上海交通大学硕士学位论文

面向电商平台的秒杀系统设计与实现

硕士研究生：朱丽叶

学　　号：115037910088

导　　师：吴刚副教授

申请学位：工程硕士

学　　科：软件工程

所在单位：电子信息与电气工程学院

答辩日期：2018 年 1 月

授予学位单位：上海交通大学

Dissertation Submitted to Shanghai Jiao Tong University for the Degree of Master

Design and Implementation of Seckill System for Electronic Commerce Platform

Candidate：	Liye Zhu
Student ID:	115037910088
Supervisor：	Associate Prof. Gang Wu
Academic Degree Applied for：	Master of Engineering
Speciality：	Software Engineering
Affiliation：	School of Electronic Information and Electrical Engineering
Date of Defence：	January, 2018
Degree-Conferring-Institution：	Shanghai Jiao Tong University

面向电商平台的秒杀系统设计与实现

摘　要

随着电子商务的发展，秒杀活动作为一种吸引顾客、聚集人气的销售手段，极大的刺激了消费者的购买欲望。在秒杀活动中，秒杀商品要么价钱超低于市场价，要么十分稀缺，一般都会在有限的时间内，以限定的数量被销售。因为这些商品对消费者的诱惑力是巨大的，往往在活动开始的几秒钟内就被售罄。一个支持高并发高可用、可伸缩可拓展的秒杀系统对电商平台而言具有相当的现实意义。

本研究与汇金百货合作，对其已经上线的 O2O 交易平台进行拓展，为了更好的支持秒杀业务，需要额外开发一个电商平台的秒杀系统满足商家和顾客的需求。电商平台秒杀系统需要达到以下要求，第一、秒杀系统需要全面的满足商家和用户的功能性需求和非功能性需求，第二、秒杀系统作为一个极度吸引客户、瞬时流量超高的线上系统，需要具备高并发的能力，同时满足 7*24 小时系统稳定性的需求。第三、秒杀系统需要满足可配置、可拓展的非功能需求。商品秒杀活动具有一定的时限性，且商家总是无法准确预计参与秒杀活动用户总量，秒杀系统需要具备能按需加入或移除系统资源的能力。

本文以上述系统为研究对象，设计并实现了面向电商平台的秒杀系统，主要的研究内容如下：第一、对秒杀系统的功能需求和非功能需求进行全面的分析，第二、设计并实现面向商场管理人员和商家的秒杀活动管理系统，满足管理员和供应商对秒杀活动和秒杀商品进行管理的需求。第三、设计并实现面向消费者的商品秒杀系统。面向消费者的分布式商品秒杀系统的设计与实现主要包含以下内容：系统基于 LVS 配合 nginx 搭建商品秒杀系统的网关层，实现流量的高效分发

并对恶意流量进行限制；系统以 springboot 微服务框架为基础，使用 dubbo 分布式服务调用框架配合 hystrix 容错机制搭建了分布式的业务层的微服务集群，将业务层划分为 API 接入服务、秒杀下单服务、秒杀信息获取服务、订单服务、商品详细信息获取服务，对所有服务的代码流程设计与实现充分考虑了分布式服务调用的幂等性和业务降级问题，使得系统具有较好的并发性、可用性、可拓展性。系统基于 Redis sentinel 方案，设计并搭建秒杀系统缓存高可用集群，对系统进行代码流程设计时考虑了缓存充分利用的问题，提高了系统的抗并发能力。

最后，在以上实现的基础上，对整个系统进行了功能测试和性能测试，测试结果表明系统能较好的满足供应商、管理员和用户的功能需求和性能需求。

关键词：秒杀、高并发、高可用、架构、系统、LVS、dubbo、hystrix

Design and Implementation of Seckill System for Electronic Commerce Platform

ABSTRACT

The rapid development of e-commerce prompted the birth of seckill activity. The goods in seckill activities has a price well below the market price, their quantity and supply time are usually limited. In a seckill activity, a limited number of products will be sold at varying degrees of discount, which brings a huge temptation for customers. Once the seckill activity begins massive users rush in the system to place an order, the discounted products are usually sold out in seconds. In this case, a seckill system with high concurrency and high availability has very practical significance.

This research cooperates with Huijin Department Store to expand its O2O trading platform which is already online. In order to support seckill business better, an additional e-commerce seckill system needs to be developed to meet the needs of sales managers and customers. E-commerce platform seckill system needs to have the following features. First, seckill system needs to fully meet the functional needs and non-functional needs of sales managers and customers. Second, seckill system is an online system which attracts a large quality of customers, demanding high concurrency while meeting the stability requirements of 7 * 24 hours. Third, seckill systems need to meet non-functional requirements like configurability and scalability. Commodity seckill activities have a certain time limit, seckill systems need to have the ability to add or remove system resources on demand.

In view of the above-mentioned system goals, this article designs and implements a seckill system for e-commerce platform. The main research work is as follows: First, this paper does a comprehensive analysis of the

seckill system functional requirements and non-functional requirements. Second, we design and implement a seckill management system to meet the needs of administrators and suppliers to manage seckill activities and seckill items. Third, it design and implement a consumer-oriented distributed item seckill system.

The design and implementation of consumer-oriented distributed commodity seckill system mainly includes the following: The system builds the gateway layer of commodity seckill system based on LVS and nginx to realize the efficient distribution of traffic and the restriction of malicious network traffic; The system is based on the springboot microservice framework , using dubbo distributed service call framework with hystrix fault tolerance mechanism to build a distributed service layer micro service cluster. The system consists of API access service, seckill service, seckill information acquisition service, order service and item details acquisition service.

The design and implementation of all services fully considers the idempotent of service calls and service degradation when sevice call failed, which makes the system have good concurrency, availability and scalability. Based on the Redis Sentinel, the system builds a seckill system cache cluster with high availability. And in the process of system implementation,we fully considers the issues such as making full use of cache and cache breakdown problem to improve the concurrent capability of the system.

Finally, based on the above implementation, the whole system has been tested for functionality and performance. The test results show that the system can meet the functional requirements and performance requirements of suppliers, administrators and users.

KEY WORDS: seckill, high concurrency, high availability, architecture, system, LVS, dubbo, hystrix

目　录

插图目录

表格目录

第一章 绪论

1.1 研究背景

近年来，由于国家对互联网行业大力支持，“互联网+”[1]行动计划的提出，互联网行业日新月异，电子商务行业更是发生了翻天覆地的变化，在短短十几年间，电子商务交易总额扩大了十倍，与传统零售行业 7*12 小时的营业时间相比，电商平台能 7*24 小时全天候提供服务，为人们的生活提供了极大的便利。秒杀活动作为一种快速吸引顾客、增加品牌影响力的销售手段，以低廉的价格，极大的刺激了消费者的购买欲望。双十一购物节[2]的秒杀活动每年都吸引成千上万的消费者参加，交易额高达一千亿，消费者秒杀到心仪的商品的同时，也为电商带来了巨额的收益。

电商举办秒杀活动有许多好处[3]。秒杀活动能快速提升品牌影响力。秒杀活动具有易引发话题、时效性快等特点，电商通过举办秒杀活动能巧妙的利用用户推广能力，快速提升电商品牌的影响力，如果电商定期举办秒杀活动，能极大的宣传电商平台并增大平台的用户粘性。各大电商平台，如淘宝、京东、苏宁易购、国美等，创造出各种各样，如双十一、618 年中大促的购物节等活动，这些活动都会以商品秒杀作为噱头，吸引了大批消费者参加活动的同时，也维持了网站的用户活跃度。秒杀活动能较大的提高电商的销售额。用户在浏览秒杀活动商品时，不可避免的会看看网站中别的商品，无形中增加了电商网站的用户流量，为电商网站带来收益。秒杀活动能适当的减缓商家积压产品负担，电商商家都不可避免地需要解决一些产品积压的问题，在这种情况下，通过举办秒杀活动能有效解决这一难题。

秒杀活动中售卖的商品，由于价格一般远低于市场价格，或者十分稀缺，往往都会被限量、限时销售。因为这些商品对消费者的诱惑力是巨大的，往往在活动开始的几秒钟内就被售罄。因此，秒杀业务与传统的业务相比，需要有较强的并发能力。秒杀系统作为一种商业盈利的手段，需要满足消费者 7*24 小时高可用的需求，即有较高的稳定性要求。秒杀活动与普通活动不同，秒杀活动参与活动的

用户数量往往是不可预期的，且具有较高的时限性，因此秒杀系统需要实现系统模块的灵活配置。秒杀活动一般采用分布式系统进行搭建，参与活动的机器较多，秒杀系统需要一定的可维护性，并且随着电商网站的不断发展，用户流量的不断增长，后期对秒杀系统支持并发数量要求会提高，秒杀系统也需要具备一定的可拓展性。

徐家汇汇金百货是一个具有前瞻性的上海著名的线下零售企业，近来，汇金百货推出了汇金百货 O2O 交易平台[4]后，实现了从传统的零售交易业务场景到支持 O2O 交易业务场景的转变，能较好的运行和维护现有的正常的商品售卖业务，并拥有良好的并发性和可用性。汇金百货为了更好的支持商品秒杀业务，需要对 O2O 交易系统业务进行进一步拓展，针对商家和顾客的需求，设计并实现面向电商平台的秒杀系统。

1.2 研究目标

原有的汇金百货 O2O 交易系统虽然较好的满足了现有的 O2O 交易场景的业务需求，但是为支持秒杀业务场景，需要对原有系统进行新业务的拓展。本文具体的研究目标如下：

1.2.1 满足秒杀的业务需求

秒杀系统需要满足商场管理员和供应商对秒杀系统后台管理的需求，同时满足用户对商品秒杀系统的业务需求。

商场管理人员需要通过秒杀系统对整个秒杀活动进行策划，对参与秒杀活动的供应商和秒杀商品进行管理，对秒杀系统黑名单用户进行管理。供应商需要通过秒杀系统参与秒杀活动，并对参与秒杀活动的商品进行管理。用户需要通过使用秒杀系统查看秒杀活动和秒杀商品信息，进行秒杀下单、商品支付、取消订单等操作。

1.2.2 支持高并发、高可用

秒杀系统需要满足消费者对系统高并发、高可用的需求。由于秒杀商品拥有巨大的吸引力，面向电商平台的秒杀系统对系统的并发性和系统的可用性要求非常高。在执行秒杀操作时，可用性不好的秒杀系统候经常会出现服务不可用、系

统卡死、宕机等情况，并发性不好的秒杀系统的响应速度特别慢。因此，为满足消费者的日益膨胀的秒杀需求，面向电商平台的秒杀系统将向广大消费者提供支持高并发、高可用的服务。

秒杀系统的高并发、高可用性将表现在三个方面：第一，在较高的并发条件下，用户的平均响应时间小于 5 秒，系统的任意服务不存在单点故障，保证系统在 24*7 小时运行的情况下，在一次活动内的正常运行率大于 99%；第二，秒杀系统流量峰值过大，导致秒杀服务不可用时，系统也应当正常运行，不影响系统进行其他服务，系统应当支持最高每秒 10000 次的并发访问量。第三，系统具备一定的安全性，能限制一些异常流量攻击，保持高可用的状态。

1.2.3 实现可配置、可拓展的分布式服务

秒杀需要满足系统可配置、可拓展的需求。由于秒杀活动具有一定的时限性，通常在某个特定的时间，商场进行大规模秒杀活动，而在活动结束后，秒杀业务将很少使用，为了让系统更加灵活且充分使用系统资源，秒杀系统需要满足可配置的特性，具体表现为：第一、系统在不进行秒杀活动的时候，可以进行秒杀系统模块的简单删除，快速安装其他服务，实现机器的有效利用。第二、当秒杀系统服务机器不足或服务机器空闲时，可以具体根据业务的情况进行服务机器的调整，比如由于秒杀下单服务部署不足导致用户秒杀下单成功率较低的时候，可以通过增加部署秒杀下单服务的机器提高下单成功率；当秒杀商品详情获取服务访问数据库次数较少、造成机器空闲时，则可适当减少部署商品详情获取服务的机器。

汇金百货的面向电商平台秒杀系统还处于发展初期，有增大的业务增长空间和功能可拓展性。可拓展性具体表现为，第一、系统服务功能定义清晰，代码规范，文档有条理，对现有服务增加或修改业务逻辑时，能较好地进行改进。第二、系统耦合度应该降低，当系统需要添加新服务时，尽量减少对原有代码的修改和影响。

1.3 主要工作

论文对当前主流的面向电商平台秒杀系统的技术要点进行分析和研究，并结

合汇金百货对于秒杀系统的功能需求和非功能需求，设计了一套完整的面向电商平台秒杀系统的解决方案。本文主要的研究内容和取得的成果包括：

第一，对汇金百货的面向电商平台秒杀系统的需求进行详细分析。需求分析包含对面向管理员和供应商的秒杀系统后台进行功能需求分析，非功能需求分析，对面向消费者的秒杀系统进行功能分析和非功能分析。

第二，对面向商场管理人员和供应商的秒杀系统后台进行架构设计、详细设计与实现。架构设计包含网关层的架构设计、业务层的架构设计；详细设计与实现是指对秒杀系统后台功能模块进行具体设计与实现。

第三，对面向消费者的商品秒杀系统进行架构设计、详细设计与实现。面向消费者的商品秒杀系统的架构设计包含对业务层的架构设计、网关层的架构设计和缓存集群高可用方案的设计。在业务层架构设计中，系统使用 springboot 微服务框架，配合 dubbo+Zookeeper 远程服务调用框架，对业务层内部的服务进行功能边界划分，实现了业务层服务的自动注册与发现，实现了秒杀系统业务层的可配置、可拓展。在网关层的架构设计中，系统使用 LVS 配合 nginx 进行系统网关层的设计，实现了流量的负载均衡、网关层的高可用和恶意流量的限制。在 redis 缓存集群的架构设计中，系统使用 redis 客户端分片方案，配合 sentinel 高可用方案进行系统分布式缓存的设计，提高秒杀系统缓存层的稳定性和可用性。

面向消费者的商品秒杀系统详细设计与实现包含秒杀信息获取服务、秒杀下单服务、订单服务、商品详细信息获取服务、API 接入服务、库存同步服务和 nginx 限流方案。在详细实现的过程中，充分考虑了分布式服务的无状态性，服务调用的幂等性，缓存的充分利用，缓存击穿问题优化等因素。另外，为了使秒杀系统具备一定的防恶意攻击的能力，系统基于 nginx 负载均衡器使用 lua 脚本对网关层进行限流控制，加强了系统的安全性。

第四，对面向电商平台的秒杀系统进行测试环境的搭建后，进行了全面的功能测试和性能测试，并对测试结果进行分析。本文通过设计测试用例，对秒杀系统进行全面的功能测试，通过使用压测工具，对秒杀系统进行全面的性能测试，最后对系统的测试结果进行总体分析。

1.4 论文组织结构

本文总共七个章节，每篇章结构及其内容具体如下：

第一章是绪论，介绍了面向电商平台秒杀系统的研究背景，并引出了本文所做出的研究内容以及成果。

第二章是相关技术研究，深入探讨了面向电商平台的秒杀系统的技术要点，并对关键技术进行阐述与分析，为接下来的秒杀系统设计与实现奠定基础。

第三章是电商平台秒杀系统的需求分析，对面向电商平台的秒杀系统进行功能需求分析、非功能需求分析，描述了电商平台秒杀系统的整体需求。

第四章是秒杀系统后台的设计与实现，对面向商场管理员和供应商的秒杀系统后台进行详细的设计与实现。

第五章是面向消费者的秒杀系统设计与实现，对面向消费者的秒杀系统进行总体的架构设计，并分模块进行详细的设计与实现。

第六章是电商平台秒杀系统测试与分析，对电商平台秒杀系统进行功能测试和性能测试，并对电商平台秒杀系统进行分析与评估。

第七章是总结与展望，对全文的工作进行了总结和展望，并提出一些未来系统优化可能的研究方向。

第二章 相关技术研究

2.1 负载均衡器技术研究与分析

目前主流的负载均衡器技术有，NetScaler[6]、F5[7]、LVS[8]、Nginx[9]、Haproxy[10]等，这些负载均衡技术可以被粗略的划分为硬件类负载均衡技术和软件类负载均衡技术，其中 NetScaler[6]和 F5[7]属于硬件类负载均衡技术，这些负载均衡技术需要大量的资金支持，所提供的负载均衡服务具有吞吐量大、可用性强、稳定性强的优点。软件类负载均衡技术，如 Nginx[9]、Haproxy[10]、LVS[8]，这些负载均衡器对硬件设备没有要求，而且所需要使用的软件均被开源，软件可以针对实际场景进行二次开发，有良好的可拓展性，对电商平台而言，使用这一类的软件负载均衡器可以极大的降低成本。由于汇金百货的电商平台缺乏大量资金和硬件的支持，本文只对软件负载均衡器进行对比。

(1) nginx[9]：nginx 工作在网络七层模型中的第七层，针对 HTTP 请求进行分流，nginx 具有配置安装简单，测试清晰方便，在高负载情况下有较好的稳定性等优点。nginx 正则规则较为灵活，且具有 cache 功能，能进行图片 web 服务器和静态网页服务器的搭建。Nginx 还能与 lua 配合，进行系统的流量控制，Nginx 由于仅支持 HTTP 和 Email 协议[11]，适用范围较窄。

(2) Haproxy：Haproxy 和 nginx 类似，不同的是他支持对 session 的保持和 cookie 的引导[10]，Haproxy 对负载均衡的算法支持较全面，由于不具备 cache 功能，不能进行 web 服务器的搭建，Haproxy 的正则规则也不如 nginx 灵活。

(3) LVS[8]：LVS 工作在网络七层模型中的第四层，当使用 LVS/DR 模式时不产生网络流量，具有稳定性强、性能强的优点，同时由于它只进行分发请求，不对流量本分进行操作的特性，LVS 的应用范围非常广泛。

为了实现负载均衡器的高可用性，负载均衡器配合 Keepalived[12]高可用方案在互联网负责均衡器集群的搭建环境中被广泛被使用。

2.2 微服务架构技术

微服务是一种架构风格，2014 年 3 月，Martin Fowler 写的一篇文章“Microservices”[13]首次提出了微服务的概念。微服务可以将应用服务分解成各个微服务的集合，每个微服务模块可被单独部署，且独立承担一件业务功能，并能较好的专注的完成这个业务功能[15]。整体式架构如图 2-1 左所示，它将系统的所有业务放到同一个进程中，并通过复制整个应用的方式实现拓展。微服务结构如图 2-1 右所示，不同的服务可以被部署在不同的进程中，并可以依据系统的需求进行系统的拓展。

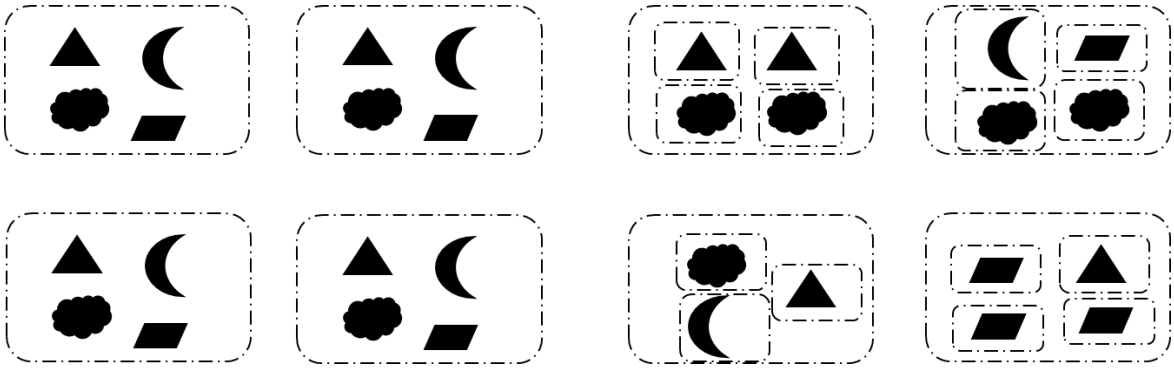


图 2- 1 整体式架构和微服务架构对比图[14]

Fig. 2-1 Comparison of monolithic architecture and microservices architecture

微服务有许多好处，依据文献[15]所述，首先，每个服务都有自己定义清楚的功能边界，易于理解和开发，提高了系统的可维护性。不同的微服务可以同时由不同的团队进行开发，提高了开发效率，加快了系统的迭代速度。微服务同时也降低了系统架构的耦合性，使得系统更加容易拓展。微服务还可以通过灵活的配置各个服务的机器数量，提高机器的使用率。Springboot 技术是目前主流的搭建微服务的技术之一[16]。

2.3 Dubbo 远程服务调用技术

根据文献[17]所述，Dubbo 远程服务调用技术是一个具有较高性能的，基于 java 的，由阿里巴巴开源的远程服务调用框架。Dubbo 由于其开源的特性，可定制性较强，在国内互联网行业中非常受欢迎。远程服务调用框架具有对远程服务进行维护、管理、

调用等功能。远程服务调用的用途在于首先定义一个服务，包括定义服务调用的方法名称，服务的参数和服务的返回类型[18]，被调用的服务端，需要实现这个服务，并使用远程服务调用框架监听客户端的请求，客户端根据需要进行远程服务调用。

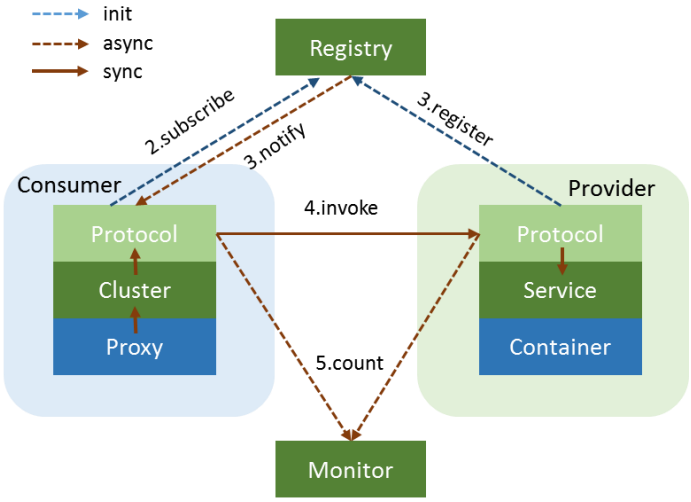


图 2- 2 Dubbo 架构图[17]

Fig. 2-2 Dubbo architecture

Dubbo 使用 Netty 作为底层的网络通信服务框架，有效的提高了分布式服务框架中每台机器支持的连接数量，提升了集群的拓展能力和对高并发的支持。Netty[19]是一种基于 NIO 的网络通信服务框架，具有易开发、高性能、可拓展的网络通信框架，简单来说就是简化了 TCP、UDP 服务器的编程。NIO 是一种事件驱动的、非阻塞的 I/O 模型[21]。网络 I/O 的代价是昂贵的，与 CPU 的处理性能有显著的差距[20]，为了提升 I/O 的性能，在 I/O 过程中提高 CPU 的利用率，出现了多种 I/O 模型，比较有代表的有以下两种：

(1) 阻塞式 I/O 模型，在 JDK1.4 版本之前，建立网络连接时通常采用这种模式[40]。在这种模式中，服务端需要给每个网络 I/O 连接分配一个线程，然后线程需要一直运行直到整个网络 I/O 执行完成，客户端在发送请求之后，需要先咨询服务端是否有线程剩余，如果服务端没有线程，客户端会一直等待或者直接被拒绝，如果服务端有线程，就建立连接[40]。

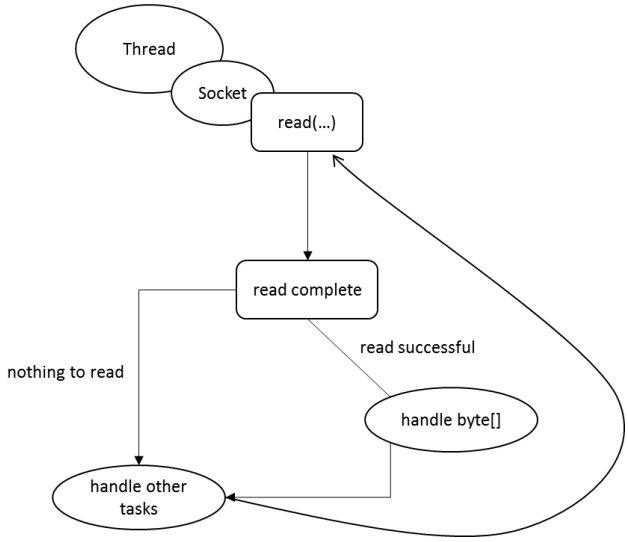


图 2- 3 阻塞式 I/O 模型[19]

Fig. 2-3 Blocking I / O model diagram

(2) 非阻塞式 I/O 模型。同步非阻塞 I/O 模型主要解决了同步阻塞 I/O 模型的并发问题，在使用同步阻塞 I/O 模型时，如果服务器需要同时进行多个客户端的请求，服务器需要为每一个客户端连接创建一个线程单独处理，由于线程需要分配一定的内存空间，并且在没有请求时，线程等待的时间过长，造成了资源的浪费，限制了服务端与客户端的连接数量[40]。NIO 对此进行了改进，所有的连接会被注册到多路复用器上，当轮询的线程发现连接上有请求，才开始进行处理，这样就能用一个线程为多个连接服务，极大的节约了系统资源[19]。

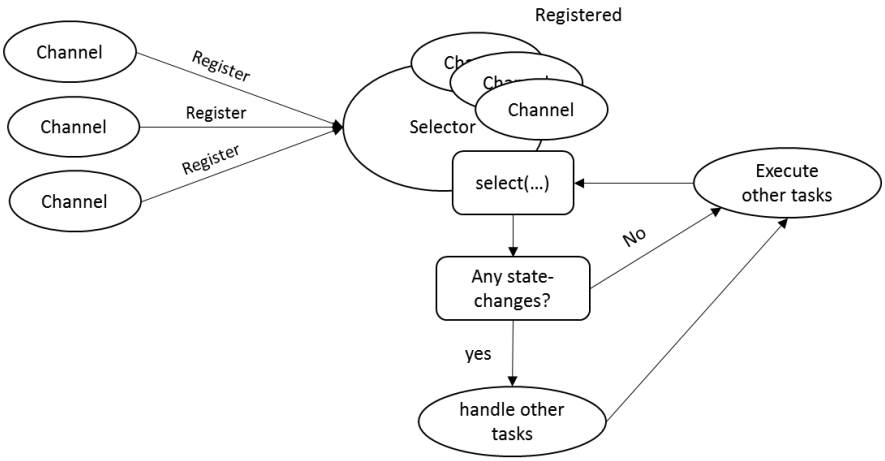


图 2- 4 非阻塞式 I/O 模型图[19]

Fig. 2-4 Non-blocking I / O model diagram

2.4 Hytrix 容错机制

Hytrix 是 Netflix 发布的、开源的、提升分布式系统容错能力的一个类库[22]。Hystrix 容错机制能有效的提高系统的自我修复能力[23]，降低业务对单一服务的依赖。下面将对这两种技术进行研究和分析。依据文献[24]所述，复杂的分布式系统通常都具有很多依赖，假设一个应用依赖 30 个服务，在某一时间点，每个服务有 99.9%的概率是正常运行的，99.9%的 30 次方是 99.7%，即 0.3%概率有一个服务不可用，假设在这个时间点有 1 千万个请求，则有 3 万个请求的服务是出错的，事实上，在现实情况中，这个数字甚至更糟糕。

当有一个服务不可用时，如果系统没有任何容错措施，在每秒 50 个请求的并发的情况下，所有依赖 F 的请求都会被阻塞，所有的请求机器在几秒之内将迅速达到饱和的状态[23]，如图 2-5 所示，甚至可能因为业务之间的相互依赖，导致所有依赖 F 的上层业务都不可用，即滚雪球效应。

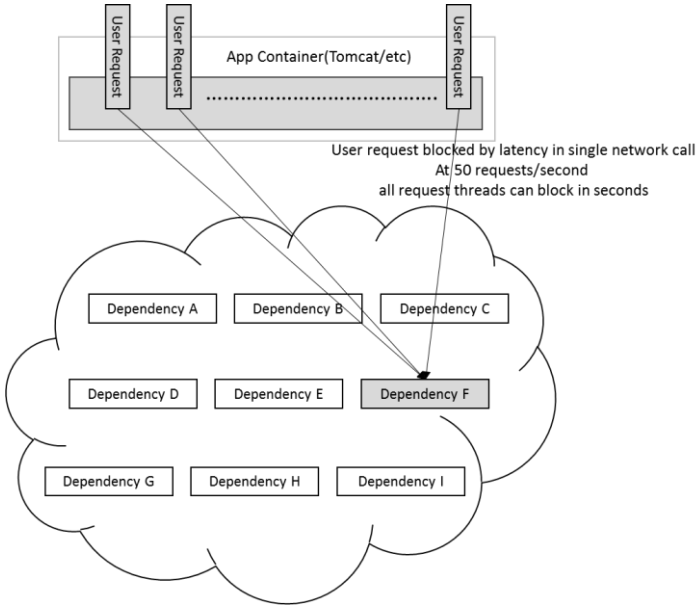


图 2- 5 系统阻塞示意图[22]

Fig. 2-5 System block diagram

Hytrix 为提高系统的容错能力，对系统的依赖和延迟进行控制[24]，并允许系统进行快速失败和迅速修复，在服务不可用时，可以回退或者降级。下图是 Hystrix 的处理流程图，Hytrix 使用舱壁模式（bulkhead pattern）来进行依赖的隔离[22]，使用独立

的线程池或信号来对并发请求进行约束和限制，如果分配的线程池已满或者分配的信号量已用完，则拒绝服务，对于被拒绝的服务，调用 getFallback()/resumeWithFallback()函数进行降级处理。如果还有线程/信号量剩余，则执行业务，如果业务执行失败或者业务超时，也将调用 getFallback()/resumeWithFallback()函数进行降级处理。Hystrix 还提供熔断器组件[22]，可以设置为自动运行或者手动调用，熔断器开始运行时将停止依赖一段时间。

2.5 Redis 集群技术对比

Redis 在互联网行业缓存中应用广泛，主要用来缓存热点数据，提高数据的访问速度，从而提高服务的响应速度和并发量[25]。随着数据量的不断增长，Redis 单实例的处理能力有限，就需要用到 Redis 集群方案。目前有三种主流的 Redis 集群方案，包括客户端分片、Twemproxy 代理 Redis[26]和 Redis Cluster[27]。

（1）客户端分片

客户端分片技术不使用任何第三方插件，由客户端直接进行分片。如图 2-6 所示，在使用这种分片技术时，由于客户端和 Redis 直接连接，性能会比较好。这种方式对客户端代码有一定的侵入性，导致客户端使用会较复杂，不能快速的进行扩容/缩容，扩容/缩容时必须手动在客户端重新分片，对缓存数据存储比较庞大的业务而言不太适合，容错能力较差，出现故障时不能自动转移，搭配 redis 高可用集群方案使用可以提升容错能力。

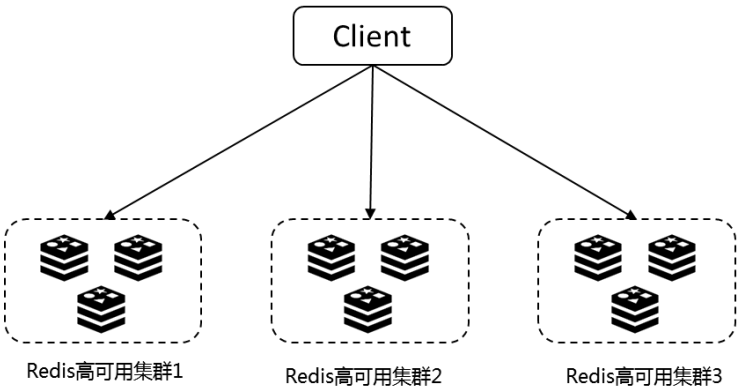


图 2- 6 Redis 集群客户端分片架构示意图

Fig. 2-6 Diagram of Redis cluster client fragmentation architecture

（2）Twemproxy 代理 Redis

Twemproxy 是 Twitter 开源的一个兼容 redis 和 memcache 的代理服务器，其优点是客户端使用方式简单，Twemproxy 对客户端屏蔽了 Twemproxy 后面的 redis 实例，客户端进行操作时与操作单机的 Redis 类似。如图 2-7 所示，Twemproxy 代理能实现 redis 连接共享，降低了 redis 的连接数量，提高了系统的资源利用率，并能即时的让无效的 redis 实例从代理转发机制中移除[26]。这种方式的缺点是由于使用代理，多了一次网络消耗，导致读写 redis 性能受损。Twemproxy 代理方式在进行 Redis 扩容/缩容的时候，部分 redis 数据会失效，不能平滑的扩容缩容，容错能力较差，出现故障不能自动转移，且使用 Twemproxy 代理时，不能使用 redis 事务操作[29]。

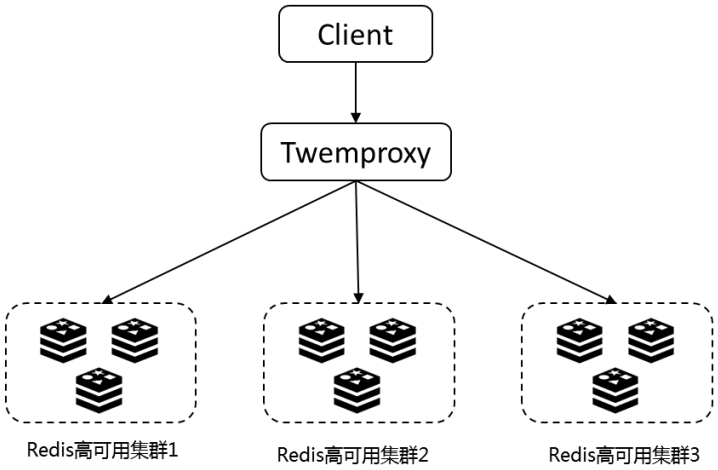


图 2- 7 Twemproxy 代理 Redis 集群架构示意图

Fig. 2-7 Twemproxy agent Redis cluster architecture diagram

（3）Redis Cluster

Redis Cluster 是 redis 官方提出的 redis 集群方案，与 Twemproxy 代理方案不同，如图 2-8 所示，它是一种无中心节点的结构。根据文献[27]所述，Redis Cluster 方案没有使用哈希进行分片，而是引入了哈希槽的概念，共有 16384 个哈希槽，Redis 操作的每个键值通过 CRC16 校验后，对 16384 取模来决定属于哪个哈希槽，每个 redis 集群中的节点负责对一部分哈希槽进行管理，数据按照哈希槽均匀的分布在 redis 集群的节点上。当集群新增节点时，Redis Cluster 只需把一部分哈希槽分配到新增节点上，并不会停止原有的节点继续提供服务[27]。为了让在部分节点失败或者大部分节点无法通信的情况下，集群依旧可用，集群使用复制模型，每个 redis 主节点都有 N-1 个

从节点，当主节点宕机后，节点之间将通过 gossip 协议交换状态信息，将从从节点选择出一个，完成从节点到主节点的提升，这种方式提高了系统的可用性，降低了运维的成本，如果所有主节点和从节点都不可用，整个集群还是会因为找不到 redis 这个范围的槽而变得不可用[27]。redis cluster 节点的状态信息需要节点之间定期交换并更新，Redis Cluster 方案增加了系统资源的消耗，redis 集群不支持多个键值的命令，不支持 redis 事务操作，因为这需要在不同的节点之间进行数据的移动，在负载量高、数据量大的情况下，将导致系统执行极其缓慢[30]。目前 redis cluster 也不能智能的进行自动扩容、缩容，扩容、缩容操作仍需要自动化工具的支持。

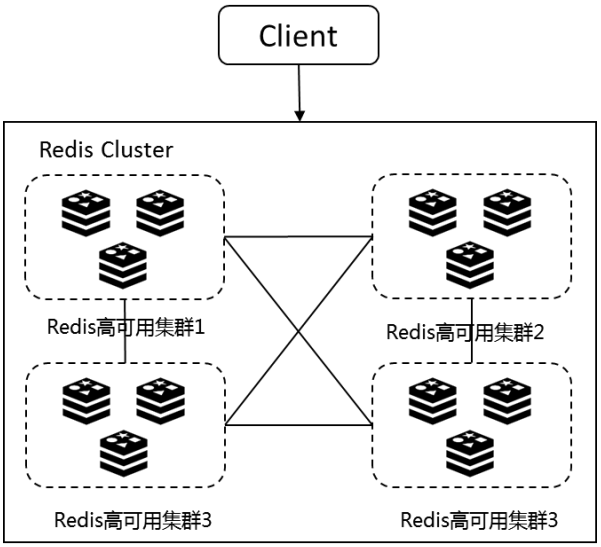


图 2- 8 Redis Cluster 集群高可用架构示意图

Fig. 2-8 Redis Cluster cluster high availability architecture diagram

2. 6 本章小结

本章按照网络数据流量的走向，对秒杀系统的不同模块进行相关技术探讨。首先对网关层的负载均衡技术进行了技术对比，之后对微服务架构进行了架构描述，接着对远程服务调用技术进行了研究分析，然后对容错机制进行了模型探讨，最后对缓存集群技术进行了理论阐述。本章相关技术的研究工作为后续的系统设计、开发奠定技术基础。

第三章 面向电商平台秒杀系统的需求分析

近十年来，伴随着互联网的飞速发展和普及，电子商务交易总额扩大了十倍。新兴的电子商务行业与传统零售行业相比，电商平台能 7*24 小时全天候、不间断提供服务，为人们的生活提供了极大的便利的同时，也改变了人们的生活方式[31]。秒杀活动作为一种快速汇聚人气、宣传电商品牌、增强用户粘性的销售手段，以低廉的价格，极大的刺激了消费者的购买欲望。

本文以汇金百货的秒杀系统为对象，进行课题研究。汇金百货属于上海徐家汇商城股份有限公司，于 98 年开业至今，定位为中高档百货，是上海有名的具有前瞻性的线下零售企业之一。汇金百货以其熟识的购物环境有口皆碑的品牌文化吸引了大批顾客的购买，其线下零售交易系统每年的销售额高达 30 亿[32]。汇金百货自推出了汇金百货 O2O 交易平台[4]后，完成 O2O(online to offline)转型后，为消费者提供了更好的购物体验，大幅提升了销售额度，为了更好的支持商品秒杀业务，需要额外开发一个面向电商平台的秒杀系统满足商家和顾客的需求。

秒杀系统可被分为秒杀系统后台和面向消费者的秒杀系统，秒杀系统后台主要面向秒杀活动后台的管理人员和供应商，面向消费者的秒杀系统主要面向消费者用户群体。

3.1 秒杀系统后台功能需求

秒杀系统后台主要参与人员有管理员和品牌供应商，可按照参与人员进行秒杀系统后台功能的划分。

3.1.1 管理员的功能需求分析

在秒杀系统后台中，管理员的用例如图 3-1 所示。商场管理员可以对秒杀活动、秒杀活动商品、秒杀活动商家、秒杀黑名单进行管理。系统管理员可以对商场管理员进行用户管理。

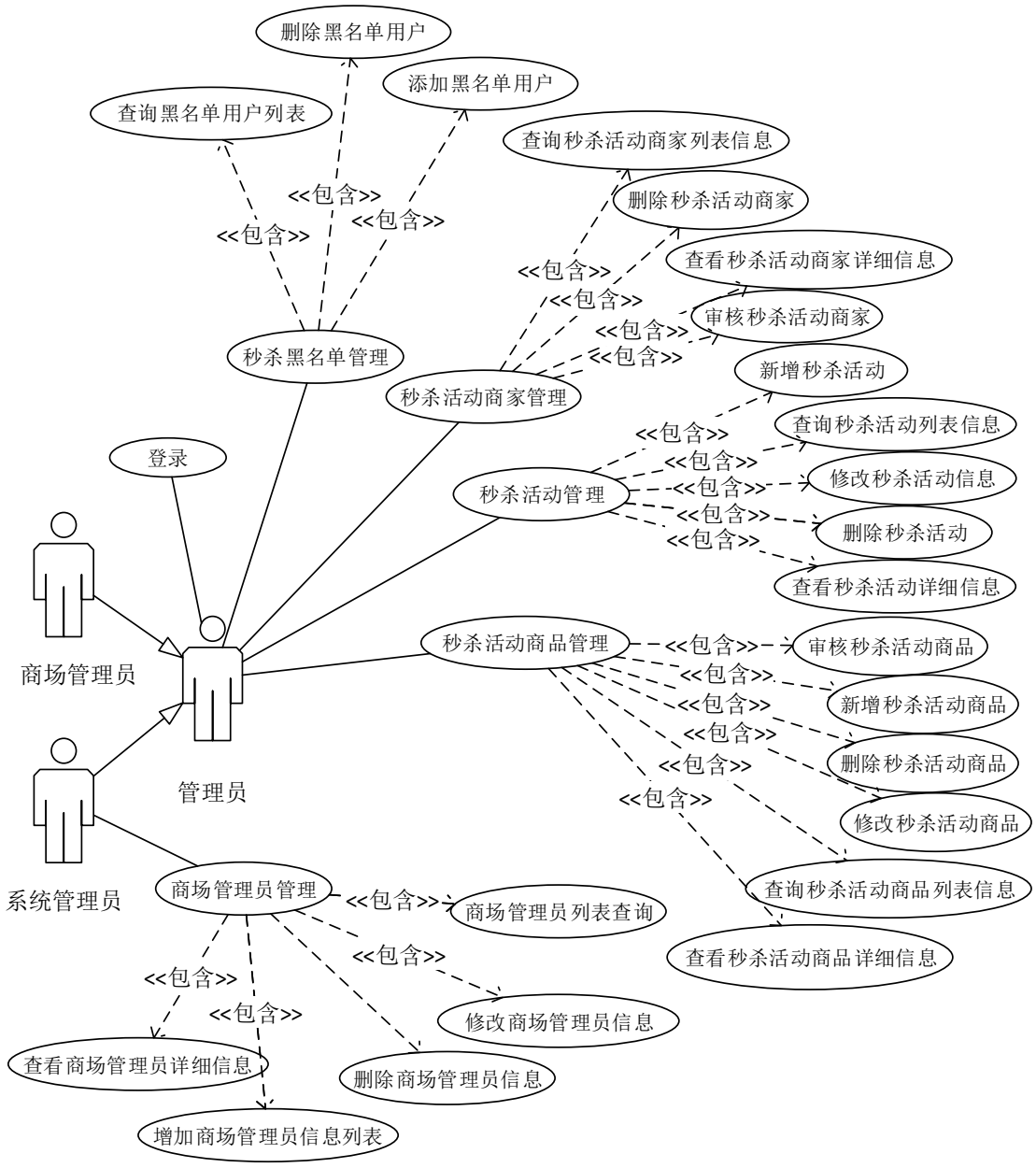


图 3- 1 秒杀系统后台-管理员用例图

Fig. 3-1 Seckill system's backend-use case of manager

秒杀活动的管理用例应该支持商场管理员进行新增秒杀活动，查询秒杀活动，修改秒杀活动，删除秒杀活动和查看秒杀活动详细信息的操作。

秒杀活动商品管理用例应该支持商场管理员对秒杀活动商品列表进行条件查询、商品审核、商品删除、商品添加、商品修改、商品详细信息查看的操作。

秒杀活动商家管理用例应该支持商场管理员对秒杀活动商家列表进行条件查询，秒杀活动商家详细信息查看，删除秒杀活动商家和审核秒杀活动商家。

秒杀活动黑名单管理用例应该支持商场管理员添加黑名单用户，查询黑名单用户和删除黑名单用户操作。

3.1.2 品牌供应商功能需求分析

在秒杀系统后台中，品牌供应商的用例如图 3-2 所示。品牌供应商可以查看秒杀活动信息、对秒杀活动进行申请和对秒杀活动商品进行管理。

秒杀活动信息查看操作包含查询秒杀活动列表操作和查看秒杀活动的详细信息操作。

秒杀活动商品管理用例包含增加、查看、删除秒杀活动商品信息，对秒杀活动商品提交审核和对秒杀活动商品列表信息进行查询。

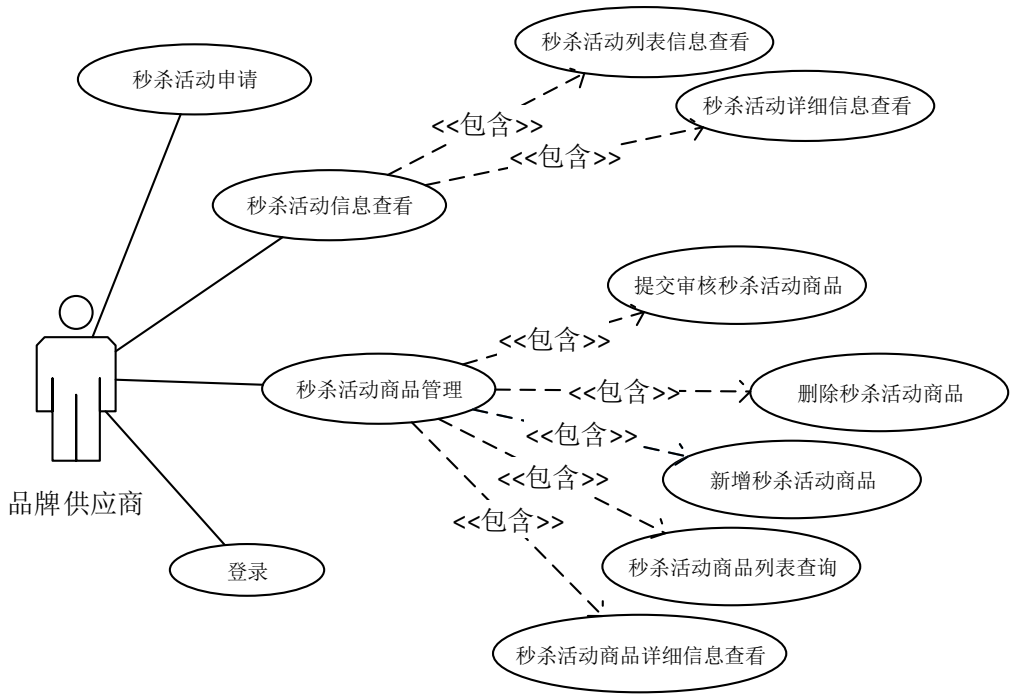


图 3- 2 秒杀系统后台-品牌供应商用例图

Fig. 3-2 Seckill system’s backend-use case of suppliers

3.2 秒杀系统后台非功能需求

秒杀系统后台仅供商场内部人员使用，故系统具有用户量不大、访问量不高的特点，因而系统主要考虑非功能需求有：

(1) 高效性：管理员和供应商需要可以快速的对秒杀信息进行发布、查看、删除、修改和审核等操作。

(2) 安全性：管理员和供应商用户角色不同，针对不同的秒杀系统后台的用户，拥有的后台权限不同，能查看的秒杀商品信息不同，系统需要针对不同用户的不同权限进行隔离。

(3) 可用性：秒杀系统后台需要有用户友好的界面，使用户操作简单、便利，为用户提供较好的用户体验，同时由于秒杀商品管理系统服务于 7*24 小时高可用的商品秒杀系统，所以该系统需要在家用的网络环境下，保证 24 小时的稳定性，能够平稳的进行用户交互，满足每个用户的响应时间不超过 2s 的需求。

3.3 面向消费者的秒杀系统功能需求

秒杀系统主要针对参与秒杀活动的用户，用例图如图 3-3 所示：

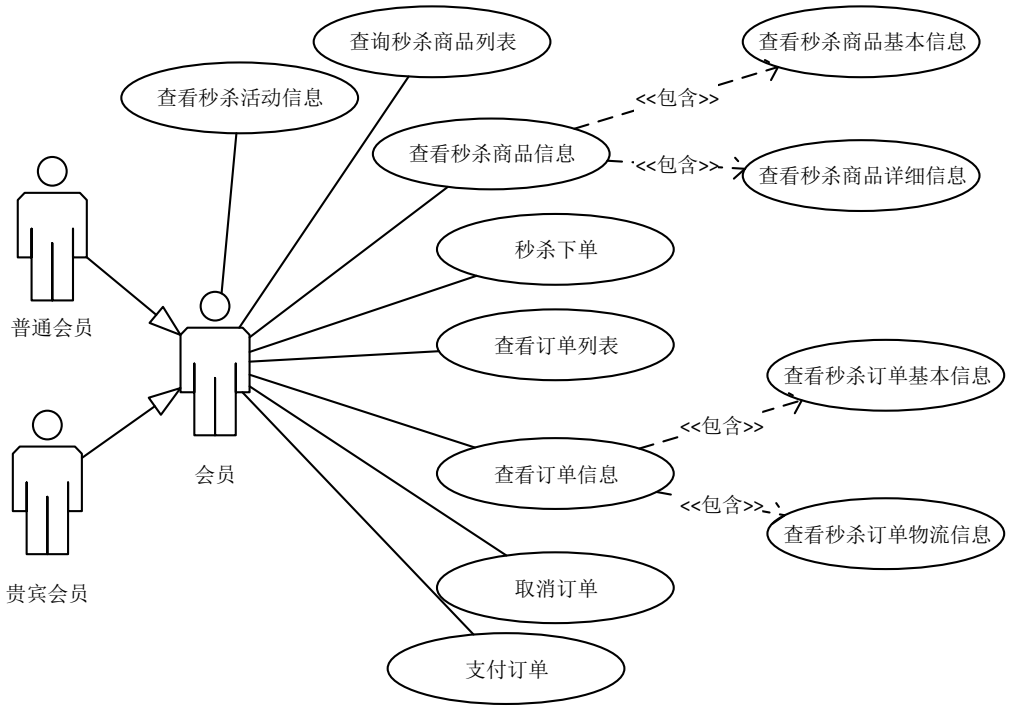


图 3- 3 秒杀系统-会员用例图

Fig. 3-3 Seckill system-use case of customers

(1) 用户查看秒杀活动信息：秒杀系统需要支持用户进行秒杀活动信息的查询。秒杀活动信息包含秒杀活动开始时间、结束时间、秒杀活动规则等信息。

(2) 用户查询秒杀商品列表：秒杀系统需要对用户进行秒杀商品列表的展示。秒杀商品列表包括参与该秒杀活动的所有商品。

(3) 用户查看秒杀商品信息：秒杀系统需要可以对商品的所有信息进行获取，包含商品的基本信息，即标题、图片、颜色尺码、商品型号、参数、价格、库存等信息和商品详细信息，包含商家提供的商品介绍图片信息、商品介绍内容信息等。

(4) 用户进行秒杀下单操作：秒杀系统需要支持满足秒杀活动要求的用户进行秒杀下单操作。如果商品数量充足，用户将对商品秒杀成功，商品数量减一，用户成功下单。如果商品数量不足，秒杀结束，用户将秒杀失败。如果用户因为网络原因或者系统并发原因秒杀失败，而此时商品数量依然充足，用户可以重新进行秒杀，直至秒杀活动结束。针对每件商品，每个用户有秒杀数量限制，如果用户超出了同一件商品秒杀的数量限制，秒杀系统需要返回秒杀商品数量已经达到上限。

(5) 用户查询订单：秒杀系统需要在返回用户成功秒杀的通知后，能在订单列

表中查找到已秒杀的商品订单。秒杀系统支持用户对用户本人所有订单信息包含普通订单信息和秒杀订单信息按照时间进行查询和检索。

(6) 用户查看订单信息：秒杀系统需要支持用户查看订单的详细信息，订单信息包含订单的基本信息和订单的物流信息，订单的基本信息包含下单时间，商品编号等信息。

(7) 用户取消订单：秒杀系统需要支持用户对秒杀订单的取消，用户取消订单后，相应的商品库存需要加一，并对用户订单标记为取消状态。

(8) 用户支付订单：秒杀系统需要支持用户对订单进行支付，支付完成后订单状态信息需要变为已支付。

3.4 面向消费者的秒杀系统非功能需求

近两年，汇金百货积极进行 O2O 商业模式的转型，实现线上电商和线下传统零售行业的结合，汇金百货电商平台发展飞快，汇金百货秒杀系统需要关注的非功能性需求有：

(1) 安全性：由于秒杀商品要么价钱十分低廉，要么十分稀缺，具有很强的商用价值。当今市场上爬虫软件，抢购软件盛行，秒杀系统需要防止爬虫的恶意价格爬取，还需要尽量防范抢购软件造成的 DDos 攻击。

(2) 高并发支持：面向消费者的秒杀系统与普通系统不同，由于秒杀商品本身价格低廉、极为稀缺的特性，秒杀系统在秒杀开始的一瞬间，涌入巨大的流量，秒杀系统需要有较高的支持突发流量的能力。经过针对汇金百货的实际用户量考察，汇金百货的秒杀系统需要支持在秒杀开始时的第 1 秒内，满足 10000 秒杀请求的瞬时并发量。

(3) 可用性：所有的电商平台，淘宝、京东、国美苏宁易购都因 7*24 小时全天候服务的特性，为用户带来了良好的购物体验，汇金百货电商平台秒杀系统同样需要在秒杀活动期间提供 7*24 全天候的购物支持，秒杀系统需要有较好的稳定性。据对汇金百货的实际考察，汇金百货商品秒杀系统需要在支持 10000/s 突发请求的前提下，在任一时间点，99%的概率系统可以正常运行，并且满足每个用户的平均响应时间不超过 5s 的需求。

(4) 可维护性：由于系统后期缺乏人力和资金的支持，汇金百货秒杀系统的耦

合性应该尽量降低，日志尽量丰富，可以对秒杀系统进行运维界面的搭建，达到系统出现故障后，方便排查的目的。

(5) 可拓展性：首先与汇金百货普通商品购买业务不同，汇金百货秒杀业务具有一定的时限性，秒杀系统需要可以具备简单配置、容易删除替换的能力，同时由于秒杀活动参与的用户量巨大、用户参与数量的不确定性，汇金百货秒杀系统必须满足可以动态的针对不同的业务进行机器的横向拓展，或者横向收缩。基于对汇金百货秒杀系统未来业务升级可能性的考虑，秒杀系统还应该尽量降低系统的耦合性，使得系统能更加方便地进行新业务模块的发展。

(6) 用户友好：汇金百货秒杀系统的用户是汇金百货零售商品的潜在消费者或汇金百货的高级会员，因此秒杀系统需要有用户友好的界面设计，使用户执行交互操作时简单、便利，为用户提供较好的用户体验。

(7) 系统开发、搭建成本低：由于汇金百货电商平台发展的时间较短，其秒杀系统还未获取较大的收益，与传统的电商秒杀系统相比，缺乏大型中间件、大型服务器等基础设施、技术沉淀和积累、人力和资金的投入。汇金百货秒杀系统在保证并发的前提下，注意开发成本和运维成本的控制。

3.5 本章小结

本章首先对面向电商平台的秒杀系统的需求进行阐述、归纳和总结，包含对面向管理员和供应商的秒杀系统后台的功能需求和非功能需求进行分析，以及对面向消费者的秒杀系统功能需求和非功能需求进行分析，为接下来的系统设计和实现奠定基础。

第四章 秒杀系统后台的设计与实现

本章将首先对秒杀系统后台进行整体的架构设计，然后再按照架构设计的模块，对秒杀系统后台进行详细的设计与实现。

4.1 秒杀系统后台架构设计

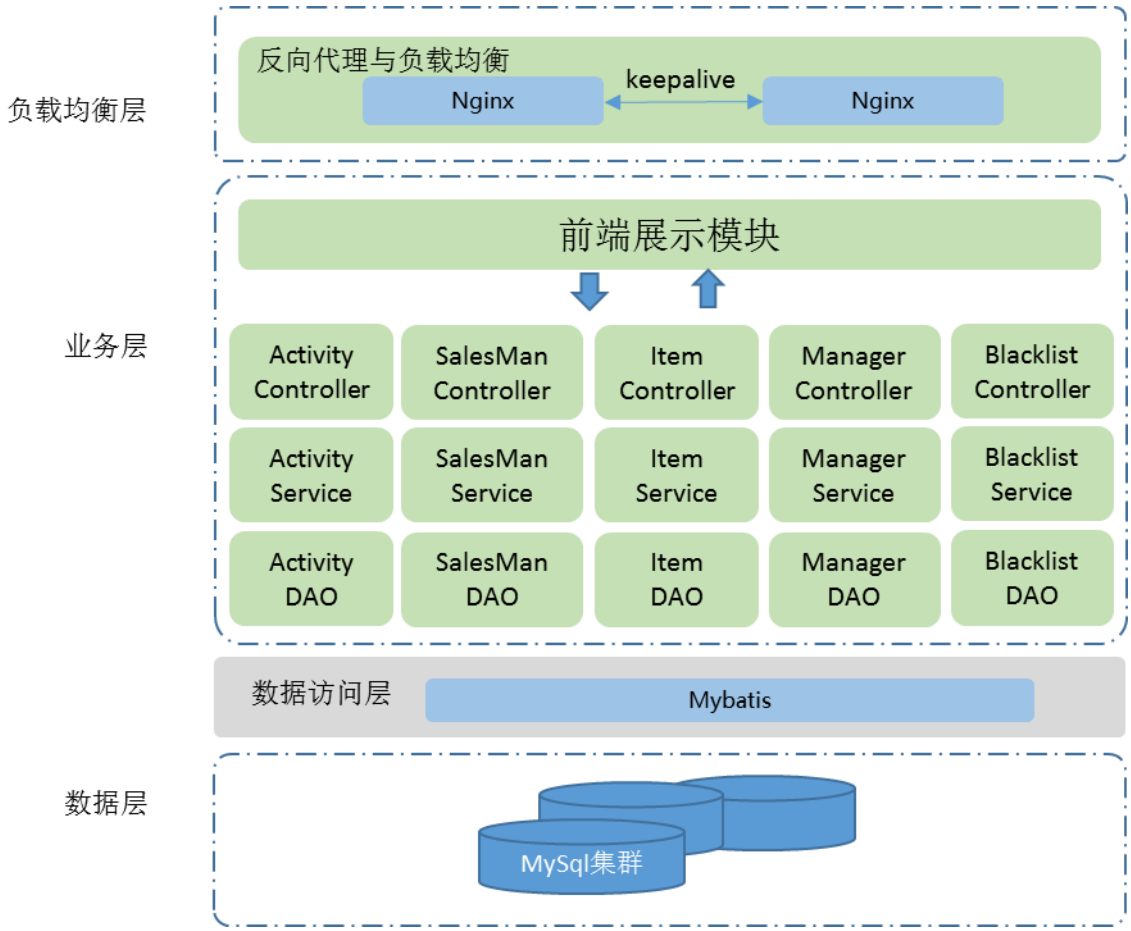


图 4-1 秒杀系统后台架构图

Fig. 4-1 Structure of seckill system's backend

秒杀系统后台架构设计图整体如图 4-1 所示，秒杀活动后台的网关层使用 nginx 配合 keepalive 方案[35]进行流量负载均衡和反向代理，保证了后台系统的高

可用性。在系统的业务层，Controller 类使用 springmvc 技术[33]对页面进行 restful 请求[34]的响应，Service 类即系统服务类，使用 spring 进行业务类的整体管理，依赖注入，Dao 接口类使用 mybatis 和数据库层进行通信。Controller 类通过首先通过 Spring mvc 接收前端请求、解析前端请求参数后，通过 spring 自动注入调用相应的 Service，Service 通过自动注入调用 Dao 层的接口获取数据并对数据进行业务逻辑处理返回给 Controller 类，Mybatis 的 Mapper 会对 Dao 层的接口进行数据库调用的底层实现[36]。

秒杀系统业务层可按照业务需求分为以上五个功能模块，五个模块分别为：秒杀活动模块、秒杀供应商模块、秒杀商品模块、秒杀管理员模块和秒杀黑名单模块，五个模块对页面展示模块提供接口，完成系统功能。秒杀系统后台详细设计与实现。

4.2 秒杀系统后台详细设计与实现

4.2.1 秒杀系统后台实体类详细设计与实现

由于秒杀系统后台在实际应用场景中，涉及到的实体类的属性非常多，这里仅对主要属性进行介绍，秒杀系统后台主要实体类图如图 4-2 所示：

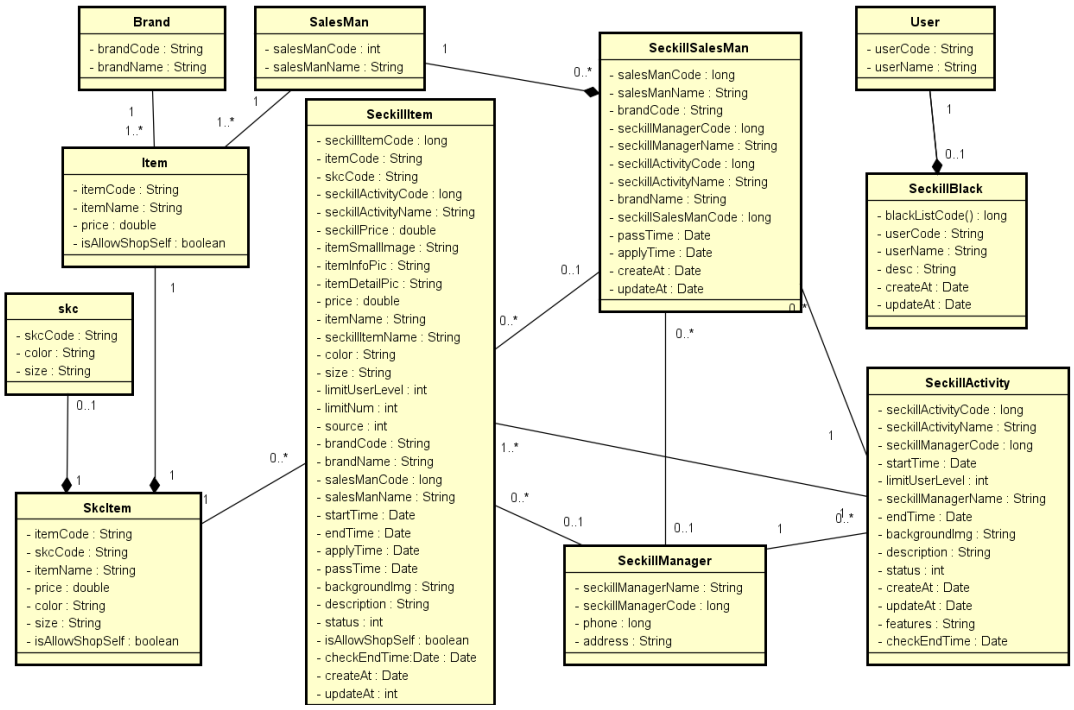


图 4- 2 秒杀系统后台主要实体类类图

Fig. 4-2 Main entity class diagram of seckill system's backend

SeckillItem：秒杀商品类，秒杀商品类和商品类关联，包含商品的品牌信息，如品牌名称、品牌编号，商品 skc 信息，如商品型号、颜色；秒杀商品类和商品供应商类关联，包含商品供应商信息，即秒杀商品是由哪个供应商添加的；秒杀商品类和秒杀活动关联，包含秒杀活动信息，如活动开始时间、结束时间；秒杀商品类和商场管理员关联，包含商场管理员信息，即秒杀商品是由哪个商场管理员添加的。秒杀商品类还包含秒杀商品的详细信息，如秒杀商品名称、申请审核时间、审核成功时间等信息。

SeckillSalesMan：秒杀供应商类，秒杀供应商类主要指示供应商参与的秒杀活动，并包含供应商参与秒杀活动的状态，如审核通过、审核未通过，申请参与活动的时间等信息。

SeckillBlack：秒杀黑名单类，秒杀黑名单类主要指示哪些用户不能参与秒杀活动，包含黑名单用户的编号，姓名等信息。

SeckillActivity：秒杀活动类，包含秒杀活动的所有信息，如秒杀活动名称、海报、开始时间、描述。

SeckillManager：秒杀商场管理员类，包含参与秒杀活动商场管理员的所有信

息，如商场管理员编号、姓名、电话、秒杀活动 ID 等信息。

4.2.2 秒杀活动管理模块详细设计与实现

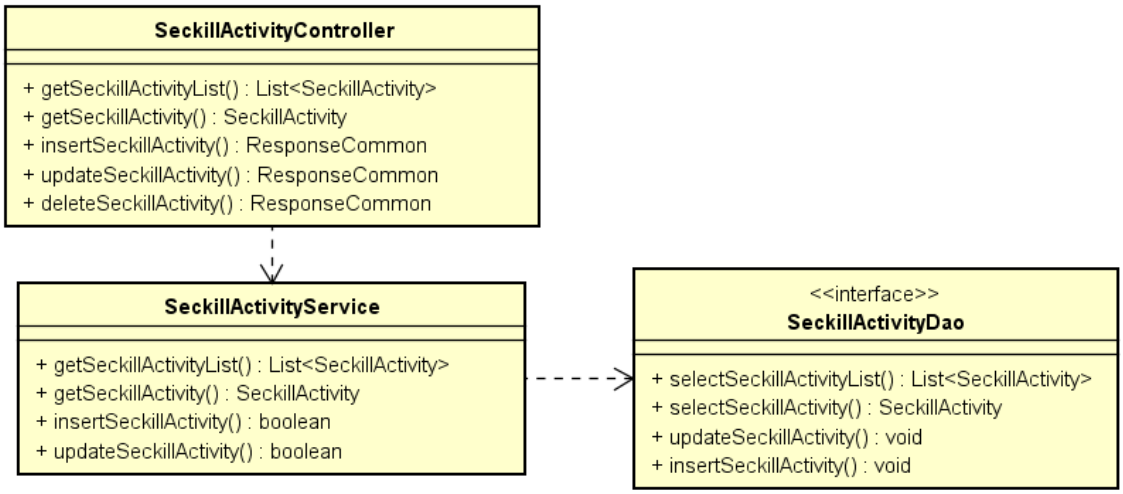


图 4-3 秒杀活动管理模块类图

Fig. 4-3 Activity management module class diagram

秒杀活动管理模块类图如图 4-3 所示，秒杀活动管理模块主要由 seckillActivityController 类提供 getSeckillActivityList()、getSeckillActivity()、insertSeckillActivity()、updateSeckillActivity()、deleteSeckillActivity()方法分别对 SeckillActivityService 类中的 getSeckillActivityList()、getSeckillActivity()、insertSeckillActivity()、updateSeckillActivity()方法进行调用。以上方法分别对查询秒杀活动列表，获取秒杀活动详细信息，新增秒杀活动，修改和删除秒杀活动的操作进行响应。删除秒杀活动只将秒杀活动的状态信息修改为已删除。

4.2.3 秒杀供应商管理模块详细设计与实现

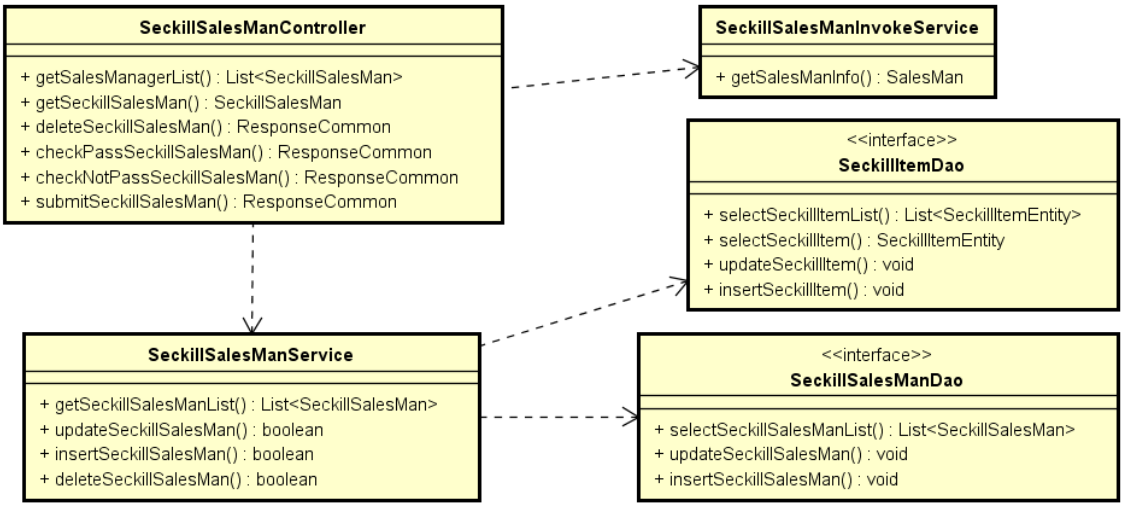


图 4-4 秒杀供应商管理模块类图

Fig. 4-4 Sales man management module class diagram

秒杀供应商管理模块类图如图 4-4 所示，秒杀供应商管理模块主要由 SeckillSalesManController 类提供 getSalesManagerList ()、getSeckillSalesMan()、deleteSeckillSalesMan ()、checkPassSeckillSalesMan ()、checkNotPassSeckillSalesMan()、submitSeckillSalesMan()方法分别对秒杀供应商进行查询，获取秒杀供应商详细信息，删除秒杀供应商信息，审核通过供应商信息、审核不通过供应商信息操作进行响应。删除秒杀供应商操作将把秒杀供应商提供商品的状态信息全部修改为已删除。SeckillSalesManController 类中 getSeckillSalesMan()方法通过调用 SeckillSalesManInvokeService 类中的 getSalesManInfo()方法对汇金内部系统的供应商数据进行获取。

SeckillSalesManService 类中 getSeckillSalesManList()方法通过调用 SeckillSalesManDao 接口中的 selectSeckillSalesManList()方法获取申请参与秒杀活动的供应商列表，updateSeckillSalesMan()方法通过调用 SeckillSalesManDao 接口中的 updateSeckillSalesMan()方法，对供应商审核通过或者不通过的状态在数据库中进行供应商状态更新，insertSeckillSalesMan()方法通过调用 SeckillSalesManDao 类中的 insertSeckillSalesMan()方法，对供应商参与秒杀活动的请求，进行秒杀供应商数据库的插入。

4.2.4 秒杀商品管理模块详细设计与实现

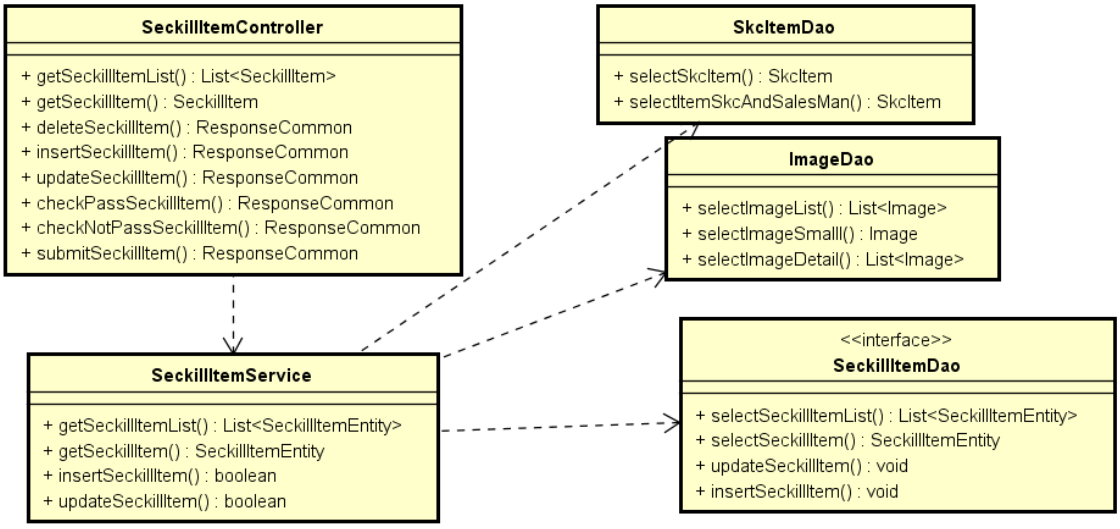


图 4-5 秒杀商品管理模块类图

Fig. 4-5 Seckill item management module class diagram

秒杀商品管理模块类图如图 4-5 所示，秒杀商品管理模块主要由SeckillItemController类提供getSeckillItemList()、getSeckillItem()、deleteSeckillItem()、insertSeckillItem()、updateSeckillItem()、checkPassSeckillItem()、checkNotPassSeckillItem()、submitSeckillItem()方法对秒杀商品进行查询，获取秒杀商品详细信息，删除秒杀商品信息，更新秒杀商品信息、审核通过商品信息、审核不通过商品信息、提交审核秒杀商品信息操作进行响应。

SeckillItemService 类 getSeckillItemList()方法通过调用 SeckillItemDao 类中的selectSeckillItemList()方法获取参与秒杀活动的秒杀商品列表，getSeckillItem()方法通过调用 SkcItemDao 中的 selectSkcItem()方法或者 selectItemSkcAndSalesMan()方法获取商品的详细信息。insertSeckillItem()方法对 SeckillItemController 类中的submitSeckillItem()和 insertSeckillItem()方法进行响应，即对管理员或供应商添加秒杀商品操作进行响应，管理员添加的商品默认状态为已通过审核，供应商添加秒杀商品默认状态为未审核。updateSeckillItem()方法对 SeckillItemController 类中的 deleteSeckillItem()、updateSeckillItem()、checkPassSeckillItem()、checkNotPassSeckillItem()方法进行响应，对秒杀商品信息或者秒杀商品状态信息进行修改。

4.2.5 用户管理模块详细设计与实现

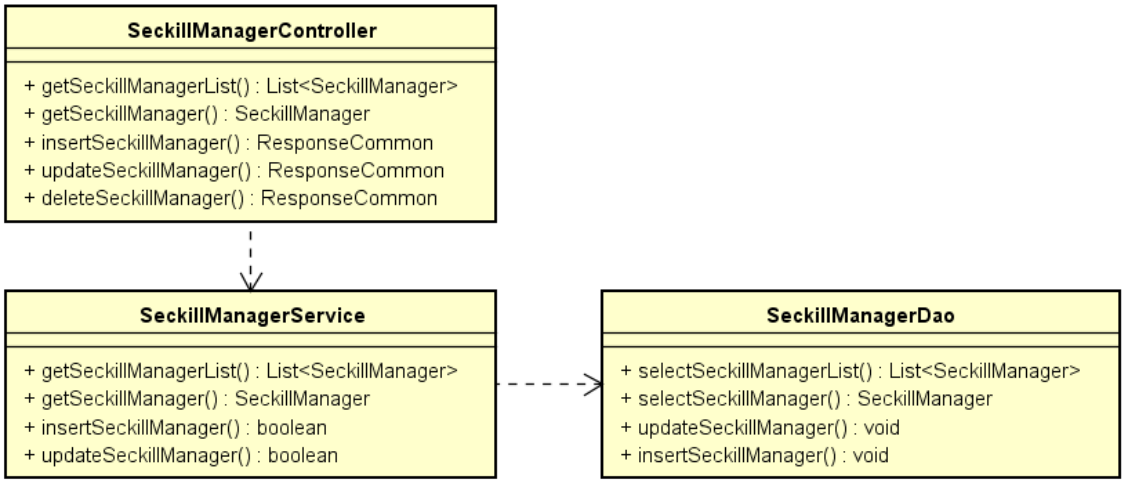


图 4-6 用户管理模块类图

Fig. 4-6 User management module class diagram

秒杀用户管理模块类图如图 4-6 所示，秒杀用户管理模块主要由 SeckillManagerController 类提供 getSeckillManagerList()、getSeckillManager ()、insertSeckillManager()、updateSeckillManager()、deleteSeckillManager ()方法对商场管理员进行查询，获取商场管理员详细信息，插入商场管理员信息，更新商场管理员信息、更新商场管理员信息和删除商场管理员操作进行响应。

4.2.6 秒杀黑名单管理模块详细设计与实现

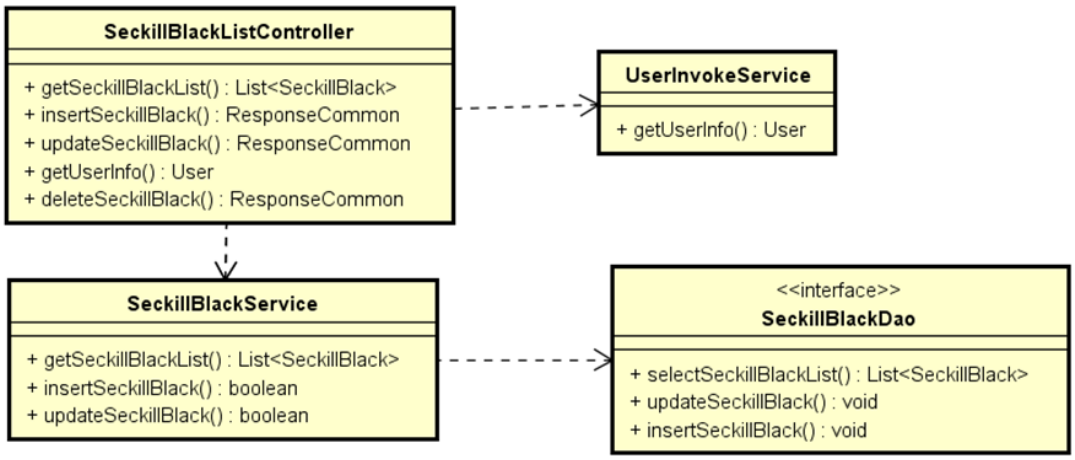


图 4-7 秒杀黑名单管理模块类图

Fig. 4-7 Seckill blacklist management module class diagram

秒杀黑名单管理模块类图如图 4-7 所示，秒杀黑名单管理模块主要由 SeckillBlackListController 类提供 getSeckillBlackList()、insertSeckillBlack()、updateSeckillBlack()、deleteSeckillBlack ()方法对黑名单用户列表进行查询，新增黑名单用户信息，更新黑名单用户信息、删除黑名单用户信息操作进行响应。getUserInfo()方法将调用 UserInvokeService 类中的 getUserInfo()方法对汇金百货内部系统的用户信息进行获取。

4.3 本章小结

本章主要对秒杀系统后台进行架构设计后，按照划分的功能模块对秒杀系统后台进行详细的设计与实现。

第五章 面向消费者的秒杀系统设计与实现

本章将对面向消费者的秒杀系统进行架构设计和详细设计与实现，架构设计包含网关层架构设计，分布式业务层架构设计和 redis 缓存层架构设计。详细设计包含对面向消费者的秒杀系统的各个业务层服务、库存同步服务、网关限流机制进行具体的设计与实现。面向消费者的详细设计与实现需要充分考虑业务操作的幂等性，流量限流，缓存更新，缓存充分利用，系统一致性等问题。

5.1 系统架构设计

汇金百货秒杀系统将以汇金百货现有的 O2O 在线交易平台[2]为背景，进行秒杀系统的设计与实现。汇金百货 O2O 平台已有供应商进销存系统、商城结算系统、用户行为分析系统、购物车管理系统、订单管理系统和正常商品售卖系统。面向消费者的秒杀系统将针对上述对汇金百货商品秒杀系统的功能性需求，以及对系统的可配置性、可拓展性、可维护性、安全性等非功能性需求，进行整体的架构设计和开发。面向消费者的秒杀系统整体架构如图 5-1 所示。

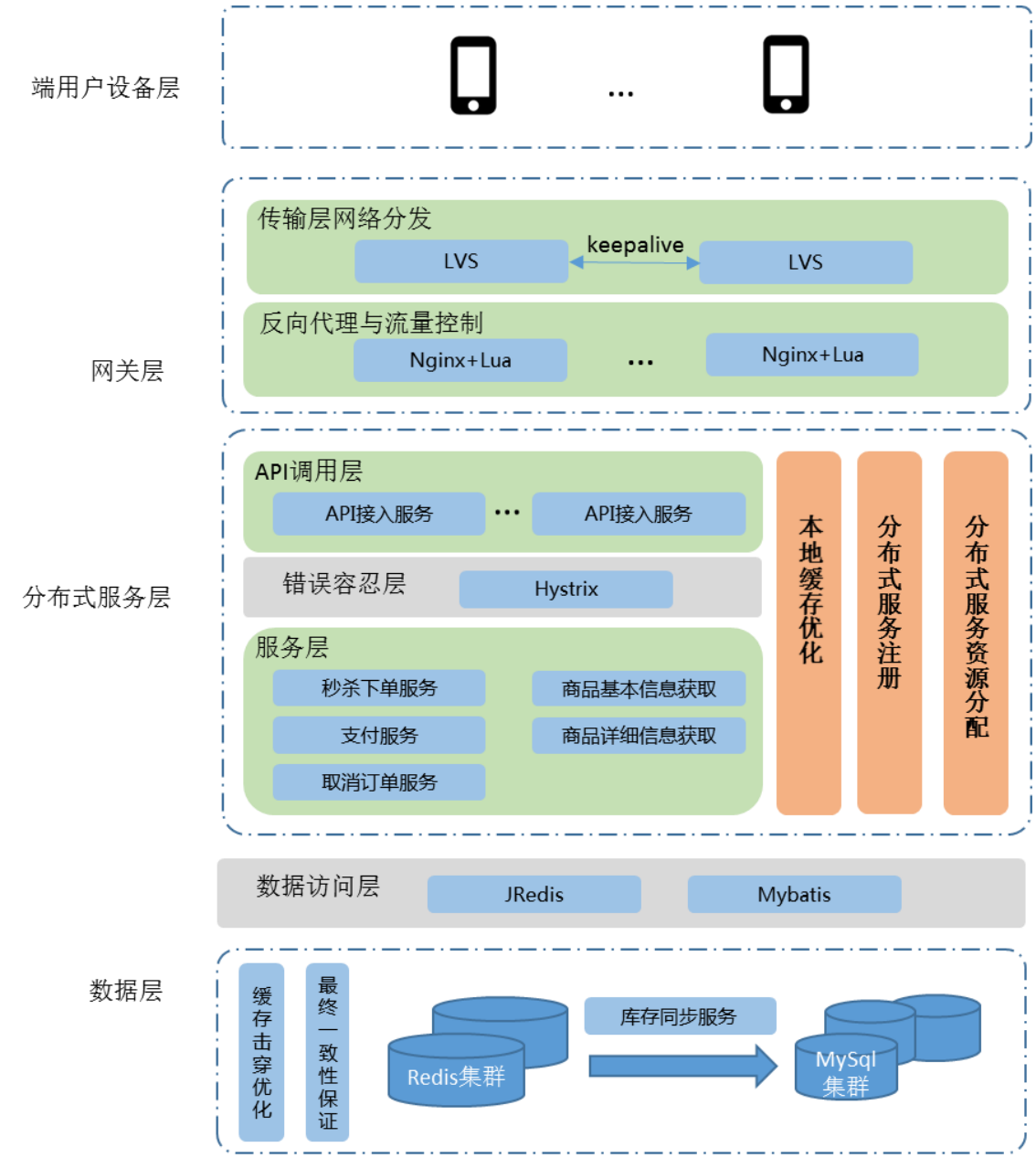


图 5- 1 面向消费者的秒杀系统架构图

Fig. 5-1 Consumer-oriented seckill system's structure

5. 1. 1 网关层架构设计

针对系统网关，商品秒杀系统使用 LVS 配合 nginx 和 keepalive[37]进行网关层的整体设计。虽然 LVS 有多种使用模型，为了满足系统对高并发的支持要求，使用 LVS 技术支持并发量最高且不产生网络流量的 LVS/DR 模型[37]进行流量分发，

网关层架构如图 5-2 所示：

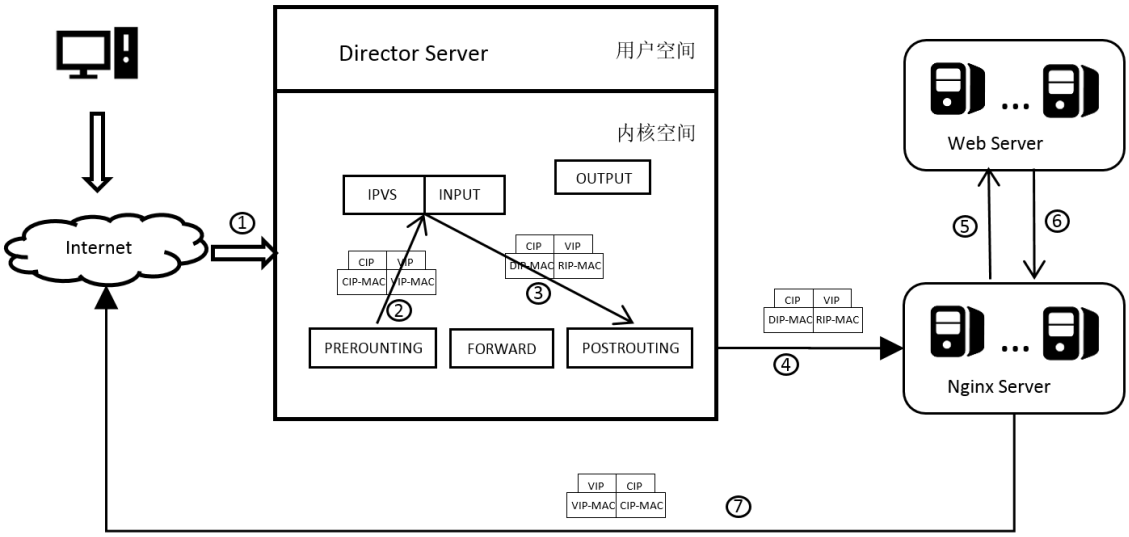


图 5- 2 网关层架构图

Fig. 5-2 Gateway layer's structure

用户对虚拟 IP 地址发送请求，请求报文首先到达 Director Server（负载均衡器节点）的内核空间，收到的报文的源 IP 地址为客户端 IP 地址，目标 IP 地址为虚拟 IP 地址。LVS 的 IPVS 检查到请求属于客户端集群请求后，根据源地址散列调度算法从正常运行的 nginx 中选出一台，并将报文的目的地址修改为 nginx 服务器的 mac 地址，将报文的源 mac 地址替换为 Director Server 本身的 mac 地址，由于 Direct server 和 Nginx server 处于同一个物理网络中，所以这里并不对源 ip 地址和目的 ip 地址进行任何修改，将报文发送给 Nginx 服务器，nginx 服务器对此报文进行限流处理后，将数据包传递给应用服务器处理，应用服务器处理业务完成后，将需要返回给客户端的数据返回给 nginx 服务器。Nginx 将数据返回给客户端服务器，该数据报文的源 mac 地址为虚拟 mac 地址，目标 mac 地址为客户端 mac 地址，源 ip 地址为虚拟 ip，目标 ip 地址为客户端 ip 地址。

网关模型的高可用表现在于，首先当一台安装 LVS 的 direct server 由于服务器宕机、web 服务终止、网线松动等原因不工作时，基于 LVS 本身的无状态性，keepalive 能使用备份的安装 LVS 的 direct server 进行切换。其次，当一台 nginx 服务器宕机时，虽然 LVS 并不支持对 nginx 服务器的转发规则自动删除，但 keepalive 除了能支持 LVS 宕机时，LVS 主备的切换外，还能自动删除 direct server 映射到

宕机 nginx 机器的转发规则。如果 nginx 服务器后面的 web server 宕机时，nginx 支持自动不往不响应的服务器发送请求。

网关架构使用 LVS 进行流量分发的效率比 nginx 进行负载均衡的效率高许多，且对请求响应时间影响较小，如果单个网关节点无法满足秒杀系统的并发请求，秒杀系统可以通过添加 DNS 规则对多个网关节点进行支持。

网关层可以对 nginx、tomcat 开启 Http 长连接优化，减少 TCP 频繁握手次数，大幅提高系统响应时间从而提高系统并发量，节约缓存 sockets 会话信息资源。在秒杀系统中，客户端平均响应时间为 3s，最大响应时间为 10s，平均请求次数为 100 次，针对客户端进行相应的长连接时间 keepalive_timeout 参数和最大空闲连接数的设置，能有效减少客户端和 nginx 服务器建立连接的次数。在秒杀系统中，由于秒杀服务中请求的到达往往是十分不平稳且具有高并发的特性，对 nginx 服务器设置最大空闲连接数量为 500。对于 API 接入服务运行的 springboot 内嵌的 tomcat 容器而言，需要对连接保持时间进行设置，在秒杀场景中，设置为 1 分钟。

5.1.2 分布式业务层架构设计

由于秒杀服务具有瞬时流量较高的特殊性，为了让系统具有较高的并发量，秒杀系统需要使用分布式系统的搭建。出于秒杀系统对配置简单、系统伸缩性有较高的需求的考虑，需要将秒杀系统拆分成一个个的分布式服务。秒杀系统使用 dubbo 配合 zookeeper 进行分布式业务层的搭建，每个服务都使用 springboot 微服务架构[38]进行搭建，业务层调用关系示意图如图 5-3 所示。

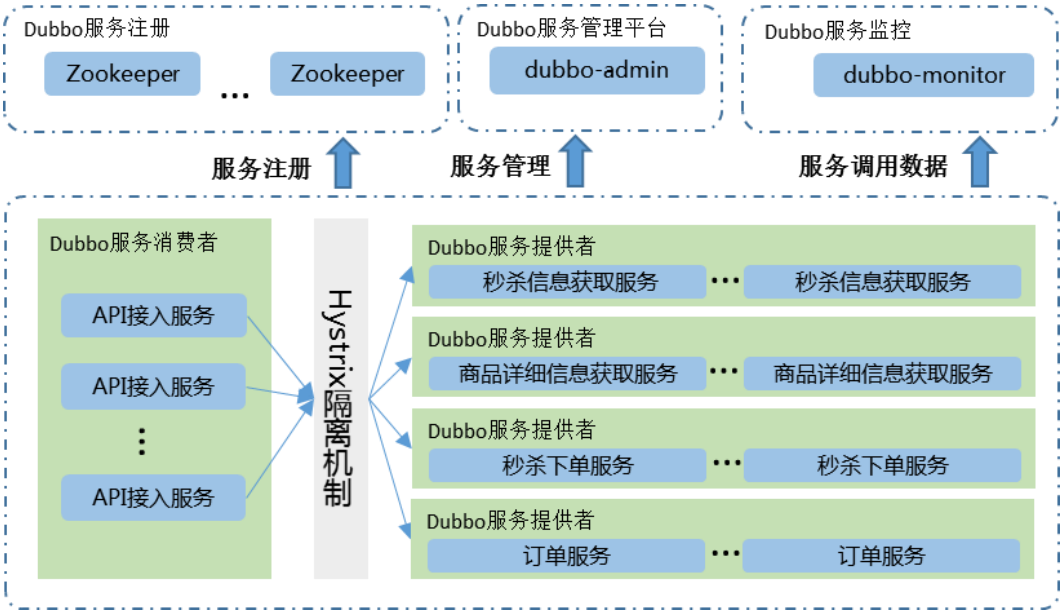


图 5-3 业务层调用关系示意图

Fig. 5-3 Service invoke relational structure

针对分布式业务层，为了支持业务的可拓展性，可以根据功能业务需求，把分布式业务层拆分为以下五个独立的服务系统：

(1) 秒杀信息获取服务：秒杀信息的获取可被分为秒杀活动信息获取和秒杀商品信息获取，秒杀商品信息获取分为参与本次秒杀活动的商品列表信息获取和商品基本信息获取，秒杀活动信息、秒杀商品列表信息、商品基本信息的获取主要在数据库中对秒杀活动信息、秒杀商品信息、商品基本信息，商品的 SKC 信息和商品库存信息等进行获取后，将其放入缓存中并返回获取结果。

(2) 商品详细信息获取服务：商品详细信息获取服务主要在数据库中对商品图片信息、介绍信息等进行获取后，插入缓存中并返回获取结果，由于商品详细信息字符串较多，读取速度较为缓慢，故将商品详细信息读取设为一个分离的服务，并设置相应的线程池进行业务隔离，这样当商品详细信息读取服务不可用时，秒杀系统依然能获取商品基本信息进行秒杀下单业务，对系统稳定性提供了保证。

(3) 商品秒杀下单服务：商品秒杀服务模块主要作用是处理每个用户的秒杀请求，每个用户进行秒杀操作后，让商品数量减一，并增加用户订单。商品秒杀服务是秒杀系统中并发量最高的服务，同时大量商品数量减一操作请求在高并发的情况下将引发库存锁争抢，而且由于秒杀下单服务运行时间较长，在单位时间内商品秒杀下单服务需要承受的并发量最高，容易引发整个系统线程等待耗尽不可

用的问题，因此这是本文重点设计和优化的对象。

(4) API 接入服务：API 接入服务主要负责对用户的 API 请求进行相应的服务的调用，将调用服务进行组合得到返回结果，返回给调用端。如果服务调用失败，系统将会进行业务降级操作，来保证系统的高可用、稳定性。API 接入服务属于主要的 Dubbo 服务消费者，API 接入服务的业务降级功能可以通过本地缓存一些服务调用结果进行实现。另外，API 接入服务可以直接从缓存中进行数据的获取，提高系统的响应速度。

(5) 订单服务：订单服务主要负责用户秒杀订单的取消，取消订单需要将数据库中订单的状态标记为取消，并把数据库和缓存中存储的商品的库存加一并更新，把用户购买的秒杀商品数量进行更新。

分布式业务层架构提供高可用支持，主要体现在以下两个方面。

第一，系统设计充分利用 dubbo 本身提供的高可用支持。zookeeper 服务注册中心，API 服务接入端，和秒杀服务提供者之间通过长连接进行网络通讯，如果任何一个服务提供者被监测出不可用时，zookeeper 服务注册中心将消息立即推送给 API 接入服务，API 接入服务收到消息后，将在服务调用的时候忽略不可用的服务[17]。如果某个服务的服务提供者全部不可用，dubbo 会让服务消费者对服务提供者进行多次重新连接，如果重新连接依旧无法调用服务，系统会进行业务降级。如果监控中心宕机，部分服务的调用信息可能丢失，但不影响整个系统服务的正常运行。如果 zookeeper 服务注册中心集群中的一台宕机，将自动切换到备份的 zookeeper 服务注册中心集群的另一台。如果全部的 zookeeper 服务注册中心集群中的机器都不可用，API 接入服务将使用本地缓存的服务提供者地址调用他们的服务。

第二，系统使用 Hystrix 进行业务降级提供高可用支持。当任何服务提供者提供的服务的不可用率在限定时间内超过设定的阈值，Hystrix 熔断机制将对 API 接入服务的调用进行相应的降级处理，降级处理是指使用功能较弱的服务代替原本的服务。后续的所有服务调用在特定时间内都将进行降级处理，保证系统的高可用。

分布式业务层提供可伸缩性支持，主要体现在，当秒杀系统需要对某种服务提供者添加机器支持，尤其是对添加秒杀下单服务时，只要在服务启动时向注册中心动态添加该服务提供者的机器，API 接入端就会自动对服务进行调用，通过这种方式实现了秒杀系统的动态扩容。分布式业务层服务功能边界划分明确，耦合

度低，分布式业务层使用 maven 进行 jar 包的管理，同时使用 nexus 实现服务接口的版本管理和共享，减少了服务接口定义的代码重复量，简化了服务之间接口的调用的流程。

5.1.3 缓存集群架构设计

面向电商平台的秒杀系统的缓存集群设计如图 5-4 所示，主要采用 redis 客户端分片方案，配合 redis sentinel 方案进行 redis 集群节点高可用方案的搭建，主要出于以下原因：

面向电商平台秒杀系统的缓存集群需要在分布式的环境下支持锁操作，即 redis 事务，redis 事务与普通数据库事务[39]不同，redis 事务不支持事务回滚，需要代码进行业务补偿机制的实现。秒杀系统缓存集群需要支持 redis 事务的具体表现为，当多个微服务进程使用 redis 进行减库存操作时，需要对商品的库存上锁，如果不对商品库存上锁，假设两个订单服务同时进行减库存操作，两个订单同时获取商品库存，对商品数量减一后，写入 redis，这时 redis 库存仅被减了一，但却产生了两个订单，引发超卖等一系列问题，所以 redis 集群必须对分布式锁操作进行支持。Twemproxy+redis 方案目前还不支持分布式锁，即 redis 事务操作。Redis cluster 方案对分布式锁支持力度不佳，redis cluster 方案在执行 MULTI/EXEC 命令时，对于主键不在当前节点上的命令，会返回执行失败，对于主键在当前节点上的命令，能全部执行成功，也就是说部分支持执行事务的 MULTI/EXEC 指令。所以这两种集群方案都不太适合进行面向电商平台秒杀系统的缓存集群搭建。

面向电商平台秒杀系统的缓存集群需要支持较高的并发量，并且需要达到高可用和较高的稳定性需求，考虑到 redis cluster 的 gossip 网络消耗，当面临大的并发状况时，redis cluster 方案存在不稳定的问题目前没有大型网站使用 redis cluster 方案搭建缓存集群。Twemproxy 集群方案支持的并发数量没有直接进行客户端分片高，且需要额外对部署 twemproxy 机器的单机故障进行处理，所以面向电商平台秒杀系统的缓存集群使用客户端分片方案设计搭建。

面向电商平台秒杀系统由于面向对象仅为汇金百货商场，秒杀商品有一定的局限性。假设每条商品信息数据包含商品库存信息，商品基本信息，商品详细信息等大小总和为 1K，4G 内存的 redis 服务器能缓存 100 万条商品信息。所以把商品基本信息和商品详细信息分片到一个 redis 实例是可行的。一条订单信息维持时间为 30 分钟，假设每条订单信息的大小为 1K，4G 内存的 redis 服务器能缓存 100

万条订单数据，把所有订单信息分片到同一个 redis 实例是可行的。

面向电商平台的秒杀系统的缓存集群需要有较高的可用性。面向电商平台的秒杀系统为了提高秒杀下单操作的并发量，需要对秒杀下单操作进行数据库事务降级操作，商品减库存的操作将由 redis 事务执行，在秒杀活动进行时，商品的库存信息将完全保存在 redis 中，待秒杀活动结束后，商品库存信息将异步更新到数据库中，因此，在秒杀活动过程中需要保持 redis 缓存集群的高可用。Redis sentinel 官方方案[25]提供对 redis 高可用支持，比较稳定，不易发生数据丢失并且支持 redis 主节点失效以后的自动切换。

面向电商平台秒杀系统缓存集群方案如图 5-4 所示，缓存调用服务器按照自身的需要可以接入商品信息缓存高可用集群和订单缓存高可用集群。Sentinel 进程负责对缓存集群进行监控管理，当某个缓存集群的 master 宕机不可用后，sentinel 进程将在 30s 内自动进行 slave 转 master 的切换，新 master 将接管旧 master 的所有任务，同时所有的 slave 节点也动态的与新 master 同步，当失效的 master 修复后，将自动变成 slave 节点。

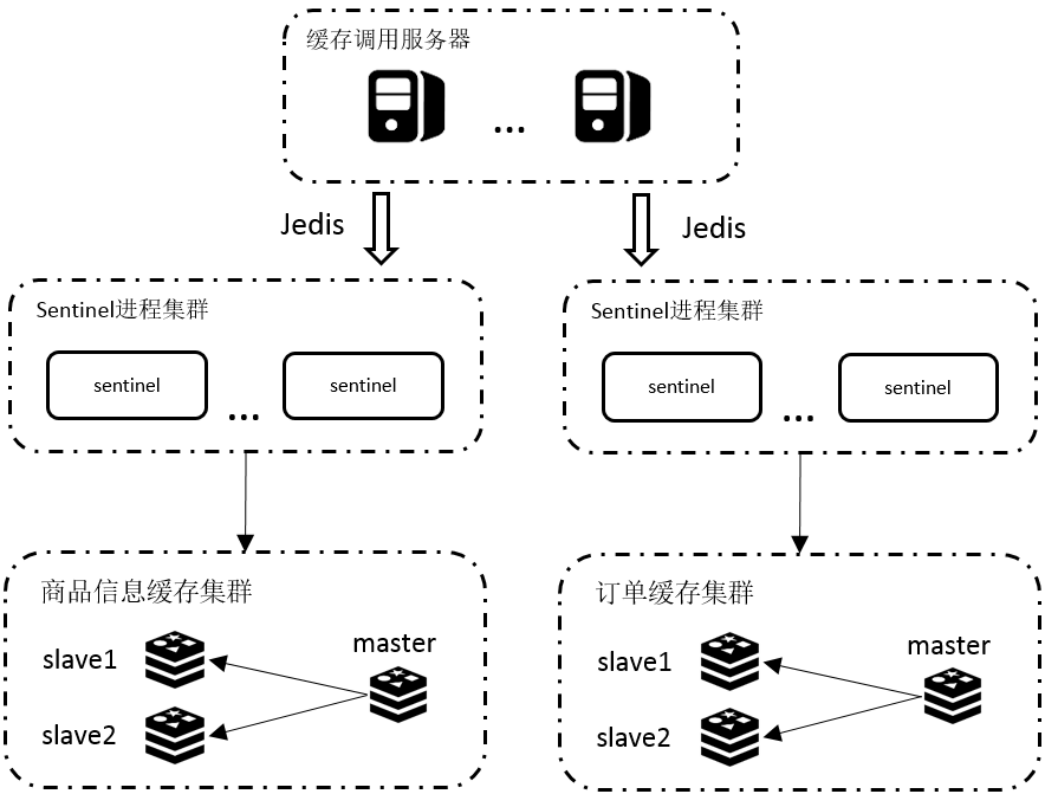


图 5- 4 秒杀系统 Redis 集群架构图

Fig. 5-4 Redis cluster’s architecture of seckill system

5.2 系统缓存划分设计

秒杀系统需要通过提高缓存的利用率来增加活动并发度，系统对缓存拆分如下：

(1) 秒杀活动信息，主键为 SEC_ACTIVITY_INFO_秒杀活动 ID，主要存储秒杀活动的开始时间，结束时间，活动名称，活动介绍等信息。

(2) 秒杀活动商品列表信息，主键为 SEC_ACTIVITY_ITEM_LIST_秒杀活动 ID_PAGEID，主要存储参与秒杀活动的 20 条秒杀商品 ID 信息。

(3) 秒杀商品基本信息的主键设置为 SEC_ITEM_INFO_秒杀商品 ID，主要存储秒杀商品名，秒杀价格，颜色，型号，规格，产地，秒杀缩略图等商品的基本信息。

(4) 秒杀商品详细图片信息主键设置为 SEC_ITEM_DETAIL_商品 ID，主要存储商品图片，商品详情介绍等信息。将其与秒杀信息分开存放的原因是由于商品详细信息字符串较多、占用内存较大，在用户刚进入商品展示页面时，系统不需要展示商品详细信息，将商品详细信息分开存放、读取，有助于提高系统的稳定性。

(5) 秒杀商品的库存信息，主键为 SEC_ITEM_REMAIN_INFO_商品 ID，主要存储商品的库存信息，分开存放的原因是在秒杀系统中库存信息的读取写入非常频繁，许多时候页面只想获取秒杀商品的库存信息，分开存放有助于提高网络传输效率。

(6) 用户秒杀商品订单信息，主键为 SEC_USER_ORDER_用户 ID_商品 ID_SKCID，记录单个用户购买秒杀商品的数量，方便系统审核用户购买单一秒杀商品是否超出数量限制。

(7) 用户订单列表信息，主键为 SEC_ORDER_LIST_用户 ID，主要存储订单的 ID，购买商品的 ID，方便用户进行用户订单列表的查询，该缓存划分属于 O2O 系统原有缓存划分，在添加秒杀订单的时候应该更新该缓存存储的列表信息。

(8) 用户订单详细信息，主键为 SEC_ORDER_订单 ID，包含 O2O 系统原有的订单详细信息缓存的所有属性，并添加是否秒杀订单，秒杀活动编号，秒杀商品编号等属性。

5.3 系统业务层的详细设计与实现

5.3.1 API 接入服务设计实现

如图 5-5 所示，API 接入服务主要对客户端产生的获取秒杀活动信息、秒杀下单、取消商品订单请求提供接口，对秒杀信息获取服务、秒杀下单服务、商品详细信息获取服务、订单服务进行调用。API 接入服务中的类分为三种，Controller 类为客户端接口类，主要负责对客户端的调用进行接入并响应，CacheService 类为缓存类，主要被 Controller 类调用，获取或插入相应的信息到缓存中，invokeService 类为服务调用类，负责进行远程服务的调用，每个远程服务调用方法都有相应的业务降级方法负责对调用失败的服务进行补偿。

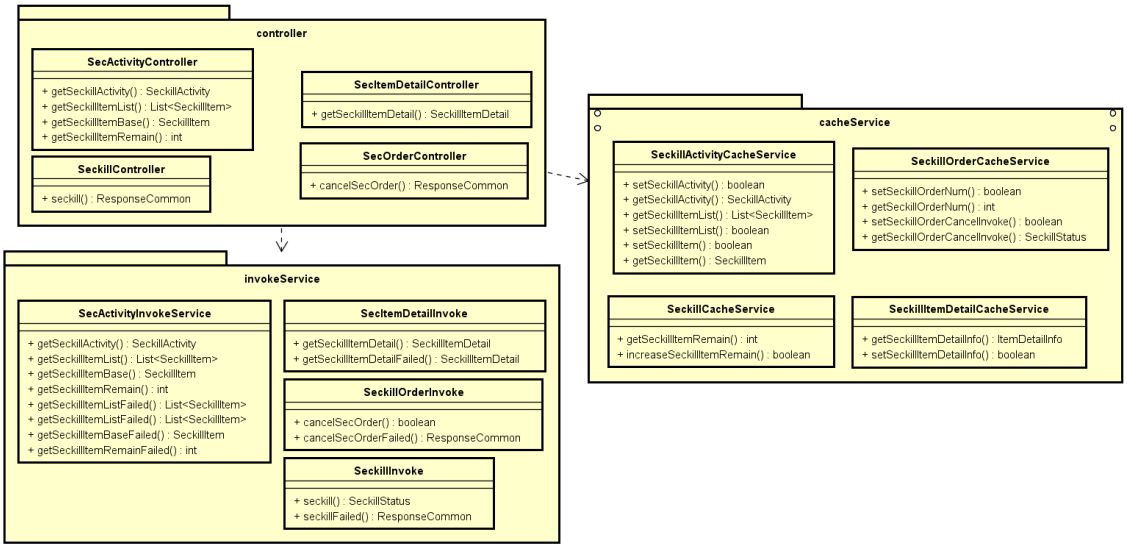


图 5- 5 API 接入服务类图

Fig. 5-5 API access service class diagram

下面将详细介绍 API 接入服务中 Controller 接口类的每个方法的代码流程。

（1）SecActivityController 类：getSeckillActivity()方法负责响应用户获取秒杀活动信息的请求，秒杀活动信息包含秒杀活动开始时间、结束时间、标题、介绍、规则等信息。getSeckillItemList()方法对用户获取参与该秒杀活动的秒杀商品列表信息的请求进行响应，getSeckillItemBase()方法对用户获取秒杀活动商品基本信息的请求进行响应，getSeckillItemRemain()方法对用户获取商品库存信息的请求进行响应。

获取秒杀商品基本信息、秒杀活动信息、商品库存信息的请求在 API 接入服务中的代码流程图如图 5-6 所示。对于完整的秒杀商品列表信息的获取，是在对秒杀商品列表缓存信息获取后，按照秒杀商品列表缓存信息中的商品编号，对每个单独的秒杀商品信息按照图 5-6 中的获取流程进行信息获取，最后将秒杀商品信息拼接成完整的秒杀商品列表信息。

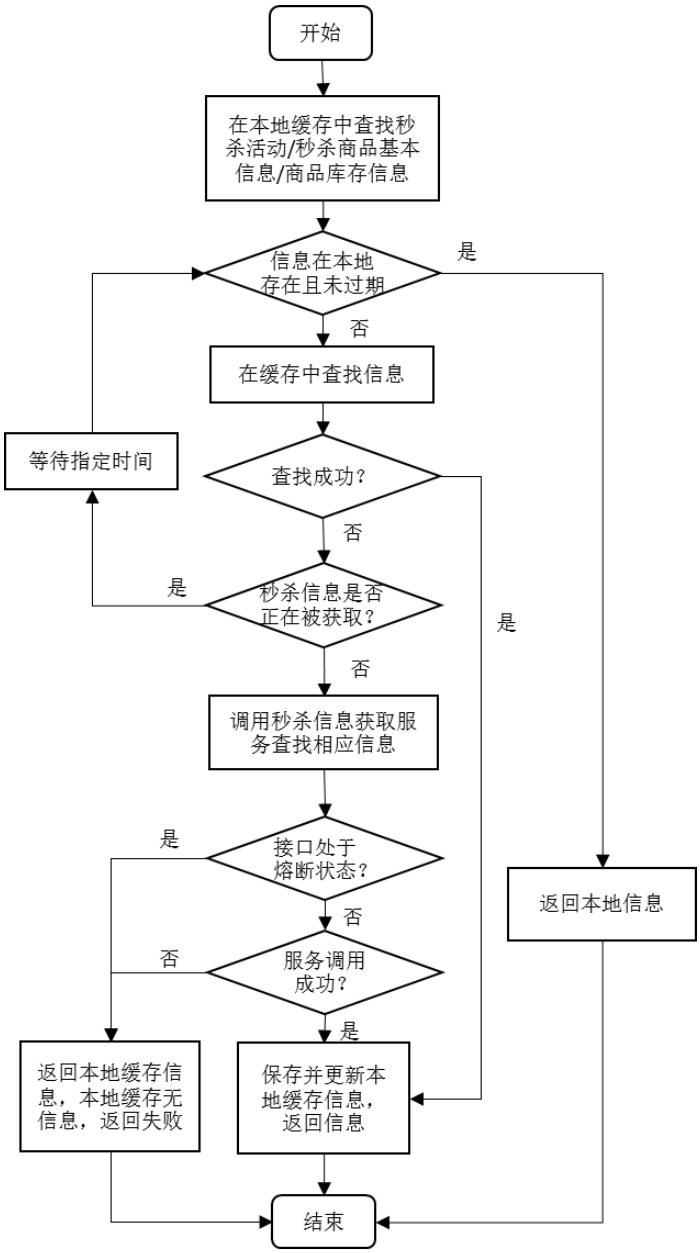


图 5- 6 API 接入服务-秒杀信息获取代码流程

Fig. 5-6 API access service-get seckill message code process

商品基础信息、秒杀活动信息等信息在秒杀活动开始后具有不易变的特性，并且即使在数据库中对这些信息进行修改，修改完成后数据库中的信息和展示给消费者的信息具有一致性容忍度较高的特性，可以对这些信息进行本地缓存优化。

首先，秒杀信息将被部分缓存在本地缓存中。在本系统中，秒杀活动基本信息和秒杀商品列表信息、商品基础信息本地缓存过期时间为 1 小时，秒杀库存信息本地缓存过期时间为 300ms，如果本地缓存信息过期，可从 redis 缓存中获取新的秒杀信息。每次在 redis 中进行商品信息获取时，都将进行本地缓存的存储和时间戳的更新。这样做充分利用了 API 接入服务器的本地缓存，有效的减少了 redis 缓存的连接数量，能较好的提高秒杀信息获取的效率，保证了缓存的充分利用。

其次，当单个 API 接入服务接收到多条对同一个秒杀信息的获取请求，并且 API 接入服务机器中没有本地缓存该秒杀信息，需要进行秒杀信息服务调用时，系统对秒杀请求的调用进行优化。API 接入端会将所有秒杀信息获取服务的调用信息保存在 InfoInvokeConcurrentHashmap 中，并设置时间戳，如果调用秒杀信息服务成功，删除秒杀信息获取服务正在调用信息。如果在后续极短的时间内，该 API 接入服务需要对秒杀信息进行秒杀服务的调用，在 InfoInvokeConcurrentHashmap 中发现该秒杀信息服务正在被调用且距离时间戳时间不超过 800ms，则等待正在调用的服务进行了 800ms 后，尝试读取本地缓存，如果本地缓存读取成功，返回商品信息，如果本地缓存读取失败，则尝试调用秒杀信息获取服务。

在实际线上操作的时候，需要首先进行秒杀活动的预热。预热后，大量的秒杀信息都可以直接在缓存中获取，秒杀信息获取服务可以少量部署，节约了系统机器资源，提高了系统的灵活性。由于秒杀信息获取服务是无状态的，系统可以根据需要灵活配置秒杀信息获取服务的机器数量。

（2）SeckillController 类：seckill()方法主要对用户秒杀下单请求进行响应。秒杀下单业务属于秒杀系统的主要业务，由于调用频率非常高，将其与其他业务分离。对于 SeckillInvokeService 类中的 seckill()方法，系统通过 Hystrix 为其配置单独的线程池，并分配较多的线程数量，这么做一来可以减少 tomcat 线程池的限制，保证服务的快速响应，二来提高系统资源的使用效率。

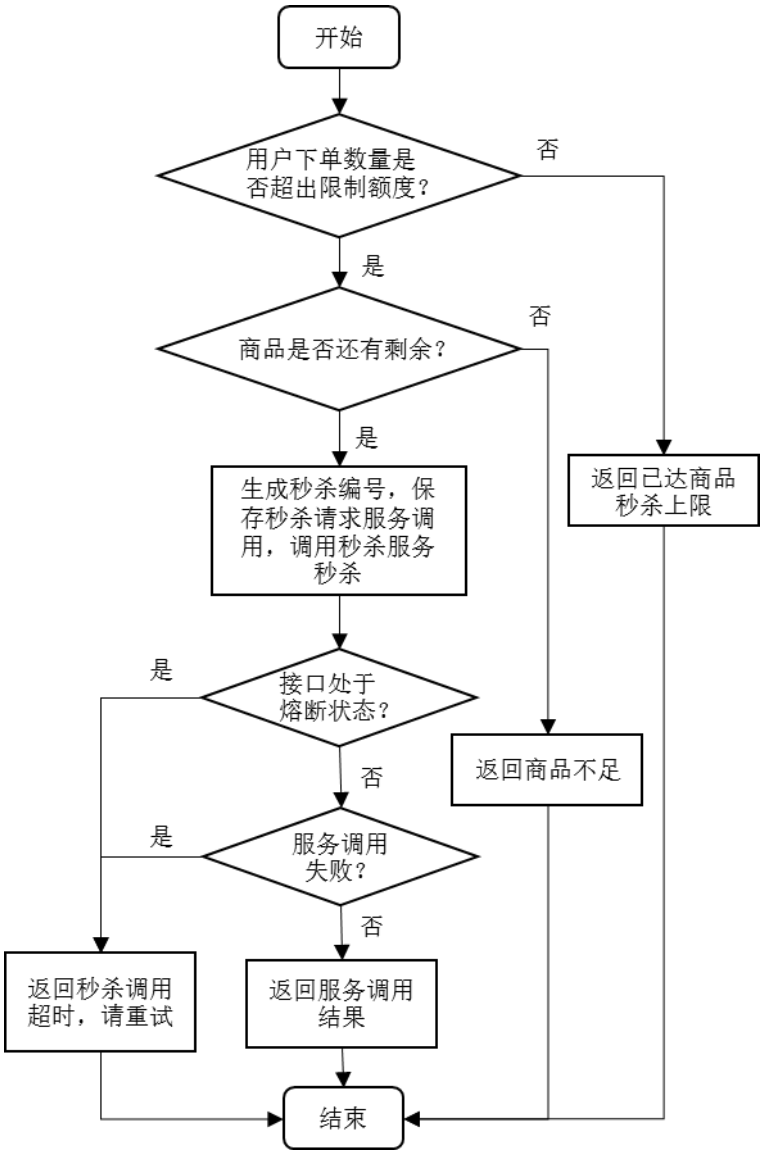


图 5- 7 API 接入服务-秒杀下单代码流程

Fig. 5-7 API access service-seckill code process

秒杀下单代码流程在 API 接入服务中如图 5-7 所示。由于秒杀商品对每个用户有下单次数的限制，API 接入端首先对缓存中的用户商品订单数量进行查询，判断用户是否超出商品限额，对超出商品购买限额的用户通知其秒杀商品已经达到上限。然后在缓存中获取商品库存，如果缓存中商品已经售完，返回商品已经售完的通知，保证商品售完后下单结果的快速反馈。

API 接入端将针对每一个秒杀请求生成秒杀编号，秒杀编号为 SECKILL_用户 ID_秒杀物品 ID_时间戳，设置时间戳是为了避免在进行秒杀服务调用的时候，由

于调用超时而进行重复调用，导致产生重复下单的情况。每个秒杀服务调用请求都有自己独立的业务编号，在进行秒杀服务调用时，秒杀服务会将业务编号保存在缓存中，这样当由于调用超时等原因收到重复请求时，可以直接返回缓存保存的结果，节约服务调用时间。如果秒杀服务正在熔断，直接返回秒杀失败；如果秒杀服务调用失败，比如发生秒杀服务响应超时等，即有可能是成功或失败的临界状态，此时需要返回用户重试。

（3）SecOrderController 类：cancelSecOrder ()方法主要对用户取消订单请求进行响应。SeckillOrderService 中的 cancelSecOrder()方法主要对订单服务的取消订单函数进行远程调用。取消订单服务远程调用失败时，seckillFailed()方法将返回取消订单失败。

（4）SecItemDetailController 类：getSeckillItemDetail()方法首先在缓存中获取商品详细信息，获取成功对商品详细信息返回，获取失败进行商品详细信息服务的调用，当商品详细信息获取服务调用失败时，调用 getSeckillItemDetailFailed()方法进行业务降级。

5.3.2 秒杀信息获取服务设计实现

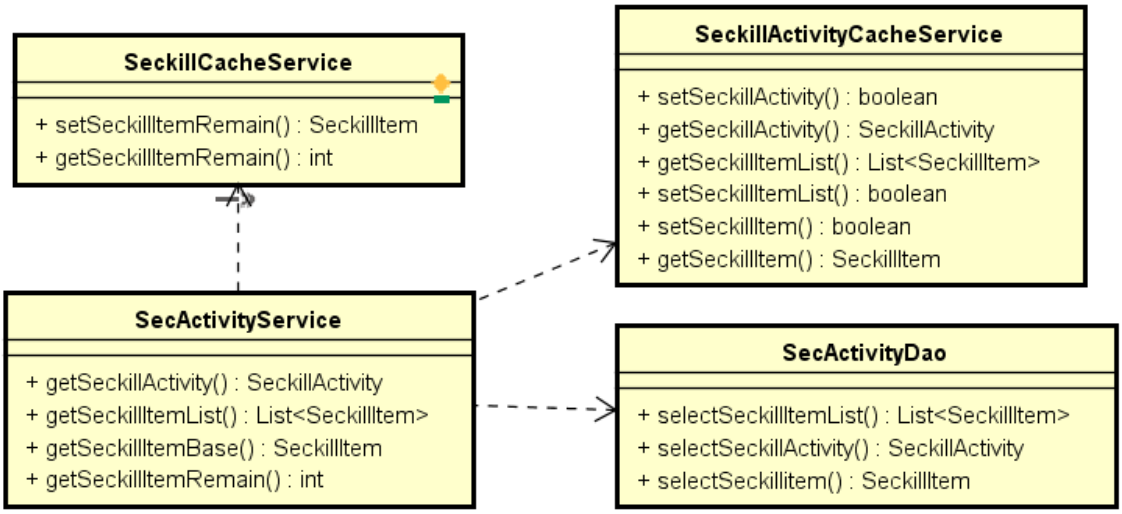


图 5-8 秒杀信息获取服务类图

Fig. 5-8 Seckill info access service class diagram

秒杀信息获取服务类图如图 5-8 所示，秒杀信息获取服务主要从数据库获取秒杀信息，并对 API 接入服务的调用进行响应，下面详细介绍秒杀信息获取服务接口的具体实现。

（1）SecActivityService 类：主要对 API 接入服务的秒杀信息获取请求进行响应，getSeckillActivity()方法从数据库获取秒杀活动信息并将其存入缓存中，包含秒杀活动开始时间、结束时间、标题、介绍、规则等信息。getSeckillItemList()方法获取参与该秒杀活动的完整的秒杀商品列表信息并将分散存入秒杀商品列表缓存和秒杀商品基本信息缓存中，getSeckillItemBase()方法获取秒杀活动商品基本信息并将其存入秒杀商品基本信息和秒杀商品库存缓存中，getSeckillItemRemain()方法获取商品库存信息实际上是获取商品基本信息后，将其存入秒杀商品库存缓存和商品基本信息缓存中。

（2）SecActivityDao 类：从数据库中获取秒杀商品列表、秒杀活动和秒杀商品详细信息。

（3）SeckillActivityCacheService、SeckillCacheService 类：从缓存中获取或插入相应的秒杀信息。

秒杀信息获取需要防止缓存击穿，在数据库中获取秒杀信息时，由于数据库获取信息速度较为缓慢，此时可能出现多次缓存击穿的问题。多次缓存击穿是指，如果有多个请求几乎同时对数据库进行秒杀信息获取操作，最早对秒杀信息进行数据库查询的操作并没有插入缓存中，将导致所有请求排队对数据库的进行重复的数据获取。为了防止缓存击穿，系统将在对数据库访问前，给对应的秒杀信息的键值设置一个锁，并给这个锁赋予一个超时时间，这里设置为 3 分钟，最先到达的请求去数据库中获取秒杀信息，随后到达的请求由于发现对应的秒杀信息被锁住后，会等待 50ms 后再次在 redis 中进行数据获取操作。给锁设置超时时间是为了防止在 redis 中删除秒杀信息后对秒杀信息进行重复获取时，出现由于秒杀信息被锁而无法在数据库中获取秒杀信息的情况。

5.3.3 商品详细信息获取服务设计实现

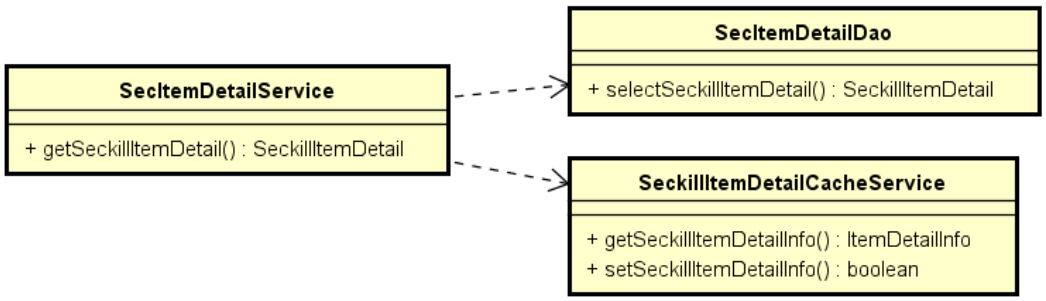


图 5-9 商品详细信息获取服务类图

Fig. 5-9 Item detail info access service class diagram

商品详细信息获取服务类图如图 5-9 所示，商品详细信息获取服务从数据库获取商品详细信息，并对 API 接入服务的调用进行响应，商品详细信息获取服务同样需要进行缓存击穿优化，下面详细介绍商品详细信息获取服务接口的具体实现。

（1）SecItemDetailService 类：主要对 API 接入服务的商品详细信息获取请求进行响应，getSeckillItemDetail()方法从数据库获取商品详细信息并将其存入缓存中，包含商品详细图片信息、商品介绍等信息。

（2）SecItemDetailDao 类：从数据库中获取商品详细信息。

（3）SeckillActivityCacheService、SeckillCacheService 类：从缓存中获取或插入相应的秒杀信息。

5. 3. 4 秒杀下单服务设计实现

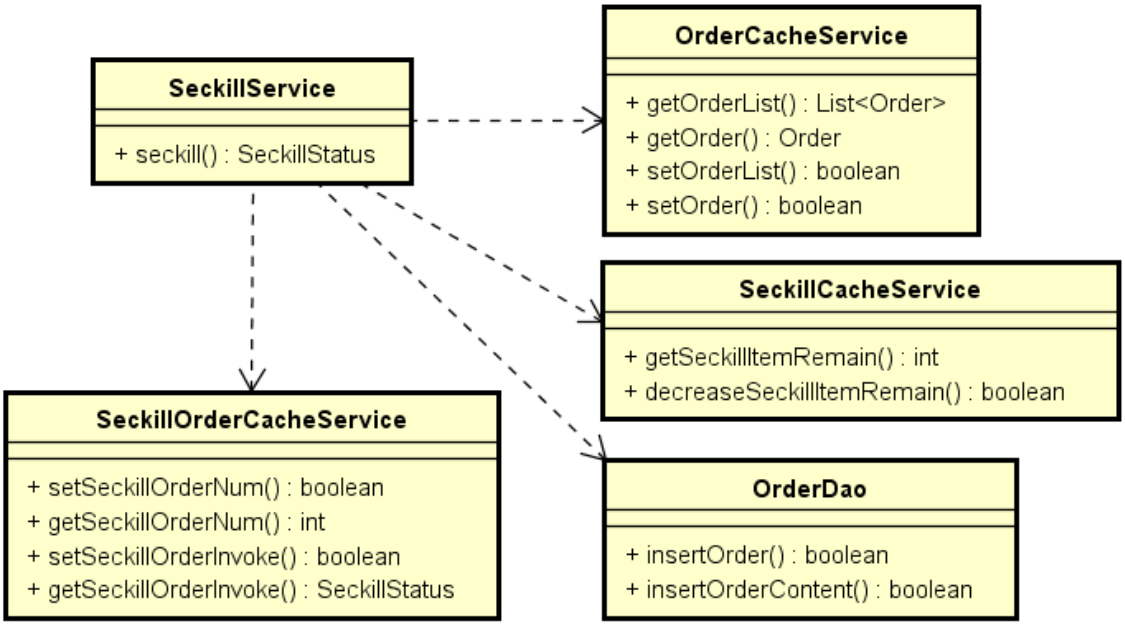


图 5- 10 秒杀下单服务类图

Fig. 5-10 Seckill service class diagram

秒杀下单服务类图如图 5-10 所示，下面详细介绍秒杀下单服务的具体实现。

（1）SeckillService 类：包含 seckill ()方法，主要对 API 接入服务的秒杀下单获取请求进行响应。

（2）orderDao 类：insertOrder()和 insertOrderContent()方法将秒杀订单信息存入数据库中。

（3）OrderCacheService、SeckillOrderCacheService 类：setSeckillOrderNum()或 getSeckillOrderNum()方法从缓存中获取或插入用户商品订单数量信息，setSeckillOrderInvoke()和 getSeckillOrderInvoke()方法从缓存中插入秒杀编号处理信息，过期时间为 10 分钟。秒杀编号的意义在于，当一台机器秒杀下单服务超时后，dubbo 将重新调用其他机器上的秒杀下单服务，导致同一秒杀编号的秒杀服务调用同时被多台机器处理，即一次秒杀下单请求得到多个商品订单。getOrderList()、setOrderList()、getOrder()、setOrder()方法作用在于，如果在缓存中保存订单列表信息或订单内容信息，那么需要对订单缓存信息进行更新。

（4）SeckillCacheService 类：getSeckillItemRemain()获取商品库存信息，decreaseSeckillItemRemain()尝试减少商品库存。

秒杀下单服务代码流程图如图 5-11 所示，在商品秒杀流程中，由于直接进行数据库库存减一操作会引发大量的数据库锁争抢，导致秒杀服务运行十分缓慢，这种情况下即使商品秒杀服务机器部署再多，也无法加快秒杀服务。为了加快秒杀服务的调用，系统对秒杀商品的抢库存操作进行事务降级，直接在 redis 上将秒杀商品的数量减一而不是在数据库进行减库存操作，通过这种方式减少了库存操作的时间。

为了提高系统的并发量和秒杀效率，秒杀操作使用 redis 乐观锁将商品锁住，而不是 redis 悲观锁，乐观锁在系统中的实现采用 watch，multi 指令进行。

由于 redis 不支持事务操作，只支持序列化，于是需要对秒杀操作进行库存减一操作和插入订单操作的分离。

插入订单操作是为了满足用户在秒杀成功后对订单快速联合查找的需求，插入订单操作在此过程中不做异步写入数据库的处理。

对于秒杀的减库存和插入订单到数据库和缓存操作，有以下几种情况。

(1) 库存减一 redis 事务操作失败，返回秒杀失败，结束业务流程。

(2) 库存减一 redis 事务操作成功，插入订单数据库和缓存成功，返回秒杀成功，结束业务流程。

(3) 库存减一 redis 事务操作成功，插入订单数据库或缓存，将重新尝试 2 次插入，重试插入成功，返回秒杀成功，结束业务流程，重试失败有以下两种情况。

如果是数据库插入出现插入数据库失败问题，可能是网络问题，也有可能是数据库发生严重故障，这种情况下只能暂停秒杀业务人工进行故障排查。秒杀服务端将打印出错误日志并返回错误，API 接入端收到错误后，进行熔断，对所有调

用秒杀服务的请求返回秒杀失败，避免较大的商业故障。

如果是插入订单缓存出现问题，系统直接返回秒杀成功，秒杀服务也将打印出错误日志并返回错误。由于 redis 高可用框架的搭建，保证了 redis 订单服务的高可用性，在小概率的情况下，用户可以进行超量次秒杀的操作是可以容忍的。

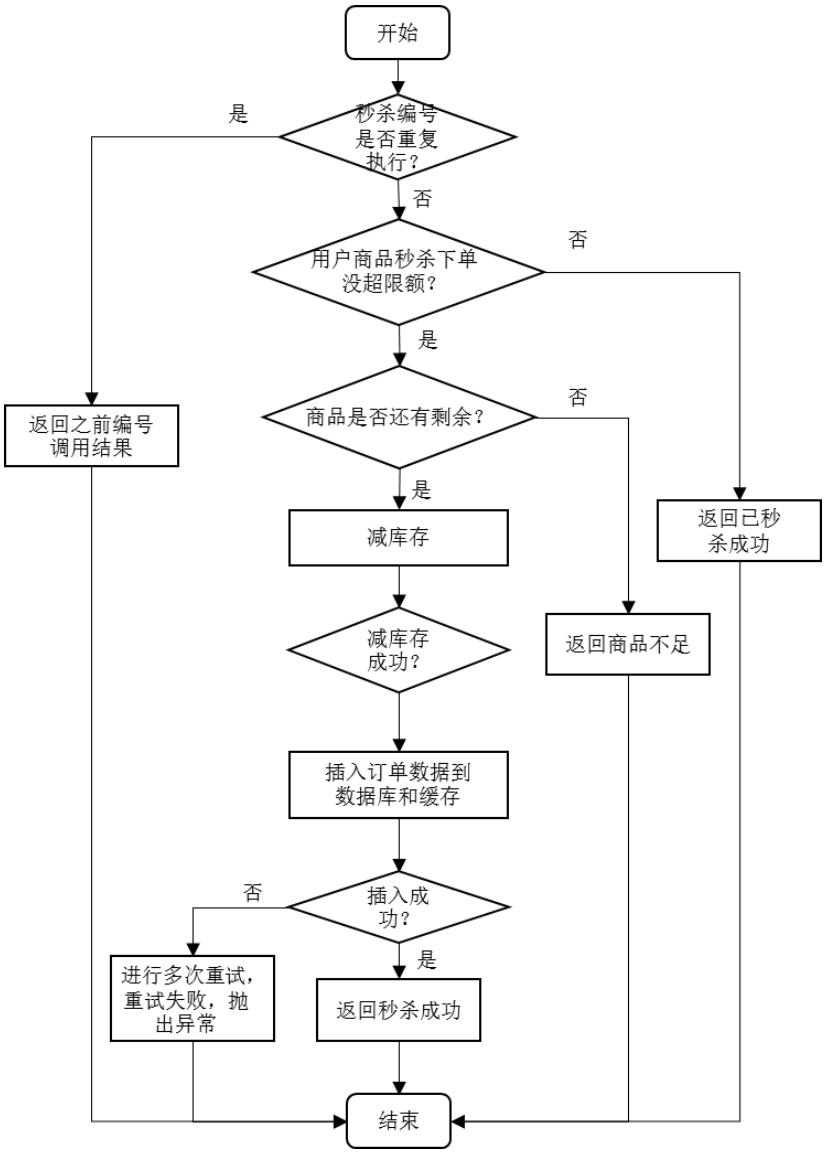


图 5- 11 秒杀下单服务代码流程图

Fig. 5-11 Seckill service code flow

5.3.5 订单服务设计实现

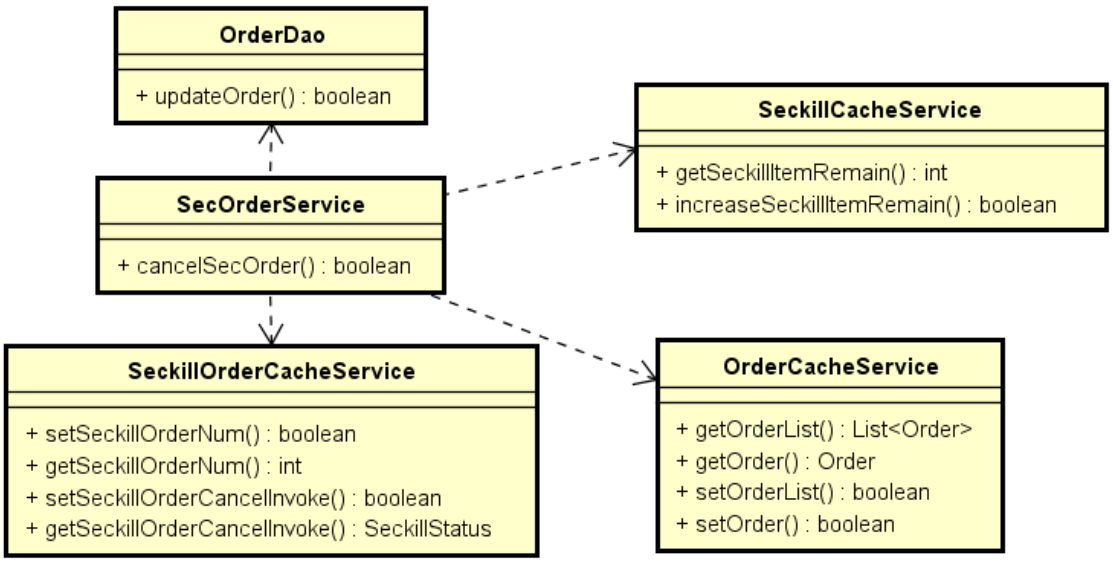


图 5- 12 订单服务类图

Fig. 5-12 Order service class diagram

订单服务类图如图 5-12 所示，由于订单列表获取服务和订单详细信息获取服务在汇金百货 O2O 交易系统[4]中已存在，并且在对订单数据库加入是否为秒杀订单、秒杀活动编号、秒杀商品编号的信息后，O2O 交易系统可以支持秒杀订单列表查看、秒杀订单查询服务的使用。秒杀订单服务仅包含取消秒杀订单功能，下面详细介绍订单服务的具体实现。

（1）SecOrderService 类：包含 cancelSecOrder ()方法，主要对 API 接入服务的取消秒杀订单服务请求进行响应。

（2）orderDao 类：updateOrder ()方法将秒杀订单信息状态变为已删除。

（3）SeckillCacheService、OrderCacheService、SeckillOrderCacheService 类：getSeckillItemRemain()、getOrderList()、getOrder()、setOrderList()、setOrder()、setSeckillOrderNum()、getSeckillOrderNum()方法与秒杀下单中方法作用类似，可以获取和设置相应信息；increaseSeckillItemRemain()方法会尝试添加商品库存；setSeckillOrderCancelInvoke()和 getSeckillOrderCancelInvoke()方法作用同秒杀下单服务中的 getSeckillOrderInvoke()和 getSeckillOrderInvoke()方法，都是为了避免取消订单操作超时重发后，系统进行多次订单取消操作，导致用户购买商品数量多次减一。

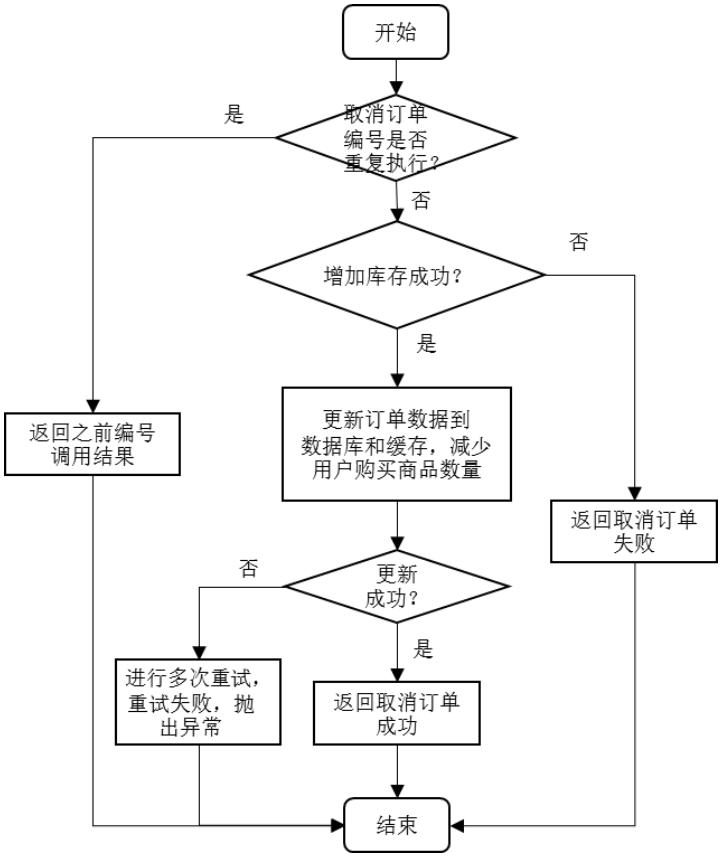


图 5- 13 取消订单代码流程图

Fig. 5-13 Cancel Order code flow

取消订单流程如图 5-13 所示，取消订单先判断操作是否重复后，再尝试添加库存，如果添加库存失败，则返回取消订单失败，如果添加库存成功，则在数据库中将订单信息标记为取消状态，并在缓存中减少该用户购买商品数量，同时更新缓存中的订单详细信息为取消状态。

5.4 商品库存同步服务设计实现

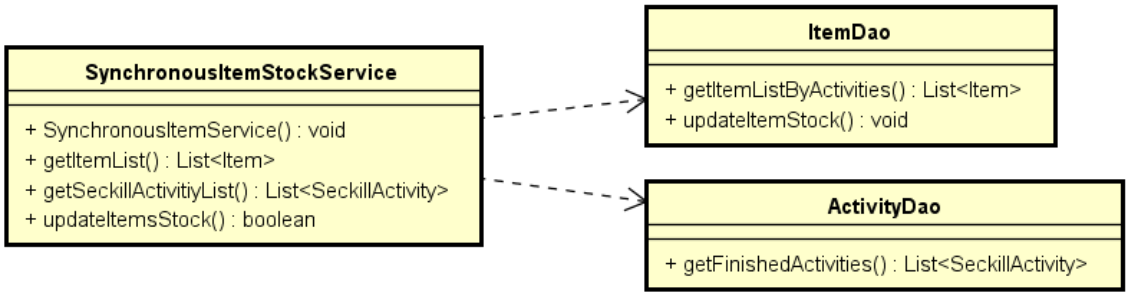


图 5- 14 商品库存同步服务类图

Fig. 5-14 Synchronous item stock service class diagram

商品库存同步服务类图如图 5-14 所示，商品库存同步服务不属于业务层服务，它是一个独立的运行的服务，它是基于 ScheduledExecutorService 定时机制搭建的，商品库存同步服务将在每晚 3 点进行秒杀商品库存的更新，下面详细介绍订单服务的具体实现。

（1）SynchronousItemStockService 类：getSeckillActivitiyList()获取已经结束的秒杀活动列表，getItemList()方法获取秒杀活动中的所有商品，updateItemsStock()方法进行商品库存更新。

（2）ItemDao 类：getItemListByActivities()方法根据获取参与秒杀活动的商品，updateItemStock()方法更新商品库存。

（3）ActivityDao 类：getFinishedActivities()方法获取已经结束的秒杀活动。

5.5 Nginx 限流的详细设计与实现

商品秒杀系统主要限制 IP 的访问速率并对某些访问频率过高的 IP 进行一段时间的封禁，具体实现如下：

网站的请求经过 LVS，LVS 采用源地址散列调度进行负载均衡，将同一 IP 地址映射到同一台 nginx 服务器，请求到达 nginx 服务器后，nginx 服务器使用 lua 脚本和单机 redis 进行防刷流程如下：

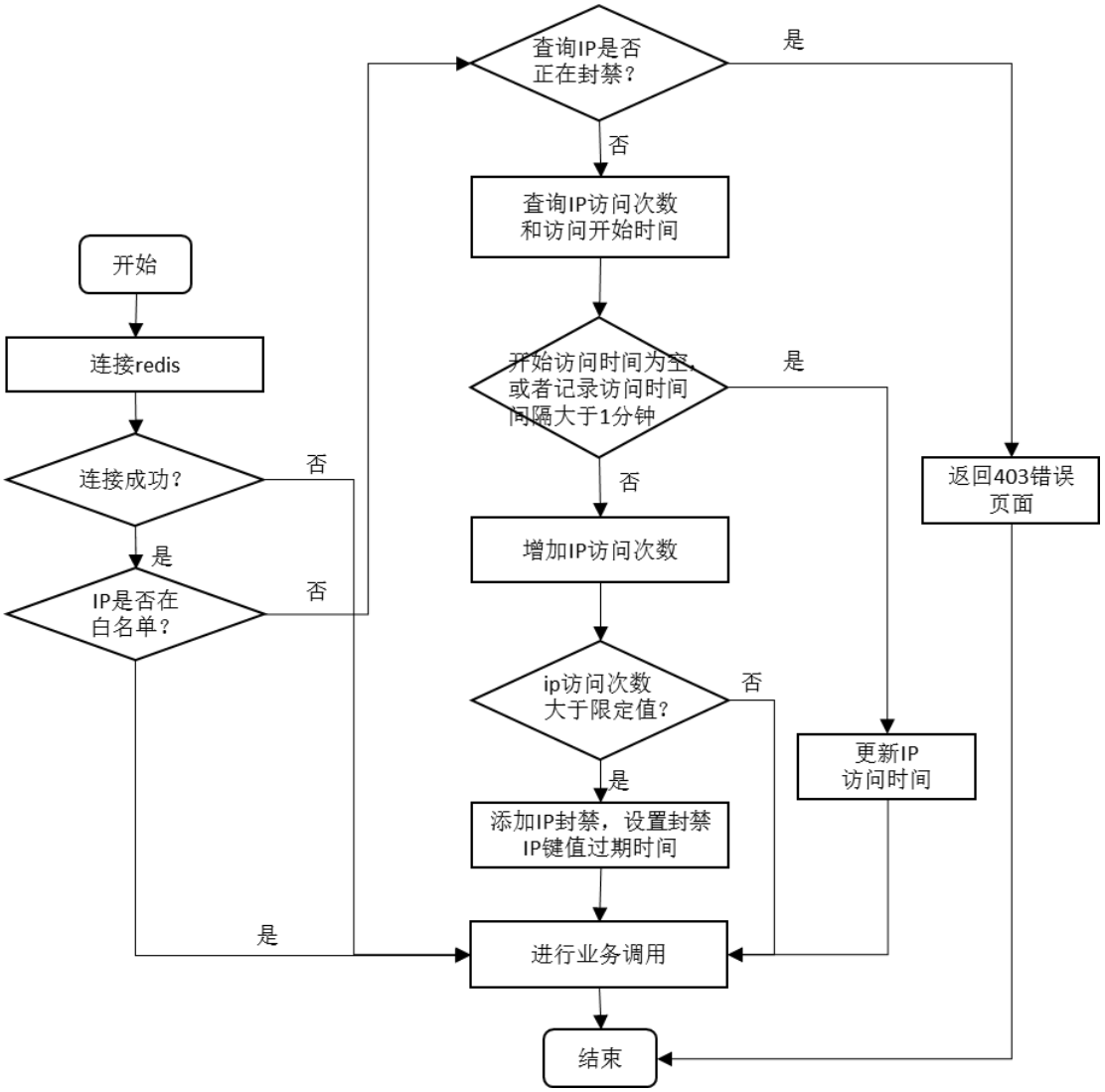


图 5- 15 Nginx+lua+redis 限流代码流程图

Fig. 5-15 Nginx + lua + redis limit-flow code flow

Nginx+lua+redis 限流代码流程图如图 5-15 所示，如果访问 ip 不在白名单中，并在一分钟内访问次数大于一分钟 150 次，将在 6 小时内禁止此 ip 进行访问请求，6 小时后 ip 封禁键值过期，ip 将又可以进行访问。由于 LVS 使用源地址 hash 策略转发，所以相同 IP 地址一般会转发到同一台 nginx 限流服务器上，所以针对源 IP 地址进行限流是可行的。这种方式与 nginx limit 模块默认提供的功能不同，limit 模块仅对连接数进行限制，但并不进行封禁。对于某些进行恶意秒杀或者网络攻击的用户，系统需要对此进行 IP 封禁一段时间，IP 封禁模块也方便拓展成和

验证码模块搭配使用。

5.6 本章小结

本章主要对面向消费者的秒杀系统进行架构设计以及详细设计与实现，架构设计包含对系统进行网关层架构设计、业务层架构设计和缓存集群架构设计。详细设计与实现包含业务层的设计与实现、库存同步方案的设计与实现和网关限流方案的设计与实现。

第六章 电商平台秒杀系统测试与分析

面向电商平台秒杀系统测试包含，对测试集群环境进行部署，对秒杀系统后台和面向消费者的秒杀系统进行功能测试，对面向消费者的秒杀系统的秒杀下单请求进行并发性能测试、可用性能测试和可配置性能测试。

6.1 测试环境部署

由于本系统的网关层使用负载性能较高的 LVS/DR 模式进行负载均衡，本系统的网关设备和系统设备必须处于同一个网络段内，本实验主要在局域网内进行仿真测试，局域网交换机 LAN 接口参数都为 1000M。

（1）秒杀系统数据库测试环境部署

首先进行秒杀系统的 Mysql Fabric 数据库集群的部署。数据库中包含已有的汇金百货 O2O 系统数据库表和秒杀系统的数据库表。由于秒杀系统进行了大量的缓存优化，在对秒杀下单操作进行性能测试时，数据库操作只需插入订单数据。本测试使用一台性能较好的计算机搭建数据库，将数据库最大连接数调为 10000，使数据库插入操作不会成为系统瓶颈，共使用两台机器搭建数据库集群。

表 6-1 数据库实验环境

Table 6-1 Experiment environment of database

机器 IP	处理器	内存	硬盘	操作系统
a@192.168.31.222	Xeon E5-2620 2.10GHz*32	32G	1T	Ubuntu 16.04
b@192.168.31.81	Intel i5-3470 CPU 3.20GHz*4	4G	500G	Ubuntu 16.04

上述两台机器分别安装 Mysql Utilities 1.5.6 和 Mysql Fabric 套件，a 为数据节点，b 为管理节点。

（2）秒杀系统后台测试环境部署

由于仅对秒杀系统后台进行功能测试，这里复用秒杀系统数据库的机器 a 和机器 b，在机器 a 和机器 b 上安装 nginx 和秒杀系统后台后进行系统功能测试。

（3）面向消费者的秒杀系统测试环境部署

对面向消费者的秒杀系统需要进行功能测试和性能测试，本测试在上述数据

库环境的基础上，另搭建 11 台主机，编号为 s1-s11，在局域网内做仿真测试，实验环境如下：

表 6- 2 系统实验环境

Table 6-2 Experimental environment of system

机器 IP	处理器	内存	硬盘	网卡	操作系统
s1 s2	Xeon E5-2620 2.10GHz*32	32G	1T	1000M	Ubuntu 14.04
s3	Intel i5-3470 CPU 3.20GHz*4	4G	500G	100M	Ubuntu 16.04
s4 s5 s6 s7 s8	Intel i5-3470 CPU 3.20GHz*4	4G	500G	100M	Ubuntu 16.04
s9 s10 s11	Intel i5-5200 CPU 2.20GHz*8	8G	500G	100M	Ubuntu 16.04

s1，s2 机器由于网卡配置较高，单位时间接收流量较高，用来搭建 LVS 转发机器集群。s3 用来搭建 zookeeper 集群，由于 zookeeper 集群不参与服务转发，对系统并发性能影响较小，为节约机器，使用 s3 同时搭建 zookeeper 集群和 dubbo-admin 监控管理器。在 s4 到 s8 共 5 台机器上安装 nginx，redis sentinel 集群，以及秒杀下单服务和 API 接入服务。在 s4，s5 机器上安装商品消息获取服务和商品详细信息获取服务，在 s6，s7 机器上安装订单服务。在 s9，s10， s11 上安装 jmeter 分布式压测工具。

6. 2 秒杀系统功能测试

由于文章篇幅的限制，对于系统功能测试用例描述在此不详细介绍，仅对测试用例名称和测试结果进行简单描述。

6. 2. 1 秒杀系统后台测试用例设计

表 6- 3 秒杀系统后台测试用例

Table 6-3 Test Case of seckill system’s backend

测试代号	测试名称	测试用例描述	测试结果描述
S1	Test_SeckillActivityList	管理员查看秒杀活动列表	正常
S2	Test_SeckillActivity	管理员查看秒杀活动详细信息	正常

续表 6-3

测试代号	测试名称	测试用例描述	测试结果描述
S3	Test_AddActivity	管理员添加一个秒杀活动	正常
S4	Test _EditActivity	管理员编辑一个秒杀活动	正常
S5	Test_RemoveActivity	管理员删除秒杀活动	正常
S6	Test_Manager_AddItem	管理员添加一个秒杀商品	正常
S7	Test_SalesMan_AddItem	供应商添加一个秒杀商品	正常
S8	Test_ItemList	供应商/管理员查看秒杀商品列表	正常
S9	Test _EditItem	管理员/供应商编辑一个秒杀商品	正常
S10	Test _EditItem	管理员/供应商查看一个秒杀商品	正常
S11	Test_SubmitSecKillItem	供应商申请秒杀商品参与活动	正常
S12	Test_AllowOrNotAllow SecKillItem	管理员批准/不批准一个秒杀商品申请	正常
S13	Test_RemoveItem	管理员/供应商删除秒杀商品	正常
S14	Test_BlackUserList	管理员查看黑名单用户列表信息	正常
S15	Test_AddOrEditBlack ListUser	管理员添加黑名单用户信息	正常
S16	Test_RemoveBlack ListUser	管理员删除黑名单用户	正常
S17	Test_SalesManList	管理员查看秒杀供应商列表	正常
S18	Test _EditSalesMan	管理员查看秒杀供应商详细信息	正常
S19	Test_AllowSecKill SalesMan	管理员批准/不批准一个秒杀供应商申请	正常

续表 6-3

测试代号	测试名称	测试用例描述	测试结果描述
S20	Test_AllowSecKill SalesMan	秒杀供应商申请参与秒杀活动	正常
S21	Test_SecManagerList	系统管理员查看商场管理员列表	正常
S22	Test_SecManager DetailInfo	系统管理员查看商场管理员详细信息	正常
S23	Test_AddSecManager	系统管理员添加商场管理员	正常
S24	Test_DelSecManager	系统管理员删除商场管理员	正常
S25	Test_EditSecManager	系统管理员编辑商场管理员信息	正常

以上是对秒杀系统后台设计的 25 个测试用例，测试结果均为正常运行。在第一次进行功能测试时，S11 用例测试结果不正常，当秒杀供应商提交审核秒杀商品后，依旧可以修改秒杀商品信息，不满足系统后台的功能需求，在进行修正后，S11 能通过测试用例。

6.2.2 面向消费者秒杀系统测试用例设计

表 6-4 秒杀系统测试用例

Table 6-4 Test Case of seckill system

测试代号	测试名称	测试用例描述	测试结果描述
C1	Test_SecActivityInfo	获取指定的秒杀活动信息	正常获取
C2	Test_ItemListInfo	获取商品列表详细信息	正常获取
C3	Test_ ItemBaseInfo	获取商品基本信息	正常获取
C4	Test_ ItemDetailInfo	获取商品详细信息	正常获取
C5	Test_Seckill_Repeat	用户反复秒杀商品	秒杀规定数量商品后，不能秒杀
C6	Test_SecKill_OutOfSale	用户秒杀库存不足的商品	秒杀失败
C7	Test_seckill_BlackList	黑名单用户秒杀商品	被禁止参与活动
C8	Test_SecKill_Fast	模拟 http 请求快速秒杀商品	被加入黑名单

表 6-4 是对面向消费者秒杀系统设计的 8 个测试用例，在进行第一次测试时，C5 用户秒杀下单用例测试异常，出现当用户秒杀商品超出秒杀商品数量限制时，依旧能秒杀商品，以及商品秒杀成功后，用户的订单列表信息不能正常获取等问题，经过修正后，测试结果均为正常运行。

6.3 秒杀系统的非功能测试

6.3.1 测试前准备工作

本测试延用 6.1 节中部署的测试环境。在实际测试过程中，为减少网络开销，调节 nginx 和 API server 之间的长连接保持时间为 1 分钟，将所有 nginx 映射到同一台机器 API 接入服务的权重调为 100，其余调为 1。将测试机器 ip 地址加入 nginx 防刷白名单中，在 s4 到 s8 共 5 台机器上安装 nginx 转发，API 接入服务和秒杀下单服务，并将 nginx 映射到本机 API 接入服务的最大空闲连接数调为 500，以此减少新建连接数量。

6.3.2 并发性能测试与结果分析

对面向消费者的秒杀系统进行并发性能测试，测试将使用分布式 jmeter 测试工具，模拟汇金百货的实际的业务场景，让 10000 名用户在 1s 内随机对 100 件库存为 10 的商品进行秒杀下单操作，每个用户被限制只能购买一件，秒杀请求将重复进行 10 次。

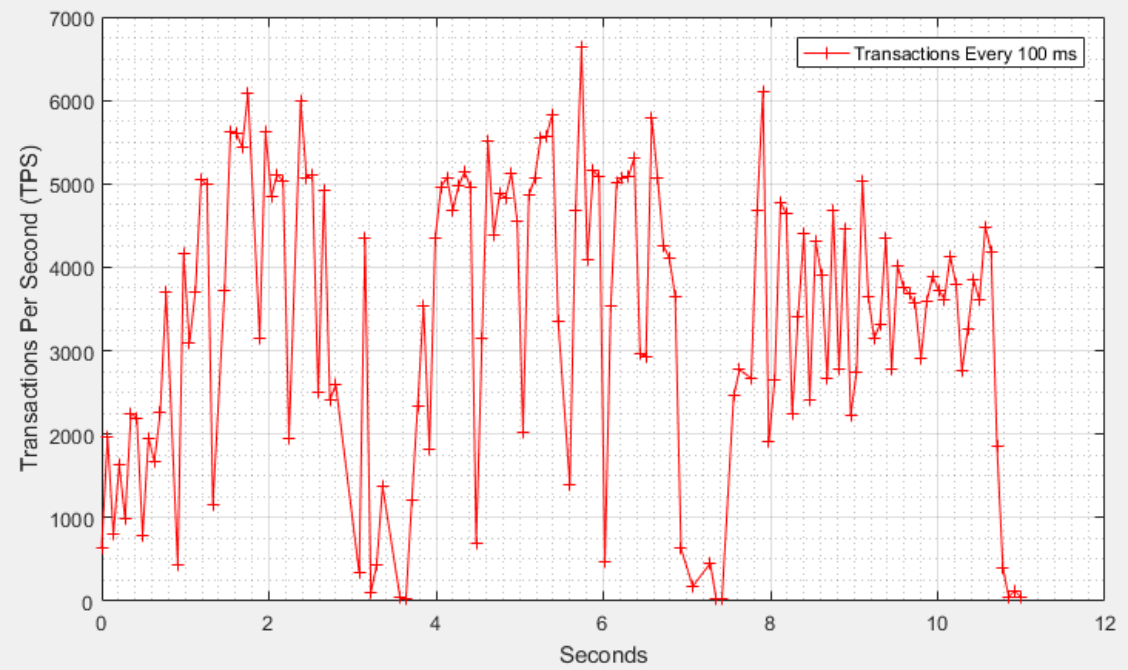


图 6- 1 秒杀系统性能测试图

Fig. 6-1 Performance test chart of seckill system

系统 TPS 数据如图 6-1 所示，系统平均响应时间为 2572ms，最高响应速度为每秒 6689 个事务，第一秒内的 TPS 为 1500 左右，其余时间内的平均 TPS 为 3000 左右。这是由于商品在第一秒内已经秒杀完成，后面所有请求将被快速失败。平均每秒的处理事务量(Trasactions Per Second)为 2878.3 次。在整个秒杀测试的过程中，系统处理错误率为 0%。系统错误率指的是在 jmeter 压力测试报告中的错误率，即由于超时等原因测试失败的请求百分比。

6. 3. 3 可用性能测试与结果分析

可用性测试对两种情况进行测试。第一种是模拟在秒杀下单场景中，部分 API 接入服务和秒杀下单服务发生故障的情况，测试标准是系统的 TPS 和请求错误率；第二种是模拟在极端情况下，秒杀下单服务全部出现故障，秒杀下单服务处于完全不可用的情况，测试标准是系统是否能正常运行。

在模拟 API 接入服务和秒杀下单服务部分发生故障的实验中，考虑到在秒杀活动的短暂生命周期内难以完成整个实验过程，因此秒杀活动的持续时间需要被延长。具体的实验设计如下：模拟 10000 名用户对 1000 件库存为 10000 的商品进行秒杀，并将每名用户秒杀商品限制为 10000 件，这个秒杀过程将重复 15 次。在

测试过程中，分别将秒杀下单服务和 API 接入服务停止一半后进行观察。

如图 6-2 所示，系统在开始时正常运转，其 TPS 约为 1900 左右；在 12s 时人工停止 2 台负责 API 接入服务的机器，系统的 TPS 约为 1100 左右，用户平均响应时间为 4520ms，系统处理错误率为 0%，系统依然能正常运行。

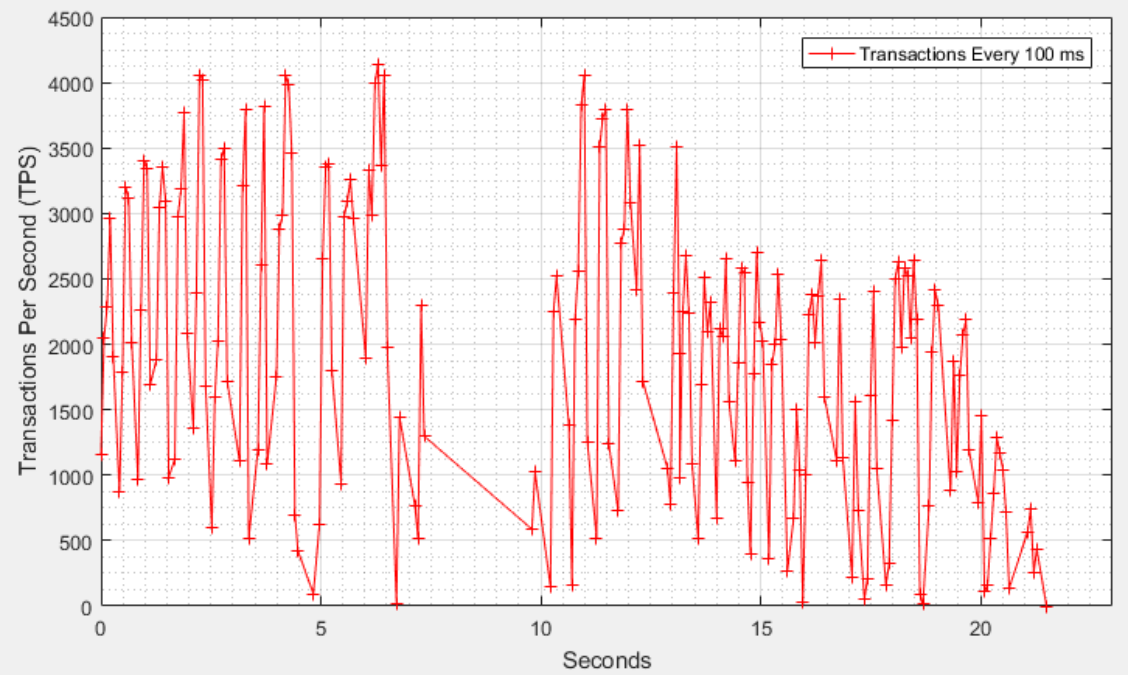


图 6- 2 API 服务一半停止后秒杀系统性能测试图

Fig. 6-2 Performance test chart when half of the API access service is dead

如图 6-3 所示，在 10s 时，将负责秒杀下单服务的机器停止 2 台后，由于 Hystix 的快速失败，系统 TPS 下降不明显，用户平均响应时间为 3456ms，系统处理错误率为 0%，系统依然能正常运行。

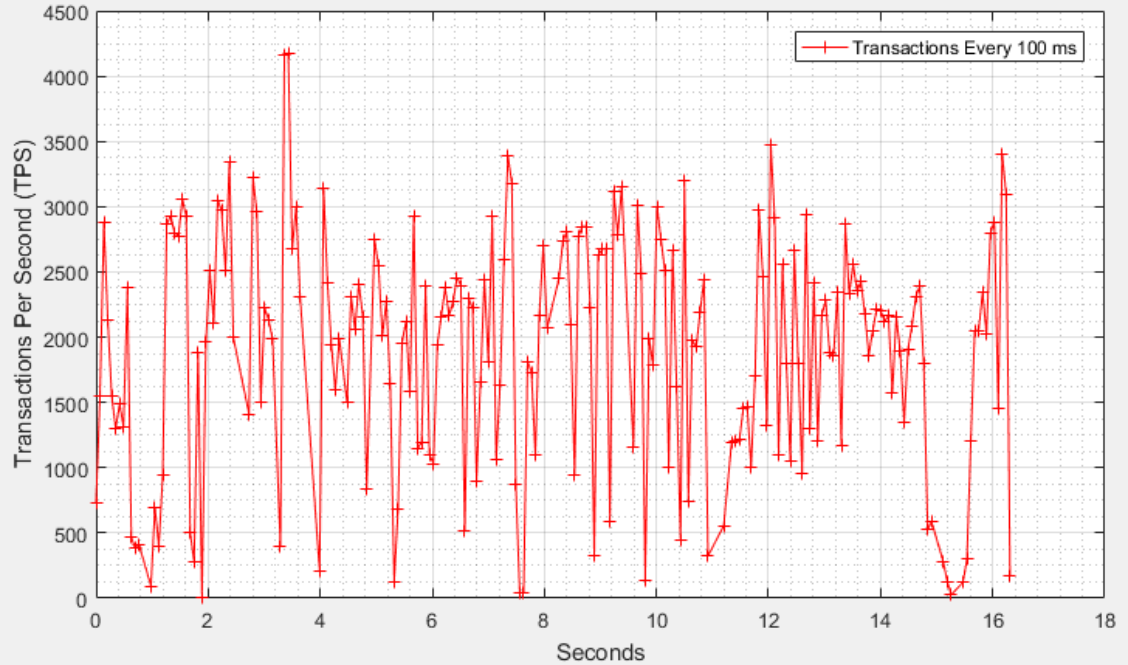


图 6- 3 秒杀下单服务一半停止后秒杀系统性能测试图

Fig. 6-3 Performance test chart when half of the seckill service is dead

在模拟秒杀下单服务全部出现故障的实验中，所有负责秒杀下单服务的机器全部被停掉。在下单服务全部失效的状态下，尝试对系统进行 10000 次秒杀下单操作，以及对商品列表信息、商品详细信息进行正常获取请求。实验结果显示，所有秒杀下单请求均返回秒杀下单失败，系统请求的错误率为 0%，商品列表信息和商品详细信息均能正常获取，这证明秒杀下单服务异常不会引发整个系统阻塞，不会影响整个系统的运行。

6. 3. 4 可配置性能测试与结果分析

对系统可配置性实验测试将模拟 10000 名用户在 1s 内对 1000 件库存为 10000 的商品进行秒杀下单请求，其中每件商品限制为 10000 件，这些秒杀请求将不断重复。在实验过程中，首先部署 0 台秒杀下单机器，添加 2 台机器部署秒杀下单服务后，对顾客秒杀下单的成功率、系统的 TPS 进行统计；再添加 3 台机器部署秒杀下单服务，对顾客秒杀下单成功率和系统的 TPS 进行统计。秒杀成功率指的是对于所有的秒杀下单请求，返回结果为秒杀成功的概率。

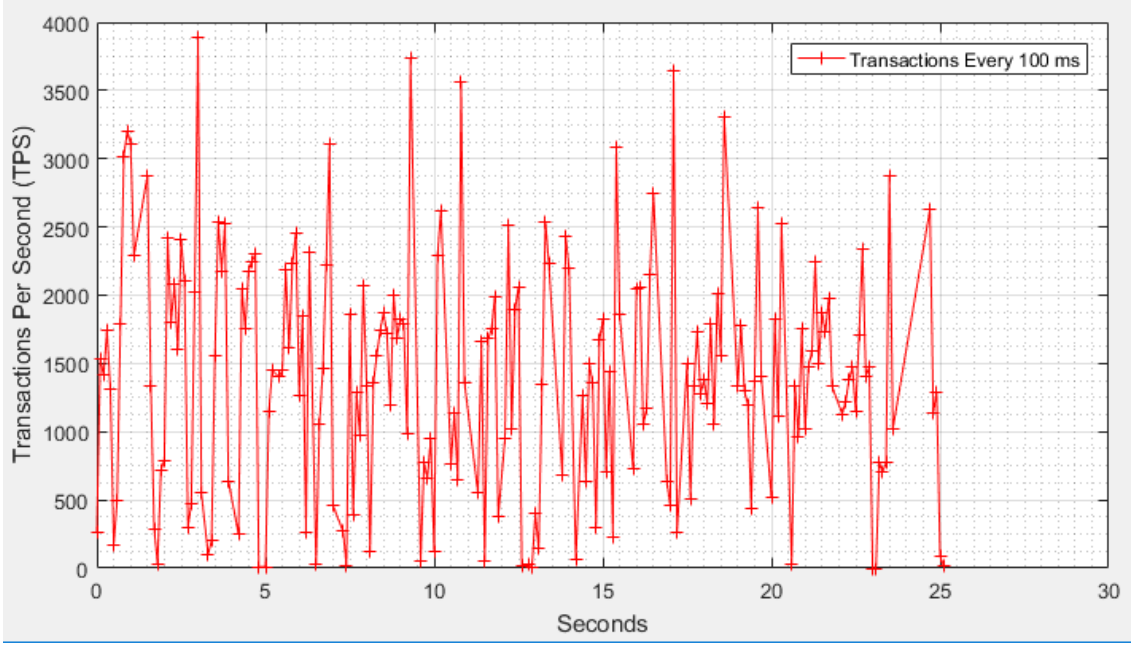


图 6- 4 系统可配置性能测试图

Fig. 6-4 System configurable performance test diagram

如图 6-4 所示，添加秒杀下单服务对系统的 TPS 影响不明显，表示系统能平滑的添加秒杀下单服务。在添加 2 台机器后，系统秒杀下单成功率为 61.9%，添加 3 台机器后，系统秒杀下单成功率为 99.8%。高秒杀下单成功率将给用户带来更好的购物体验，秒杀系统可配置性较强。

6. 4 本章小结

本章首先对测试环境进行部署后，对面向电商平台的秒杀系统进行功能测试与非功能测试，非功能测试包含对系统的并发性能、可用性能和可配置性能进行测试。测试结果表明面向电商平台的秒杀系统能满足汇金百货秒杀系统的功能需求和性能需求。系统已初步上线试运行，运行情况良好。

第七章 总结与展望

7.1 总结

秒杀活动作为一种快速吸引顾客、增加品牌影响力的销售手段，以低廉的价格，极大的刺激了消费者的购买欲望。双十一购物节的秒杀活动每年都吸引成千上万的消费者参加，交易额高达一千亿，消费者秒杀到心仪的商品的同时，也为电商带来了巨额的收益。

本文以汇金百货的秒杀系统为背景，徐家汇汇金百货是一个具有前瞻性的上海著名的线下零售企业，汇金百货推出了汇金百货 O2O 交易平台后，实现了从传统的零售交易业务场景到支持 O2O 交易业务场景的转变，能较好的运行和维护现有的正常的商品售卖业务，并拥有良好的并发性和可用性。原有的汇金百货 O2O 交易系统虽然较好的满足了现有的 O2O 交易场景的业务需求，为了更好的支持秒杀业务场景，本文对原有 O2O 系统进行新业务的拓展，主要工作包含：

(1)对面向电商平台的秒杀系统对汇金百货秒杀业务的需求进行具体分析，包含供应商、管理人员和消费者三种用户角色的需求。

(2)设计并实现了秒杀系统后台，对秒杀系统后台进行功能模块的切分后，对其进行相应的实现。

(3)设计并实现了面向消费者的秒杀系统，基于面向消费者秒杀系统高并发、高可用、可配置、可拓展的非功能需求，对秒杀系统进行详细的架构设计。架构设计包含网关层设计、分布式业务层设计和高可用的 redis 缓存层设计。系统的网关层使用 LVS/DR 模式配合 nginx 实现高效率流量分发的同时，实现了流量限制和网关层的高可用。系统的分布式业务层以 springboot 微服务框架为基础，使用 dubbo 分布式服务调用框架配合 hystrix 容错机制搭建，将系统划分为 API 接入服务、秒杀下单服务、秒杀信息获取服务、订单服务、商品详细信息获取服务，对所有服务的代码流程设计与实现充分考虑了分布式调用的幂等性、缓存的充分利用、业务降级，使得系统具有较好的可拓展性、可用性。系统的缓存层使用客户端分片技术、redis sentinel 高可用方案进行搭建，实现了 redis 缓存的高可用。

(4)对秒杀系统后台和面向消费者的秒杀系统进行全面的测试并进行测试结果

分析，包含设计测试用例对整个系统进行功能测试，和模拟正常业务的秒杀流程，对面向消费者的秒杀系统进行并发性能的测试，和高可用性能测试。测试完成后对系统测试结果进行分析，秒杀系统能较好的完成汇金百货秒杀系统的需求。

7.2 研究展望

由于时间和技术的限制，秒杀系统还可以进一步完善，具体研究方向包含：

(1)对于本文实现的面向消费者的秒杀系统将系统拆分成多个模块，且基于springboot 搭建的系统具备易部署的特性，后续的研究可以对服务调用失败率进行分析监控，实现系统的自动扩展。比如，秒杀下单服务处于繁忙的状态，服务调用失败率较高，监控系统可以获取失败率后，监测出秒杀下单服务繁忙的信息，自动在一台新的机器上部署秒杀下单服务，使得后续的请求可以快速完成秒杀下单服务的调用而不需要进行业务的降级，提高用户秒杀成功的概率。

(2)对于秒杀系统的网关层，后续可以实现验证码模块，配合限流机制一起使用，更好的对人类和机器用户进行甄别，降低误杀的概率。

(3)对秒杀系统的业务层，可以依据业务的具体情况，对线程池参数、长连接保持时间，hystrix 业务降级参数进行调节。

参考文献

[1] Wang Z, Chen C, Guo B, et al. Internet plus in China[J]. IT Professional, 2016, 18(3): 5-8.

[2] Akram U, Hui P, Khan M K, et al. Online Impulse Buying on “Double Eleven” Shopping Festival: An Empirical Investigation of Utilitarian and Hedonic Motivations[C]//International Conference on Management Science and Engineering Management. Springer, Cham, 2017: 680-692.

[3] 张小红. 电子商务营销策略之秒杀[J]. 机械管理开发, 2011(2):150-151.

[4] Wang Xiaoxi. A Clustering Strategy For Heterogeneous Server Farm In High Concurrency High Availability Scenario[C]. International Conference on Communication Systems and Computing Application Science(CSCAS), 2016

[5] LiJunnan, Wu Gang. Benchmark and Optimization Implementation for O2O Commercial System[C]. International Conference on Computation Techniques in Information and Communication Technologies(ICCTICT), 2017

[6] Chu F. NetScaler's flagship Web content manager scales up[J]. Eweek, 2003.

[7] Hochmuth P. F5, Citrix get apps up to speed[J]. Network World, 2005.

[8] Zhang W. Linux virtual server for scalable network services[C]//Ottawa Linux Symposium. 2000, 2000.

[9] Reese W. Nginx: the high-performance web server and reverse proxy[J]. Linux Journal, 2008, 2008(173):2.

[10] Tarreau W. HAProxy-the reliable, high-performance TCP/HTTP load balancer[J]. 2011-8)[2013-4]. http://haproxy. lwt. eu, 2012.

[11] Wijeratne W. Integrated Simple Intrusion Detection and Prevention System for Nginx Web Server[D]. , 2016.

[12] Hollenback P. Improving network reliability with Keepalived[J]. 2008.

[13] Fowler M, Lewis J. Microservices[J]. ThoughtWorks. http://martinfowler. com/articles/microservices. html [last accessed on February 17, 2015], 2014.

[14] Namiot D, Sneps-Sneppe M. On micro-services architecture[J]. International

Journal of Open Information Technologies, 2014, 2(9): 24-27.

[15] Newman S. Building microservices: designing fine-grained systems[M]. " O'Reilly Media, Inc.", 2015.

[16] Walls C. Spring Boot in action[M]. Manning Publications Co., 2016.

[17] Baeldung. Introduction to Dubbo [J]. http://www.baeldung.com/dubbo,2017

[18] Weisshaar B, Smith M, Bhaskaran P, et al. Service framework supporting remote service discovery and connection: U.S. Patent 6,757,262[P]. 2004-6-29.

[19] Maurer N, Wolfthal M. Netty in Action[M]. Manning Publications Company, 2016.

[20] 崔宝江, 刘军, 王刚, 等. 网络存储系统 I/O 响应时间边界性能研究[J]. 通信学报, 2006, 27(1): 70-74.

[21] Yoo H C. Comparative analysis of asynchronous I/O in multithreaded UNIX[J]. Softw., Pract. Exper., 1996, 26(9): 987-997.

[22] B. Christensen, "Introducing Hystrix for resilience engineering", Netflix Tech Blog., November 2012, [online] Available: http://techblog.netflix.com/2012/11/hystrix.html.

[23] Schmaus B. Making the Netflix API more resilient[J]. The Netflix Tech Blog, December, 2011.

[24] Christensen B. Fault tolerance in a high volume, distributed system[J]. blog, Feb, 2012.

[25] Carlson J L. Redis in Action[M]. Manning Publications Co., 2013.

[26] Xinyan W, Lijun M. Design and implementation of Redis cluster solution based on Twemproxy[J]. Electronic Test, 2016, 6: 009.

[27] 温涛. Storage system based on Redis cluster:, CN 105677251 A[P]. 2016.

[28] Macedo T, Oliveira F. Redis Cookbook: Practical Techniques for Fast Data Manipulation[M]. " O'Reilly Media, Inc.", 2011.

[29] 柳皓亮, 王丽, 周阳辰. Redis 集群性能测试分析[J]. 微型机与应用, 2016 (2016 年 10): 70-71, 78.

[30] Wang Z, Wei Z, Liu H. Research on high availability architecture of SQL and NoSQL[C]//AIP Conference Proceedings. AIP Publishing, 2017, 1820(1): 090021.

[31]苏涛. O2O 电子商务商业新模式分析[J]. 全国商情 (理论研究), 2012, 1: 34-35.

[32]证券. 徐家汇商城拟在上海南站开设汇金百货[J]. 上海百货, 2016 (1): 52-52.

[33]LIU J, DAI J. Research of lightweight Web application based on Spring MVC and iBATIS frameworks [J][J]. Journal of Computer Applications, 2006, 4: 26.

[34]Richardson L, Ruby S. RESTful web services[M]. " O'Reilly Media, Inc.", 2008.

[35]Aivaliotis D. Mastering Nginx[M]. Packt Publishing Ltd, 2016.

[36]Zhang D, Wei Z, Yang Y. Research on lightweight MVC framework based on spring MVC and mybatis[C]//Computational Intelligence and Design (ISCID), 2013 Sixth International Symposium on. IEEE, 2013, 1: 350-353.

[37]Cassen A. Keepalived: Health checking for LVS & high availability[J]. URL http://www. linuxvirtualserver. org, 2002.

[38]Omilusik D. Spring boot: U.S. Patent 4,660,299[P]. 1987-4-28.

[39]Stonebraker M. SQL databases v. NoSQL databases[J]. Communications of the ACM, 2010, 53(4): 10-11.

[40]许会元, 何利力. NodeJS 的异步非阻塞 I/O 研究[J]. 工业控制计算机, 2015, 28(3):127-129.

致 谢

不知不觉，已经在 ADC 实验室度过了两年半美好的研究生求学时光，这两年半的时间里，不论是学习研究还是生活成长，我都受益匪浅。即将离开这个地方，结束我的研究生求学生涯，我才发现对这里有着深深的不舍和眷恋。在本文项目研究和撰写的过程中，接受了许多人的帮助，在这里对他们由衷的进行感谢。

首先要感谢我的导师，吴刚副教授，感谢导师在我攻读硕士的两年半时间对我学习上的指导和生活上的关怀。我非常荣幸能成为吴刚老师的学生，吴刚老师严谨认真的工作态度，勇于创新的科研作风深深的感染了我，让我在今后的学习中能不惧挑战，砥砺前行。我的研究生毕业设计，从项目的规划搭建到文章的结构设计，都是在吴刚老师的尽心教导下努力完成的，吴刚老师在论文的前期工作中，给予我极大的鼓励、支持，后期工作中对我的论文一字一句的审核、订正让我十分感动。在生活中，吴刚老师说话做事的睿智，对生活积极乐观的态度，令我受益匪浅，是我学习的榜样。吴老师经常组织实验室活动，对学生给予极大的生活上的关心和爱护，给我的研究生生涯留下很多美好的回忆。

其次要感谢实验室所有的同学给我的帮助和支持，感谢实验室的学长学姐在学业和项目工作中给我的支持和鼓励。感谢实验室同级的小伙伴们两年半的陪伴，在研究生生涯中，一起学习，一起科研，互帮互助，共同进步，在一起的时光虽短暂却格外的美好，我会永远珍藏在心间。

我还要感谢我的家人，我的家人永远是我最坚实的堡垒，感谢他们不辞辛苦的将我培养成人，感谢他们在我每一个失落、失意的瞬间，都能在我身边温柔的安抚我，使我在长大成人之际，有一颗更平和、更安定的心去面对整个人生。

最后，感谢上海交通大学电子信息与电气工程学院，感谢所有论文评审老师和领导，感谢你们对我的论文仔细的评审和宝贵的意见，在离开学校后，我一定再接再厉不辜负你们的培养和期望！

攻读硕士学位期间已发表或录用的学术论文

[1] Liye Zhu. Configurable E-commerce-oriented Distributed Seckill System with High Availability[C]. International Conference on Information Science, Automation and Material System (ISAM),2018(第一作者，已录用)

上海交通大学
学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密□。

（请在以上方框内打“√”）

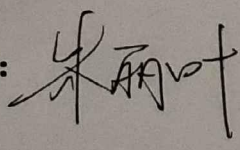
学位论文作者签名：宋丽叶　　指导教师签名：

日期：2018年 1月11日　　日期：2018年 1月11日

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文《面向电商平台的秒杀系统设计与实现》，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名： 

日期：2018 年 1 月 11 日