在一起的服务抽离,大大地降低了系统功能耦合度,为系统功能的不断扩展和维护奠定了基调。在秒杀业务场景下,随着接口调用次数不断增高,此时的秒杀服务单节点的负荷率将直线上升,由于服务器资源的有限性,将直接影响到其他非秒杀业务的稳定性。如图 2 所示,该系统在设计中,将在商品服务中划分出以秒杀业务为核心的单独服务集群,从而解决了在秒杀业务洪峰流量到来之时,保证其他业务的稳定运行,不受其影响。

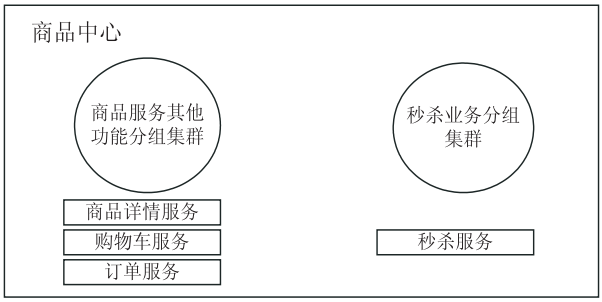


图 2　服务集群分组

2　平台功能实现

2.1　系统基础服务层实现

系统中设计的公共类为基础服务层,在此层中集成了基础工具类、接口统一返回格式、各枚举类、各配置类等,作为系统底层服务提供给各业务服务中心使用。基础服务层最重要的一个功能是与统一配置中心 Disconf 交互,在项目启动时,将各配置文件从配置中心拉取到配置文件夹中,供各个服务中心使用;在此层中通过. xml 文件配置扫描包所在位置以及使用托管方式的 Disconf 配置,其中包含配置文件名称等。如图 3 所示,在初始化 Disconf 项目后通过浏览器 WEB 界面录入配置文件名,所属模块,版本号,开发环境。此设计将系统繁杂的配置信息从系统内部抽离出,由配置中心统一管理,一旦配置信息如数据库地址改变,在配置中心修改即可,配置中心发生修改会立即通知程序重新拉取配置文件,而不必通过停止程序的运行去一个个地修改配置,从而大大提高了系统的执行效率和维护成本。

图 3　统一配置中心 Disconf 配置录入

2.2　系统服务层实现

系统服务层在系统整体架构中扮演着服务提供者的角色,是系统各功能模块具体实现的汇集地,也是整个系统中业务处理能力要求最高的核心区域。该层使用了分布式搜索引擎 ElasticSearch 作为商品中心的商品搜索服务快速搜索框架,使用缓存数据库 Redis 完成商品详情信息、商品维度和用户维度存储等,使用开源中间件 RabbitMq 作为秒杀业务的通讯组件。

2.2.1　用户中心

在功能模块划分中,用户中心具有用户基础信息管理、用户行为管理、用户角色权限管理、用户地址管理、用户购物车管理、用户登录授权管理等功能。用户通过注册页面录入基础信息,系统根据普通用户和管理员设置不同权限,用户行为管理记录用户每次登录信息、搜索行为等。

2.2.2　商品中心

商品中心在整个服务层占据业务核心地位,根据业务细粒度划分有商品详情服务、购物车服务、订单服务、支付服务,评价服务,秒杀服务和库存服务等,业务详细流程如图 4 所示。用户浏览商品列表,选择商品查看详情,选择购买或者加入购物车后系统判断登录状态,未登录则完成用户登录流程后选择收货地址以及完成订单支付,订单支付后进入物流服务和评价服务。其中,当订单支付后以消息队列的方式通知库存服务进行库存调整操作。选择消息队列方式进行库存操作,是因为消息队列方式可以保证库存操作的原子性,避免订单数量超过及时库存数,造成负库存情况出现。其中秒杀服务独立于其他商品服务,划分为独立的服务分组,如图 5 所示,当秒杀业务开始时,用户订单支付后进入消息队列排队,消息队列根据及时库存数的 3 倍作为进入消息队列的请求数,将排队数量限制在一定的范围内,超过这个范围内的数量将直接返回秒杀结束的状态,成功进入消息队列的请求按照队列先进先出的原理,同时运用分布式事务机制[14],将订单支付和库存服务作为事务管理,订单支付和库存数的递减同时完成,否则执行回滚操作。这样既减轻了队列的负担,又能保证库存服务的准确性和系统的稳定性。

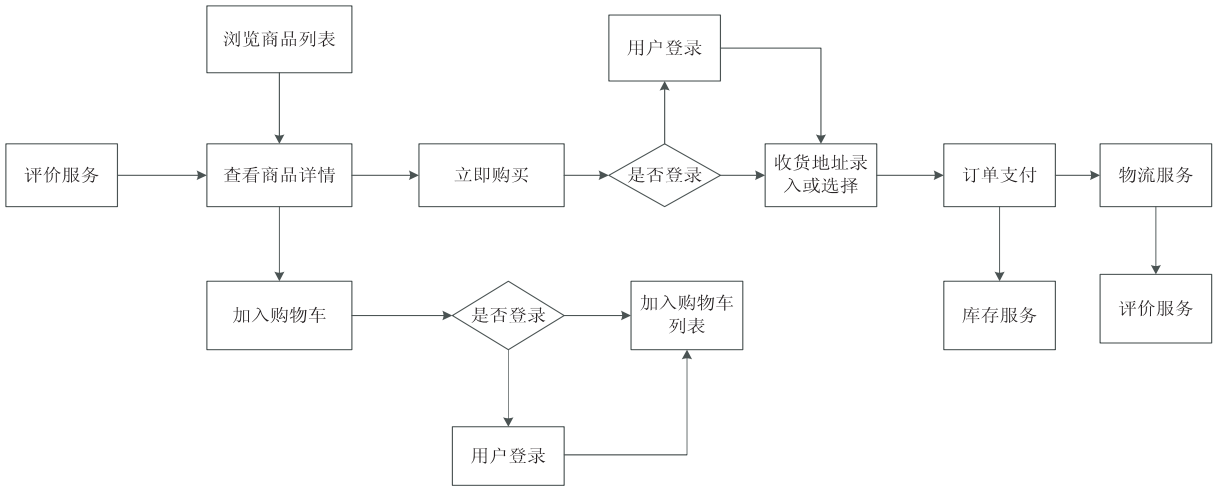


图 4 系统业务流程

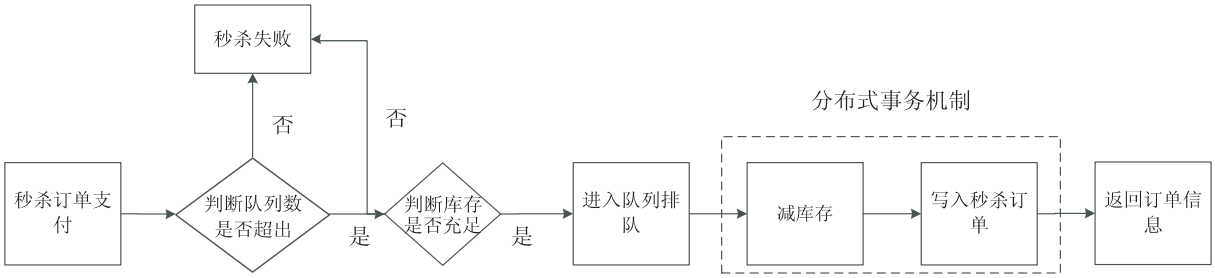


图 5 秒杀业务流程

2.2.3 营销中心

营销中心主要是为了完成商品优惠券发放、商品预热等功能。后台管理员通过优惠券面额、发放对象、使用期限等信息的填写,将优惠券存入系统表中,系统将以两种方式向用户发放优惠券,一种是以系统通知的方式向用户推送领取通知,用户点击通知跳转至领券页面完成优惠券领取,另一种方式是以用户查看商品详情的方式在商品详情页面放置优惠券领取栏,在用户查看商品的同时就可以领取相应的优惠券完成购买流程。商品预热功能主要是为了向添加在用户购物车里的商品推送商品降价信息,为了刺激商品交易额的增加,商品预热服务还将根据用户中心的用户行为服务获取到用户个人喜好,将通过个人喜好匹配相对应类型的商品,通过首页商品列表展示的方式向不同的用户推送。

2.2.4 物流中心

物流中心对接商品中心中的订单服务,完成用户订单支付后的物流信息跟踪、收货提醒等功能。物流中心对接阿里云物流接口,通过物流订单号以及快递公司名称的输入,获取到阿里云物流信息,将物流信息返回格式转换成系统显示格式以步骤条的形式显示在客户端,当物流信息更新至即将配送的状态,以系统通知的形式向用户推送收货提醒。

2.2.5 客服中心

客服中心使用 WebSocket[15] 实现,主要完成的功能是用户购买前的客服咨询功能以及购买后的售后功能。分为客服端和用户端,在聊天中支持商品链接发送、商品图片发送等功能。

2.2.6 数据统计中心

数据统计中心体现数据中台思想,主要统计周销量、月销量、季节销量、地区购买力、年龄段购买力等,完成不同维度销量对比,输出对应对比结果,以 echart 图表显示在后台 WEB 端,在抢购时间段,还将实时显示各商品销量,供商家进行实时库存调整。数据统计中心还将完成各类报表制作与查询,具体输出报表有销售报表、库存报表、财务报表等。

3 系统部署

为了适用分布式架构高并发高可用原则,系统部署采用 Docker 容器化部署方法,运行系统统一采用 Linux 发行版 CentOS7.6,选用服务器 16 台,每台主机为 8 核 3.0 GHz CPU,8 GB 内存,阿里云 OSS 存储对象主要用来存储用户头像、商品图片等系统静态资源。在运行系统 CentOS 上通过 Docker 镜像分别配置分布式框架开发环境和上线环境,如图 6 所示,相应配置集成 Tomcat、Nginx、Disconf、Zookeeper、Dubbo、Redis、MySql、RabbitMq 等,各服务中心代码打成 war 包后将放置在对应的 Tomcat 实例里,保证各服务之间相互隔离,互不影响。

图 6 为 CentOS 集成示例。

```
drwxr-xr-x 5 root     root     4096 Dec 27 10:15 disconf
drwxr-xr-x 3 root     root     4096 Nov  2  2019 dubbo
drwxr-xr-x 9 elsearch elsearch 4096 May  3 22:53 elasticsearch-6.2.4
drwxr-xr-x 3 root     root     4096 Mar 25 18:22 kibana
drwxr-xr-x 3 root     root     4096 Nov  3  2019 mongodb
drwxr-xr-x 2 root     root     4096 Nov  2  2019 mysql
drwxr-xr-x 3 root     root     4096 Nov  2  2019 nginx
drwxr-xr-x 2 root     root     4096 Nov  3  2019 rabbitMq
drwxr-xr-x 3 root     root     4096 Nov  2  2019 redis
drwxr-xr-x 3 root     root     4096 Nov  1  2019 zookeeper
```

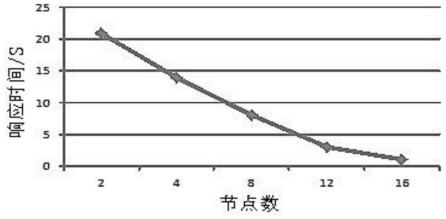
图 6　CentOS 集成示例

4　性能测试

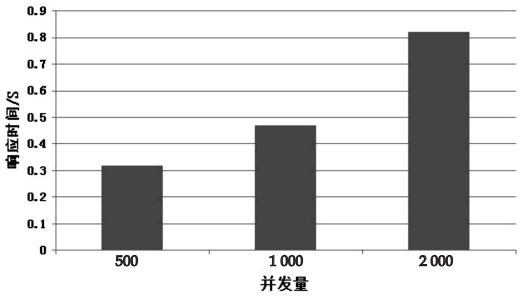
接口响应速率、CPU 占有率和磁盘 I/O 是系统高可用性的重要性能指标,系统从这三个方面展开进行测试验证。

4.1　接口响应速率

接口响应速率测试是从同一并发量下不同数量节点接口响应速率和不同并发量下多节点接口响应速率两方面展开。同一并发量不同数量节点请求的商品列表显示效率,主要测试点是商品中心模块的商品详情服务,测试用例为并发量 500,通过 Jmeter 脚本分别向不同节点的商品中心接口同时发起并发请求,结果如图 7(a)所示,随着节点数的不断增多,接口平均响应时间不断递减,在节点数为 16 的时候,接口响应时间达到 0.56 s,完全达到了测试要求。在不同并发量下多节点的接口平均响应时间,将测试点选在单个商品详情请求接口,测试用例为并发量 500、1 000、2 000,测试接口在这三种并发量下的平均响应时间。结果如图 7(b)所示,在节点数为 16 的相同条件下,接口的平均响应时间随着并发量的增多而不断增多,在测试 2 000个并发量的时候,接口平均响应时间达到 0.91 s,符合测试高并发原则要求。



(a)同一并发量不同数量节点接口响应速率



(b)不同并发量下多节点接口响应速率

图 7　接口响应速率测试

4.2　不同并发量下单节点 CPU 占有率和磁盘 I/O

随机抽取一台主机,以商品搜索为测试点,“棉服”为搜索关键字,并发量设置为 1 000 和 2 000,测试分布式架构下 CPU 占有率和磁盘 I/O 占有率情况,结果如图 8 所示。在并发数为 1 000 的条件下,CPU 占有率达到 57%,属于中等水平,磁盘 I/O 占有率低于 40%,符合高并发下系统平稳运行的标准。

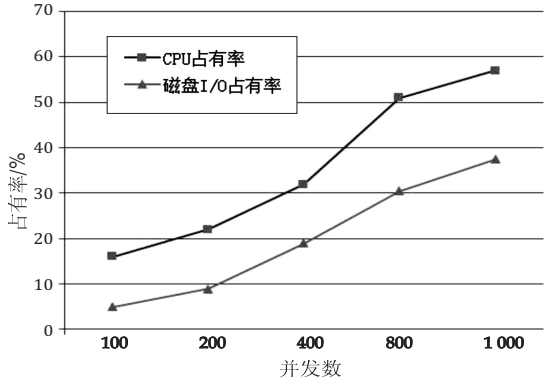


图 8　不同并发量下单节点 CPU 占有率和磁盘 I/O

5　结束语

系统使用分布式软件架构设计并实现了基于 B2C 的新零售购物平台。开发过程中充分考虑了不同业务流程的高并发处理能力,在威胁系统稳定的节点运用了消息队列、缓存数据库、分布式事务等技术,为系统的平稳运行提供了可靠的保障。通过高可用性的几个重要性能指标,测试并验证了在高并发场景下的功能可用性和性能可靠性,为百万级流量系统的软件架构技术选型提供了一种可行的服务化治理方案。

参考文献:

[1] NAMIOT D,SNEPS-SNEPPE M. On micro-services architecture[J]. International Journal of Open Information Technologies,2014,2(9):24-27.

[2] 冯显力,韦　化,韦洪波,等. 含微服务的调度自动化系统分布式实时数据库[J]. 电力系统保护与控制,2018,46(21):138-144.

[3] 范　迪,朱志祥. 一种 Dubbo 框架的授权认证方案[J]. 计算机技术与发展,2017,27(11):115-118.

[4] 苟丽美,张锋叶,林国华. 基于 Zookeeper 的 GIS 集群实现

(下转第 121 页)