文章编号:1007-1423(2019)02-0095-06 DOI:10.3969/j.issn.1007-1423.2019.02.021

一种高并发电商秒杀系统的设计与实现

刘磊

(广东开放大学,广州 510000)

摘要:

针对电子商务秒杀系统的"短时间、高并发"需求,设计和实现一个健壮的、高扩展、高性能的秒杀系统解决方案。实现技术选用业界流行的 SSM 框架、Bootstrap 框架、MySQL 数据库、Redis 缓存等,重点分析影响系统高并发的瓶颈点,并给出具体的优化措施。最后,测试和应用实践表明该方案具有可用性,能有效解决秒杀系统高并发的问题。

关键词:

秒杀系统; SSM 框架; 高并发; 系统优化; 集群部署

基金项目:

广东省优秀青年教师培养项目(No.YQ2015181); 广东开放大学创新强校项目(No.2018LGCQ04-02)

1 秒杀业务分析

作为现代电商的重要促销手段,商品秒杀、抢红包这类"短时间、高并发"的需求越来越多。一个典型的电商秒杀系统围绕着商家、库存、用户、订单四者展开[1]。首先,商家会添加、调整商品库存,库存则会反映商家的发货、核帐信息;用户成功秒杀商品,库存进行减法操作,同时用户订单进行加法操作,插入一条秒杀记录明细;用户的付款、退货行为,也会反映在库存和订单。秒杀系统处理的是大量用户瞬时对有限商品的激烈竞争,它的高并发点出在前端用户秒杀商品环节,其核心是库存的高效处理。

用户每个成功的秒杀行为都包括两个动作:减去商品库存和记录购买明细,业务要求这两个动作必须同时成功,若减库存而没有记录购买明细就会出现超卖,记录了明细却没有减库存就出现了少卖,这两种现象都是不正常的。因此,减库存与记明细这两个原子操作要么都成功,要么都不成功,它们组成了一个完整的事务,秒杀业务场景具有典型的"事务"特性,如图 1 所示。要保证以上操作数据顺利存储,有两种方案:使用 RDBMS(关系型数据库)或使用 NoSQL(非关系型数据库),NoSQL 的重要场景在缓存,对事务支持较差;RDBMS 支持鲁棒性事务,业界研究表明,事务机制依然是目前最可靠的数据落地方案[2-3]。

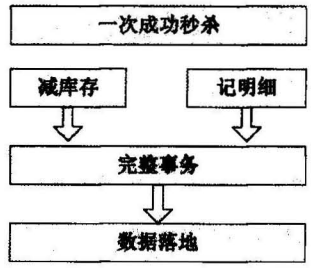

图1 秒杀的事务特性

2 系统设计

2.1 功能设计

为着重验证秒杀功能,笔者设计了一个业务具有典型性的秒杀系统[4],功能设计如下:

(1)用户注册登录:用户注册账号,使用账号登录系统。

(2)秒杀列表查询:以列表形式展示秒杀商品。

(3)秒杀详情查询:以单页面展示秒杀商品详细信息,包括秒杀倒计时、秒杀按钮。

(4)生成秒杀地址:为了避免用户作弊,即秒杀开

启前提前在浏览器输入秒杀地址进入秒杀，设计只有在秒杀开始时才能暴露秒杀地址，使用 MD5 加盐加密生成地址，防止用户撞库破解。

(5)执行秒杀：系统最核心功能，用户执行秒杀，成功则依次进行减库存和记录购买明细两个动作，同时返回秒杀成功响应给前端用户，秒杀失败也返回相应失败信息。

(6)用户中心：以列表形式展示用户成功秒杀的商品，对接付款模块，显示订单状态。

2.2 数据库设计

围绕秒杀业务和功能分析，本方案数据库设计三个核心实体对象：用户、商品和订单。用户实体记录注册登录信息，用户 ID 设置为主键；商品实体记录一件商品的名称、库存量、秒杀开始时间和结束时间等库存详情，商品 ID 设置为主键；订单记录秒杀成功后的明细，即哪个用户秒杀了哪件商品，当前状态是未付款还是已付款。根据业务分析，一个用户可以秒杀多个商品，一个商品也可以被多个用户秒杀，用户与商品是多对多的关系，同时秒杀业务要求同一用户对同一商品不能重复秒杀，也就是订单表不能插入用户 ID 和商品 ID 两者都相同的记录，解决策略是只需要将商品 ID 与用户 ID 组成订单表的联合主键即可。本秒杀系统数据库模型设计如图 2 所示。

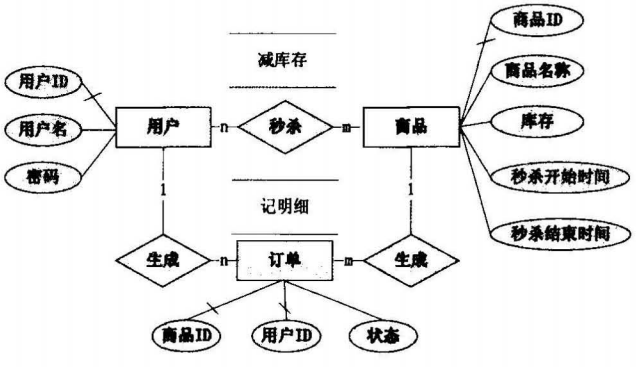

图2 数据库模型图

2.3 流程设计

用户秒杀商品的典型流程是：①首先，用户查看所有的秒杀商品列表，点击选中的秒杀进入商品详情页；②详情页除了显示商品名称、详情、开始时间、结束时间等“静态信息”外，每次加载详情页，都要动态请求服务器标准系统时间，若系统时间小于开始时间，则返回前端提示，页面使用特效展示秒杀倒计时；③若系统时间大于开始时间并且小于结束时间，则意味着秒杀进行中，系统生成秒杀地址并返回前端显示秒杀按钮，用户需要登录或提前登录执行秒杀，成功秒杀则后端数据库执行减库存和记明细两个事务操作，返回前端秒杀结果；④若系统时间大于结束时间，则意味着秒杀结束，返回前端结束信息。用户在商品详情页的交互流程如图 3 所示。

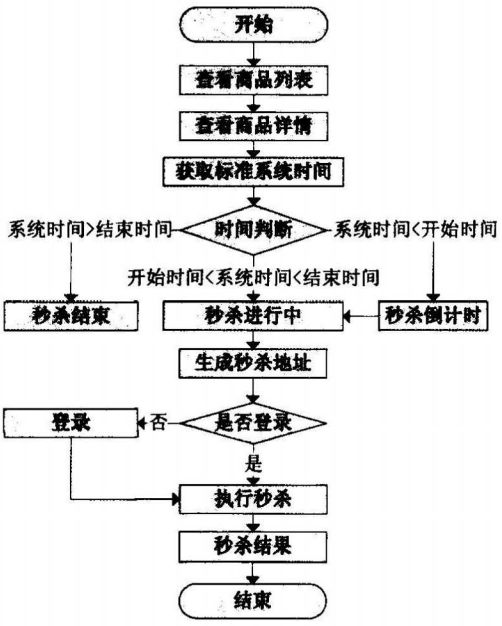

图3 用户秒杀流程图

3 技术实现

本系统实现方案采用业界流行的技术组合：Linux + Tomcat/Nginx + SSM + Bootstrap + MySQL + Redis + Maven，这套组合被阿里、京东等互联网公司广泛应用于大中型 Web 应用，非常适于开发高并发、高性能、高扩展的 Web 系统。

(1)SSM：SpringMVC、Spring、MyBatis 三大框架的完美组合，主流的 J2EE 企业级开发框架，具有轻量级、代码侵入性低、技术成熟的特点，支持典型的三层架构：DAO(数据层)、Service(业务层)、Web(表示层)[5]。MyBatis 是数据层框架，支持定制 SQL 语句，传参自由、灵活，结果集自动赋值，接口设计和 SQL 语句的分离，方便代码 Review。Spring 提供了一个统一托管对象的

容器工厂，允许通过一致的访问接口，访问工厂里的任意实例，也就是对象控制反转(IoC)；Spring 还支持声明式事务，对于高并发应用，带事务的方法往往是瓶颈所在，一不小心就可能导致数据库访问延迟，使用声明式事务可以方便地开发单一、纯净的事务方法。SpringMVC 是 Web 层框架，支持 RESTful 风格的 URL 和 MVC 开发模式。

(2)Bootstrap：简洁、直观、强悍的前端开发框架，基于 HTML5、CSS3、jQuery 技术构建，提供了导航、分页、面板等可复用的静态组件，下拉菜单、标签页、弹出框等动态插件，强大灵活的栅格布局系统，使用 Bootstrap 可以快速、高效地开发健壮和优雅的响应式静态页面[6]。

(3)MySQL：最流行的用于 Web 开发的关系型数据库，支持事务和锁机制，可以单点、主从复制、集群多种规模运行，经测试，MySQL 执行同一条 update 压力测试约为 4 万 QPS，即一秒可以卖 4 万个商品，能够顶住大部分秒杀压力，本方案使用 MySQL 持久化数据[7-8]。

(4)Redis：基于内存的 NoSQL 型数据库，经测试 Redis 性能非常高，每秒钟 SET 操作可执行 110000 次、GET 操作可执行 81000 次，本方案使用 Redis 缓存热点数据[9-11]。

(5)Maven：强大的项目构建工具，以标准化的方式管理编译、生成、发布、文档等整个项目建设生命周期，提供更佳的项目依赖和持续集成管理。

(6)Tomcat/Nginx：Tomcat 轻量、稳定，用作应用服务器，Nginx 具备强大的并发处理、灵活的反向代理能力，用作负载均衡服务器。

使用以上技术组合，编码分为 SQL 编码、DAO 层编码、Service 层编码、Web 层编码，系统各层关键接口对接情况如表 1 所示。

表 1 系统关键接口对接表

SQL	DAO	Service	Web (Restful URL)	JSP
select	queryAll	getSeckillList	/list	List.jsp
select	queryById	getById	/{seckillId}/detail	Detail.jsp
update	reduceNumber 减库存	executeSeckill 执行秒杀	/{seckillId}/{md5}/execution	Detail.jsp
insert	insertOrder 记明细			
select	queryById	exportSeckillUrl 生成秒杀地址 (MD5 加盐加密)	/{seckillId}/exposer	Detail.jsp
insert	insertUser	register	/{user}/register	Register.jsp
select	queryUser	login	/{user}/login	Login.jsp

4 高并发瓶颈分析及优化

秒杀的本质是大量用户瞬时集中竞争有限的商品，秒杀系统最大的瓶颈点在商品秒杀环节。当秒杀开启时，大量用户涌入查询商品详情页、同时点击秒杀按钮，请求流量瞬间达到高峰，若将所有请求都直接发送到服务器，将耗损服务器大量响应资源，单台服务器并发处理能力有限，严重时可能导致服务器宕机，为了保证大量请求的及时处理，使用其他服务器分流依然是最有效的应对策略。

一个成功的秒杀行为在服务器要执行两个业务逻辑：减库存、记明细，反映在数据库层面，这两个原子操作组成一个事务，要么都提交，要么都不提交，示意代码如下：

```
Start Transaction
Update 减库存
Insert 记明细
Commit/Rollback
```

大量用户在有限的时间内竞争有限商品，意味着很多用户同时去执行同一个商品 ID 的减库存操作(如：update table set num=num-1 where id=1001)，但是数据库对同一行数据修改是有行级锁机制的，即一次运行完提交后，下一个才能运行，这就让很多用户“堵塞”在 update 操作，并发量大时系统响应性能呈指数级下降，这也是秒杀业务逻辑的并发瓶颈所在。对于秒杀系统来说，高并发优化就是疏通“堵塞”、引导“分流”，让更多人更快通过“瓶颈”参与“竞争”[5-6]。

针对以上分析提出以下具体优化措施[12-13]。

4.1 使用 CDN 缓存静态化资源

CDN 即内容分发网络，通过在网络各处放置节点服务器构成智能虚拟网络，能够实时将用户请求重定向到响应最快的服务节点，使用户就近取得所需内容，开发人员可以自己搭建或租用云 CDN 服务。商品秒杀详情页是并发请求量最集中的页面，为减少服务器请求，可将详情页里静态化的资源剥离出来推送到 CDN 节点，例如 JavaScript 脚本、CSS、图片等不常变化的资源，也就是在前端缓存静态内容，这样一部分请求就被重定向到 CDN 节点，如果命中资源就不需访问后端服务器，减轻了服务器压力，也提高了响应速度。

4.2 使用 Redis 优化生成秒杀地址

详情页里有部分资源是不能缓存到 CDN 节点的，例如请求服务器标准时间，Java 执行一次 new Date()约只需 10ns，因此不会成为瓶颈。请求秒杀地址也不能

缓存到 CDN，秒杀地址为防止被猜到，是由服务端在秒杀期间动态生成的，每次查询详情页都要请求一次秒杀地址是否开启，如果开启则通过 MD5 加盐后再加密生成不可逆的地址字符串发送到前端，判断是否开启需要查询秒杀商品的开始时间和结束时间，为提高响应速度，可将秒杀商品查询结果缓存在服务端。本方案选用高性能的 Redis 用作服务端缓存，方法是首先在 Redis 查找商品，若找到则直接返回；找不到则查询数据库，同时缓存一份在 Redis。为避免单机故障，也可以做成 Redis 集群使用。

4.3 使用存储过程优化行级锁持有时间

上面分析出秒杀环节的瓶颈点在对同一商品“竞争”执行 update 操作，数据库对一行数据执行 update 操作时，会加上行级锁，此时其他 update 操作处于等待，等事务提交或回滚后释放行级锁，其他 update 操作才能依次执行，而 insert 操作是可以并发运行的，也就是行级锁持有时间阻塞了秒杀进程；Java 与 MySQL 交互，使用 Java 执行 SQL 语句的耗时，包括 SQL 语句执行耗时、发送语句到 MySQL 和返回结果的网络延迟、GC 操作耗时，一次秒杀行为的耗时计算公式为：一次秒杀耗时=update 耗时+insert 耗时+网络延迟+GC 耗时+其他耗时。所有耗时加起来，秒杀行为的总耗时就变长了，其实 Java 不慢，MySQL 也不慢，但是使用 Java 操作 MySQL，中间耗时浪费，响应就慢了[14-16]。

因此减少行级锁的持有时间和降低网络延迟即可大幅降低一次秒杀耗时，方法是：①将 insert 调到 update 之前运行，insert 可以并发执行，若 update 成功，则可以同时提交，否则，则同时回滚，只有 update 需要锁住，性能提升一倍；②将业务逻辑封装成存储过程放在 MySQL 服务端运行，Java 客户端只需要拿到执行结果即可，MySQL 本地执行存储过程是很快的，这样大部分网络延迟和 GC 耗时会被消灭掉。行级锁持有时间优化前后对例如图 4 所示。

经过多组测试，对同一行数据执行一次 update 操作，使用存储过程事务的行级锁持有时间大约为 6ms，使用 Java 客户端托管的事务行级锁持有时间大约为 40ms，相差 34ms，这意味者如果有 500 人同时竞争同一个热点商品，优化后的事务排队时间可以减少 17s（34ms×500）。

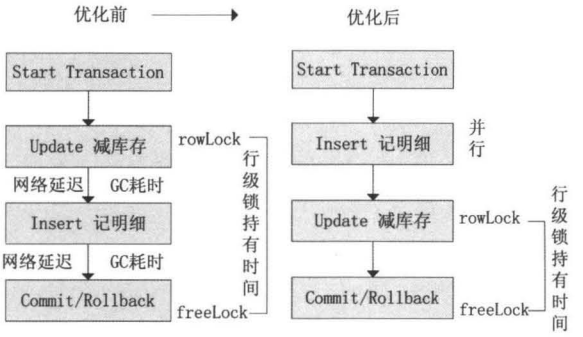

图4 行级锁持有时间优化

4.4 使用集群化部署提高系统抗压能力

理论上，一台服务器若能顶住 1000 的并发量，两台服务器就能顶住 2000 的并发量，一般来说，当上线的 Web 系统并发量达到一定数量级时，都可以通过大规模服务器集群化部署提高系统的抗压能力，也就是常说的“当一头牛拉不动时，那就用三头牛”。本方案设计的系统架构、选用的开发框架都非常适于以集群的方式运行，对于实际线上的电商秒杀系统，通过不断增加服务器可以快速提高系统的并发处理能力[17-18]。本系统线上集群化部署可以分为五层：①CDN 缓存层：缓存静态化资源；②负载均衡层：根据访问量，负责将请求智能转发到不同服务器；③Web 应用层：在 Tomcat 集群上部署相同的 Web 应用；④Redis 缓存层：缓存数据库查询结果；⑤数据库存储层：数据落地，使用主从库同步、读写库分离、分库分表等技术增强抗压能力。线上秒杀系统集群化部署方案参考图 5 所示。

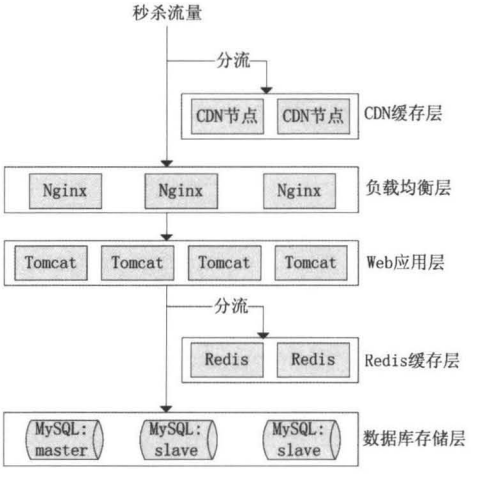

图5 秒杀系统集群化部署方案

5 测试与应用

为了测试本文设计的秒杀系统的性能、可用性和并发处理能力，笔者使用 Apache JMeter 进行了三组压力测试，为避免硬件带来的误差，三组测试环境使用了相同的机器配置，统一 CPU 为 Intel Core i5 3.3GHz、内存为 8GB，网络环境为实验室局域网。三组测试分别为：①测试一使用 1 台服务器，MySQL 与 Tomcat 安装在一起；②测试二使用 2 台服务器，1 台安装 Tomcat，1 台安装 MySQL，Web 服务器与数据库分开部署；③测试三使用 10 台服务器模拟小型集群环境，2 台安装 Nginx 作负载均衡服务器，1 台安装 Nginx 作静态资源服务器，3 台安装 Tomcat 作应用服务器，1 台安装 Redis 作缓存服务器，3 台安装 MySQL 作数据库服务器。从测试一到测试二再到测试三，不断增加服务器数量和并发数，以验证系统并发承载能力，三组测试得到的并发数、服务器 CPU 利用率、平均响应时间如表 2-4 所示。

表 2 测试一

并发数	单台服务器	平均响应时间
1 万	90%	6ms

表 3 测试二

并发数	Web 服务器	数据库服务器	平均响应时间
1.5 万	70%	60%	7ms

表 4 测试三

并发数	负载均衡服务器	Web 服务器	缓存服务器	静态资源服务器	数据库服务器	平均响应时间
5 万	45%	65%	20%	30%	50%	5ms

由此可见，通过不断增加服务器，分层部署系统，将请求流量分发到不同节点；采取一定的优化策略，将静态化资源分流，将部分查询结果缓存，将网络延迟降至最低。这些措施可有效提高秒杀系统的性能、可用性和并发处理能力。本方案实现的秒杀系统作为笔者参与的电商平台核心模块已经上线运行，通过生产环境中的并发优化和集群化部署，在多次的促销活动中，顶住了上万的并发压力，表现稳定。

6 结语

本文设计和实现的电商秒杀系统解决方案，使用 SSM 框架实现高扩展性，使用 MySQL 事务机制保证秒杀数据完整性，使用 Redis 缓存提高系统访问性能，使用集群化部署增强系统并发处理能力，本方案具有典型性，对于解决秒杀、抢红包这类“瞬时高并发抢资源”的需求有较普遍的参考价值，本文没有更多讨论秒杀系统的安全性，有待进一步研究。当然秒杀作为各大电商的促销手段，其线上实现方案不止一种，也有利用分布式 MQ 记录行为消息、再异步数据落地或者利用队列在内存中操作的方案，没有最佳和一劳永逸的方案，只有最适应业务需求的方案，读者可以根据场景灵活选用。

参考文献：

[1]邵斐. 面向电子商务的秒杀系统设计与实现[J]. 微型机与应用,2015(06).

[2]李冯筱,罗高松. NoSQL 理论体系及应用[J]. 电信科学,2012(12).

[3]徐安令. 事务处理技术在银行系统转账模块中的应用[J]. 信息系统工程,2015(02).

[4]李军锋,何明昕. 高并发 Web 航空票务秒杀系统的设计与实现[J]. 计算机工程与设计,2013(03).

[5]王艳清,陈红. 基于 SSM 框架的智能 Web 系统研发设计[J]. 计算机工程与设计,2012(12).

[6]张子杰,庄育飞. 基于 Bootstrap 和 SSH 的求职招聘系统设计与实现[J]. 软件导刊,2016(10).

[7]李现艳,赵书俊,初元萍. 基于 MySQL 的数据库服务器性能测试. 核电子学与探测技术,2011(01).

[8]邵志远,金海,唐晓辉. 基于主动 TCP 连接复制的高性能高可用 MySQL 数据库集群[J]. 计算机研究与发展,2005(06).

[9]马豫星. Redis 数据库特性分析[J]. 物联网技术,2015,03:105-106.

[10]杨艳,李炜,王纯. 内存数据库在高速缓存方面的应用[J]. 现代电信科技,2011,12:59-64.

[11]Fiebig Tobias,Feldmann Anja,Petschick Matthias .A One-Year Perspective on Exposed In-memory Key-Value Stores.9th ACM Workshop on Automated Decision Making for Active Cyber Defense (SafeConfig),OCT 24,2016.
[12]王亚楠,吴华瑞,黄锋. 高并发 Web 应用系统的性能优化分析与研究[J]. 计算机工程与设计,2014(08).
[13]李军. 高并发 Web 系统的设计与优化[D]. 北京交通大学,2009.
[14]汪维富,黄海于,陈娟,曾阳红. 基于存储过程的高性能数据库应用模型研究[J]. 计算机工程与设计,2008(10).
[15]程博,阎楚良,叶舸. 利用存储过程优化 CMS 系统查询. 微计算机信息,2008(12).
[16]郑华. 基于存储过程的 Java 持久层的解决方案.微计算机信息,2006(24).
[17]包立辉,黄彦飞. 高并发网站的架构研究及解决方案[J]. 计算机科学,2012(S2).
[18]梅华威,张铭泉,李天. 高并发高负载网站系统架构研究[J]. 计算机与网络,2009(14).

作者简介：
刘磊(1984-),男,硕士,讲师,高级信息系统项目管理师,高级程序员,研究方向为项目管理、互联网 Web 应用开发、大型网站架构、数据库技术、大数据技术与应用等
收稿日期:2018-10-30　　修稿日期:2018-11-19

Design and Implementation of a High Concurrency Seckill System

LIU Lei

(Guangdong Open University, Guangzhou 510000)

Abstract:
In view of the short time and high concurrent demand of the electronic commerce system, designs and implements a robust, high scalability and high performance system solution. Selects the popular SSM framework, Bootstrap framework, MySQL database, Redis cache and so on, and focuses on analyzing the bottleneck of the system, and gives the specific optimization measures. Finally, the test and application results show that the scheme is feasible, and can effectively solve the problem of high concurrency.
Keywords:
Seckill System; SSM Framework; High Concurrency; System Optimization; Cluster Deployment