**A22**

**银行产品秒杀系统设计**

|  |  |
| --- | --- |
| 团队编号: | 2100094 |
| 队 名: | 摘月 |
|  |  |

2022.4.7

目录

[1.环境 3](#_Toc101115755)

[1.1 技术点 3](#_Toc101115756)

[1.2 目录说明 3](#_Toc101115757)

[2.运维 4](#_Toc101115758)

[3.数据库 4](#_Toc101115759)

[3.1 表介绍 4](#_Toc101115760)

[3.2 建表语句 7](#_Toc101115761)

[3.3 主要字段 8](#_Toc101115762)

[4.接口 8](#_Toc101115763)

[5.业务 8](#_Toc101115764)

[5.1 基本业务流程描述 8](#_Toc101115765)

[5.2 脚本防刷功能实现说明 8](#_Toc101115766)

[5.3 秒杀地址隐藏功能实现说明 10](#_Toc101115767)

[5.4 初筛功能实现说明 12](#_Toc101115768)

[5.5 秒杀功能实现说明 12](#_Toc101115769)

[6.亮点 13](#_Toc101115770)

[6.1 秒杀商品定时定量发布 13](#_Toc101115771)

[6.2 简单灵活的风险决策引擎 14](#_Toc101115772)

[6.3 百度云身份证识别与风险检测 18](#_Toc101115773)

[7.测试 18](#_Toc101115774)

## 1.环境

### 1.1 技术点

Springboot、Redis、Mybatis-Plus、RocketMQ、Guava令牌桶、ApachePoi、JWT、JavaMail、Vue、axios、vite、less等。

### 1.2 目录说明

1.2.1后端程序

Jar包位于：/sxapp/sxappopt/app/server/zy-server.jar；

源码位于：2100094-摘月-【 A22】 银行产品秒杀系统设计【 三湘银行】-项目补充材料/A22\_2100094\_摘月/code/server.zip。

src.main.java下有两个文件夹：com.second\_kill、resources，

在com.second\_kill下共有7个文件夹：

* config：存放本项目的配置类，例如：redis配置文件、RocketMQ配置文件等；
* controller：存放控制层类；
* entity：存放实体类，例如：客户属性、管理员属性；
* interceptor：存放拦截器，例如：JWT拦截、令牌桶拦截；
* mapper：存放数据库映射类，用于和resources下的\*mapper.xml配合使用；
* service：存放业务层类，用于处理业务逻辑；
* utils：用于存放工具类，便于开发。

在resources 下共有1个文件夹、1个文件：

* mapper文件夹：编写sql语句的xml文件，与com.second\_kill下的mapper配合使用；
* application.yml：配置相关信息，例如：数据库配置信息、redis配置信息。

1.2.2前端程序

源码位于2100094-摘月-【 A22】 银行产品秒杀系统设计【 三湘银行】-项目补充材料/A22\_2100094\_摘月/code/client-web.zip、2100094-摘月-【 A22】 银行产品秒杀系统设计【 三湘银行】-项目补充材料/A22\_2100094\_摘月/code/admin-web.zip

管理端dist文件位于：/sxapp/sxappopt/app/admin/dist

客户端dist文件位于：/sxapp/sxappopt/app/client/dist

代码目录

* src
  + apis：接口封装
  + assets：静态资源
  + components：通用组件封装
  + router：路由配置
  + utils：工具类函数
  + views：页面目录
* vite.config.js / vue.config.js：项目工程化配置

管理端zy-web-admin\src\utils\request.js中baseURL为请求基础路径。

客户端zy\_client\_web\src\utils\request.js中baseURL为请求基础路径。

## 2.运维

后端程序：后端项目位于/sxapp/sxappopt/app/server/zy-server.jar，进入该目录后，运行nohup java -jar zy-server.jar & 即可启动。

前端项目：已部署于/sxapp/sxappopt/app/，nginx模块配置位于/sxapp/sxappopt/conf/nginx/modules/zy.conf。

## 3.数据库

### 3.1 表介绍

描述数据表及表用途，描述每张表中的主要字段。下方为demo。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 表名 | 中文名 | 主要字段 | 主要字段描述 | 备注 |
| 1 | admin | 管理员信息表 | id  name  username  avatar  phone  password  salt  create\_time | 主键id  真实姓名  用户名  头像  手机号  密码  盐值  创建时间 | 无 |
| 2 | client | 客户信息表 | id，name，id\_card，salt，username，  password，avatar，phone，create\_time，  balance，attribute | 主键id  真实姓名  身份证  盐值  用户名  密码  头像  手机号  创建时间  余额  客户属性 | 客户属性：  用于存储工资、是否大于18岁等 |
| 3 | deposit\_favor | 存款商品收藏 | id  user\_id  good\_id  create\_time | 主键id  用户id  创建时间 | 无 |
| 4 | deposit\_good | 存款商品表 | id  name  price  total\_number  remain\_number  start\_time  end\_time  restrict\_number  total\_price  image  is\_filter  description  rule\_list  status  repeat\_times  repeat\_delay  total\_repeat\_times | 主键id  商品id  单价  总数  剩余数量  开始时间  结束时间  每人限购数量  总价  商品照片  是否进行初筛  描述  规则列表  商品的出售状态  循环添加间隔  剩余添加次数  总共循环次数 | repeat\_times，repeat\_delay，  total\_repeat\_times  这三个字段用于定时定量发布秒杀组。共有20000份秒杀产品参与到活动中，但不一次性全部秒杀完成，而是将这20000份产品分为5天秒杀完成，1天秒杀4000份，这种秒杀机制既灵活，并且因某些原因未能参加活动，或者未秒杀成功的客户可以有多次机会抢购。 |
| 5 | loans\_good | 贷款商品表 | id  name  price  total\_number  remain\_number  start\_time  end\_time  restrict\_number  total\_price  image  is\_filter  description  rule\_list  status  repeat\_times  repeat\_delay  total\_repeat\_times | 主键id  商品id  单价  总数  剩余数量  开始时间  结束时间  每人限购数量  总价  商品照片  是否进行初筛  描述  规则列表  商品的出售状态  循环添加间隔  剩余添加次数  总共循环次数 | 同deposit\_good表 |
| 6 | deposit\_order | 存款商品订单表 | id  good\_id  create\_time  user\_id  number  total\_price  good\_name | 主键id  商品id  创建时间  用户id  购买数量  总价  商品名称 | 无 |
| 7 | loans\_order | 贷款商品订单表 | id  good\_id  create\_time  user\_id  number  total\_price  good\_name | 主键id  商品id  创建时间  用户id  购买数量  总价  商品名称 | 无 |
| 8 | deposit\_result | 存款初筛结果表 | id  user\_id  good\_id  result  reason  create\_time | 结果ID  用户ID  商品ID  初筛结果  原因（仅拒绝时）  创建时间 | 无 |
| 9 | loans\_result | 贷款初筛结果表 | id  user\_id  good\_id  result  reason  create\_time | 结果ID  用户ID  商品ID  初筛结果  原因（仅拒绝时）  创建时间 | 无 |
| 10 | rule | 规则条目表 | id  name  description  type  value  control  variable | 规则ID  规则名  规则描述  判断方法  用于判断的常量  处理方式(通过/拒绝)  驱动变量 | 驱动变量为存在于用户属性中的值，初筛中将会用该变量的值与用于判断的常量进行比较判断（判断方法取决于control），并返回结果 |
| 11 | ruleGroup | 规则组表 | id  name  description  rule\_list | 规则组ID  规则组名  规则组描述  规则列表 | 用于自动填入规则 |
| 12 | dash\_board | 仪表盘信息表 | id  content | 主键id  统计数据的年月日json字符串 | 无 |
| 13 | loans\_favor | 贷款收藏表 | id  user\_id  good\_id  create\_time | 主键id  用户id  创建时间 | 无 |

### 3.2 建表语句

见sql文件夹。

### 3.3 主要字段

下方表格。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段中文名 | 字段所在表格 | 字段英文名 | 备注 |
| 1 | 用户账户 | client | username | 与用户存储与同一表中 |
| 2 | 用户余额 | client | balance | balance |
| 3 | 内部户账户 | client | username | id为0的记录 |
| 4 | 用户名 | client | username | 无 |
| 5 | 密码 | client | password | 无 |
| 6 | 订单号 | deposit\_order  /loans\_order | id | 无 |
| 7 | 初筛结果 | deposit\_result  /loans\_result | result | 存放pass/reject |
| 8 | 重复发布间隔 | deposit\_good  /loans\_good | repeat\_times | 用于自动重新发布商品 |

## 4.接口

在线文档（样式精美）访问链接: <https://www.apifox.cn/apidoc/shared-c4ea323d-9f6b-45bc-aca0-4cba575630ca> 访问密码 : U4Wgxg33，md版本见api.md。

## 5.业务

### 5.1 基本业务流程描述

具体功能及业务详见《2100095—摘月—【 A22】 银行产品秒杀系统设计【 三湘银行】—项目详细方案》。

打开52.82.45.193:8888/admin 进入后台管理界面，默认管理员用户名：admin密码：123123，登陆后即可使用管理端功能。

打开52.82.45.193:8888进入客户端界面，默认客户用户名：gu\_777，密码：123123，登录后即可使用客户端功能。

### 5.2 脚本防刷功能实现说明

#### 5.3.1 实现原理

传统的IP封禁策略是使用Nginx的deny功能封禁IP，为了保障系统的安全性与稳定性，IP封禁需要较高的实时性，而deny之后需要系统重载Nginx配置，重载过程需要切换worker进程，频繁重载的过程会影响系统的的性能。

对比之下，Linux防火墙（iptables）主要工作与网络层，针对TCP/IP数据包实施过滤和限制，典型的包过滤防火墙，基于内核编码实现，具有非常稳定的性能和更高的效率。

iptables 的作用：为包过滤机制的实现提供规则，通过各种不同的规则，来告诉netfilter对来自某些源以及前往某些目的或具有某些协议特征的数据包是如何进行处理的。

本项目通过shell脚本统计、分析Nginx日志，识别出异常流量IP，将其标记为恶意用户，使用iptable将其封禁。

在iptables 中filter表新建输入规则，将恶意IP加入黑名单，从而实现在网络层即可封禁恶意IP的功能。

实现方式如下：

max\_access\_num=1500

list=$(tac /sxapp/sxappopt/logs/nginx/access.log \

    | awk 'BEGIN{ "date -d \"-1 minute\" +\"%H:%M:%S\"" \

    | getline min1ago } { if (substr($4, 14) > min1ago) print $0;else exit }' | tac \

    | awk '{print $1}' | sort -rn | uniq -c | awk '{print $0}' \

    | awk '{if($1>"'$max\_access\_num'")print ""$2""}')

for i in $list; do

    iptables -I INPUT -s $i --dport 80 -j DROP

done

Nginx在涉及大流量时，会发生非常庞大的日志文件，包含access.log和error.log，日志会随着连接不断增加，到无限大。如果日志文件太大，会导致Nginx运行缓慢，卡顿，也是存储资源的浪费。

因此，本项目通过shell脚本+Linux的定时任务，实现了无需重启Nginx进程即可完成日志平滑切分的功能。

实现方式如下：

access\_log\_path=/sxapp/sxappopt/logs/nginx/access.log

access\_backup\_path=/sxapp/sxappopt/logs/nginx/log\_backup/access\_$(date +%Y%m%d -d 'yesterday').log

cp $access\_log\_path $access\_backup\_path && echo >$access\_log\_path

error\_path=/sxapp/sxappopt/logs/nginx/error.log

error\_backup\_path=/sxapp/sxappopt/logs/nginx/log\_backup/error\_$(date +%Y%m%d -d 'yesterday').log

cp $error\_path $error\_backup\_path && echo >$error\_path

#### 5.3.2 使用方式

输入 crontab -l 查看当前定时任务，其中:

0 4 \* \* \* /sxapp/sxappopt/shell/log\_backup.sh

为每日凌晨4点定时切分nginx日志，可使用 crontab -e 将其关闭，

为保证压测正常进行，系统已关闭IP封禁功能，若要启动，请使用crontab -e

输入 \* \* \* \* \* /sxapp/sxappopt/shell/black\_list.sh 保存后使用crontab -l 查看是否已开启。系统将每分钟执行一次日志分析并对异常IP进行封禁。该功能同样使用 crontab -e 关闭。

对于已封禁IP，输入iptables -L -n -v --line-numbers

即可查看，输出如下：

# iptables -L -n -v --line-numbers

# Chain INPUT (policy ACCEPT  packets,  bytes)

# num   pkts bytes target     prot opt in     out     source               destination

#               DROP       all  --  \*      \*       45.77.183.135        0.0.0.0/

#               DROP       all  --  \*      \*       120.239.199.152      0.0.0.0/

#               DROP       all  --  \*      \*       47.52.132.105        0.0.0.0/

使用iptables -D INPUT ( 例如：iptables -D INPUT 1 就是删掉上面的第一条 45.77.183.135 ) 进行单个IP解封，使用iptables –flush 进行所有IP解封。

### 5.3 秒杀地址隐藏功能实现说明

#### 5.3.1 实现基本思路

通过点击F12打开浏览器控制台，就能在点击抢购按钮后获取抢购接口的链接。所以会有客户编写抢购脚本对接口进行访问。为防止这种情况的发生，团队决定使用MD5算法，根据用户id、商品id、UUID字符串以及点击时间生成特定的MD密钥，存入Redis中缓存。下单时客户将会携带一个MD5密钥作为抢购接口。链接的一部分进行访问，将此MD5密钥与Redis中正确的密钥进行对比，若对比成功则可进行下单；若对比失败则返回404页面。具体流程如下图所示。

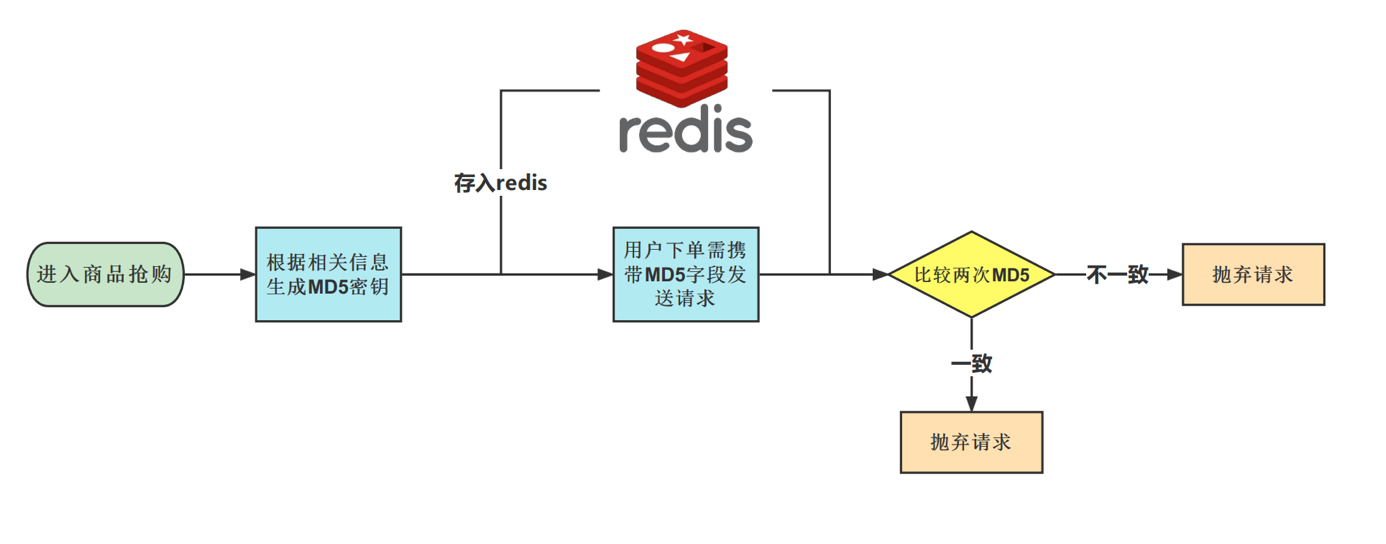


图 1秒杀地址隐藏实现方案

#### 5.3.1 具体实现方法与效果

在请求地址中获取到MD5密钥后，通过Redis查询数据库中存在的正确的MD5密钥进行对比，若对比后不匹配，或者用户通过脚本（未携带MD5密钥）直接进入抢购页面，则跳转到404页面，如图2所示。对比成功后才可以进入下单页面。



图2 404页面

### 5.4 初筛功能实现说明

详见6.2。

### 5.5 秒杀功能实现说明

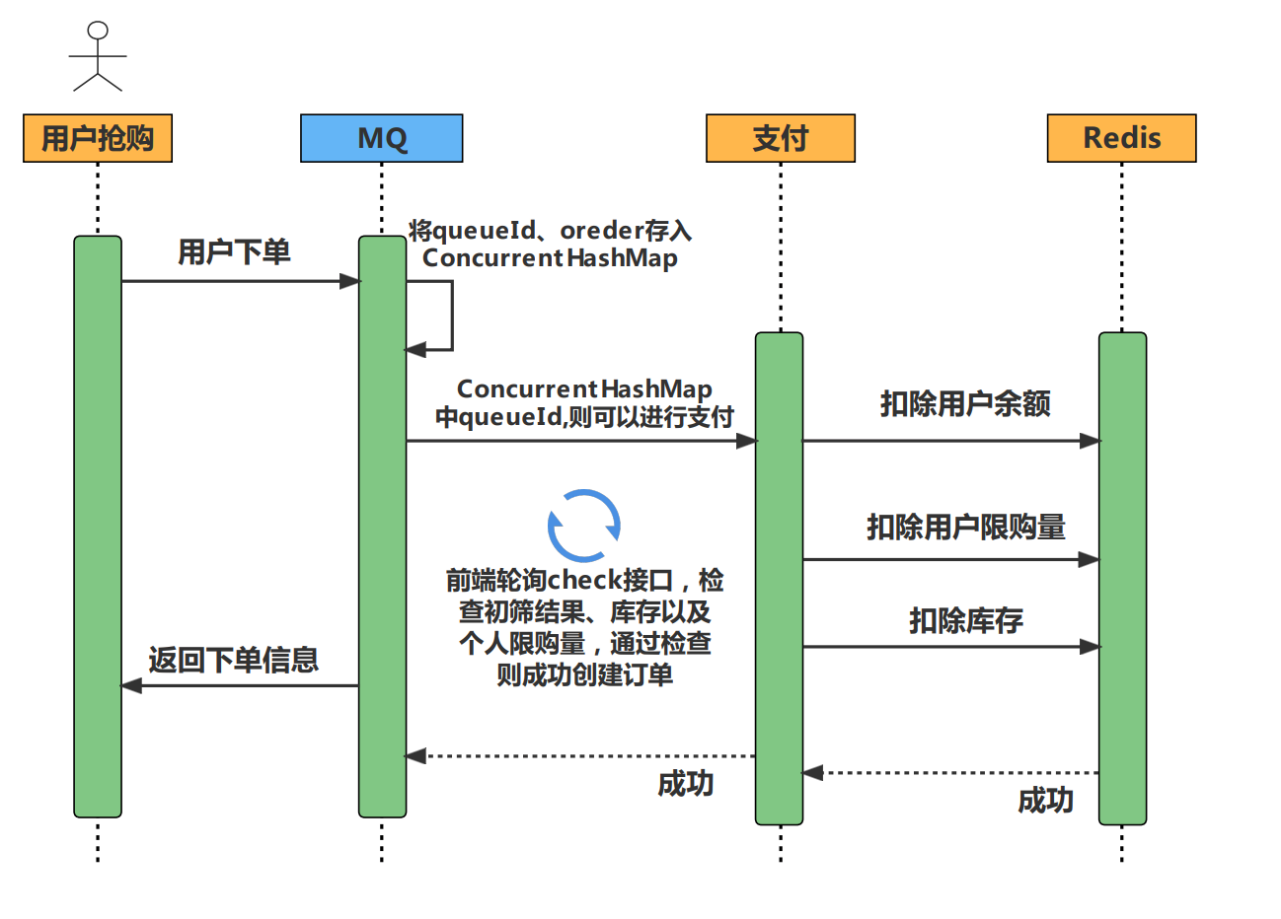
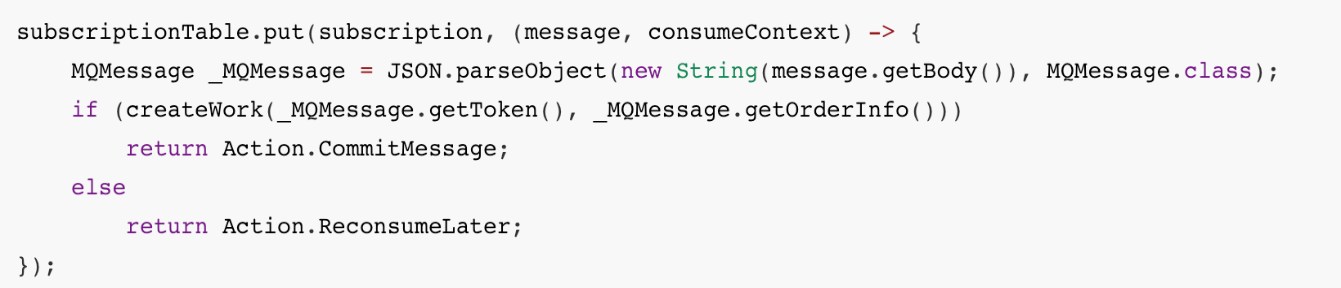
本系统对秒杀开始的时间进行了严格的控制，在秒杀尚未开始的时候，用户无法进行秒杀。秒杀开始后，用户开始下单，后端将生成的预订单信息用RocketMQ的生产者发送给消费者进行处理。在RocketMQ的消费者中调用了其他方法，用来将信息存入特定的数据结构：以queueId为键、预订单信息为值存入ConcurrentHashMap中。（ConcurrentHashMap是HashMap的一个线程安全的、支持高效并发的版本，在多线程环境下，操作HashMap会导致各种各样的线程安全问题，比如在HashMap扩容重哈希时出现的死循环问题，脏读问题等。在ConcurrentHashMap中，无论是读操作还是写操作都能保证很高的性能）。在另一边，前端对“/client/order/check”接口进行1秒1次的轮询，用来检查ConcurrentHashMap是否已经存在给定的queueId，若已经存在，则对购买者的初筛结果、限购量以及库存进行检查，若全部通过，才能成功创建订单。用户的下单购买流程如图3所示。

图3 下单流程图

另外，团队考虑到如下情况：在库存仅剩1份，而用户发出的下单请求为购买2份，则用户在下单时仅会支付部分金额。

以下为消费者订阅消息代码，具体代码可查看团队提交的源码。



­



## 6.亮点

### 6.1 秒杀商品定时定量发布

考虑到银行业务中可能存在如下业务要求：共有20000份秒杀产品参与到活动中，但不一次性全部秒杀完成，而是将这20000份产品分为5天秒杀完成，1天秒杀4000份，这种秒杀机制既灵活，并且因某些原因未能参加活动，或者未秒杀成功的客户可以有多次机会抢购。以下为实现方案：

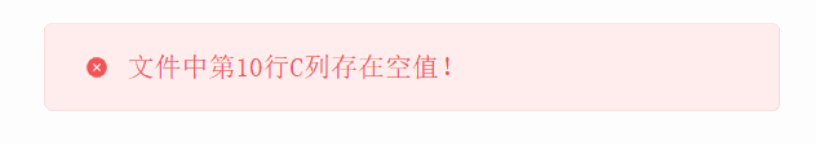
* 给每份商品添加repeat\_times(剩余发布次数)、repeat\_delay(两次秒杀之间所隔的天数)、repeat\_total\_times(总共发布次数)三种属性。
* 发布第一次秒杀活动前10分钟，将商品信息存入redis当中，每一次活动结束，更新repeat\_times以及活动的开始时间与结束时间（根据repeat\_delay来判断下一次活动开始时间）。
* 后台设置一个定时器，定时将redis持久化到MySQL数据库中以便后台管理系统查看详细数据。

### 6.2 简单灵活的风险决策引擎

#### 6.2.1 用户信息Excel上传

为了获取用户经济收入、年龄、是否工作等相关信息，团队设计在后台管理系统中以Excel表格的形式储存用户的相关信息并上传进行处理。用户信息表如下图所示。

关于处理Excel文件，团队采用了Apache提供的Apache POI，用于处理基于 Office Open XML 标准 (OOXML) 和 Microsoft 的 OLE 2 复合文档格式 (OLE2) 的各种文件格式。

管理员可以在管理后台下载用户信息表的模板，且用户的姓名与已存在的用户属性已经被自动导入。后端将用户姓名与用户的属性作为一个json字符串存入MySQL数据库中。另外，若存入的数据中有某个字段的数据为空，则页面会提示某行某列为空，提示管理员去修改Excel，保障了信息的正确性。出错提示如下。

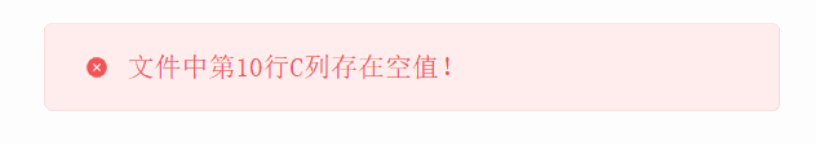


图4 出错提示图

#### 6.2.2高性能高可靠的初筛引擎

初筛引擎是初筛系统的核心，该引擎提供了以多种灵活的方案处理初筛数据的能力。初筛展示在客户端体现为提示用户“可以购买“或“不可购买”，而在管理端可以看到所有曾经触发过初筛的客户及其初筛结果的记录表，方便对其客户的组成进行分析。

初筛引擎中的每个初筛工作可以由以下事件触发：

* 商品即将开始售卖，此时执行初筛预处理，缓解售卖过程中的系统压力。
* 商品信息变更以及客户信息变更，且商品已经完成预初筛，此时将删除预初筛结果并再次执行预初筛。
* 用户注册，商品添加。

当商品的预初筛被触发时，该预初筛请求会被加入到预初筛执行队列中，等待预初筛进行。此时，如果预初筛线程未在运行，则将启动预初筛线程，否则在队列中等待被预初筛。预初筛的进入队列时，会先检查队列中有没有同一商品的其他预初筛请求。如果有，则将把当前请求和前一请求合并以减少用时，否则将当前请求加至队尾。

图5 用户属性表示例

#### 6.2.3初筛规则

初筛工作的基本单元是规则，规则指示了满足某一条件或不满足某一条件的客户可否购买对应商品。例如，在规则组为下方所示时，客户的初筛结果如下表所示（结果A对应规则A、结果B对于规则B）。

* 规则A 工资 < 2000元/月，控制：不通过
* 规则B 年龄 < 60周岁，控制：通过

表6-1初筛结果示例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用户及属性 | 结果A | 结果B | 最终结果 |
| 客户A  工资：1500元/月  年龄：25周岁 | 不允许 | 允许 | 不允许 |
| 客户B  工资：15000元/月  年龄：65周岁 | 允许 | 不允许 | 不允许 |
| 客户C  工资：8000元/月  年龄：35周岁 | 允许 | 允许 | 允许 |

#### 6.2.4初筛流程

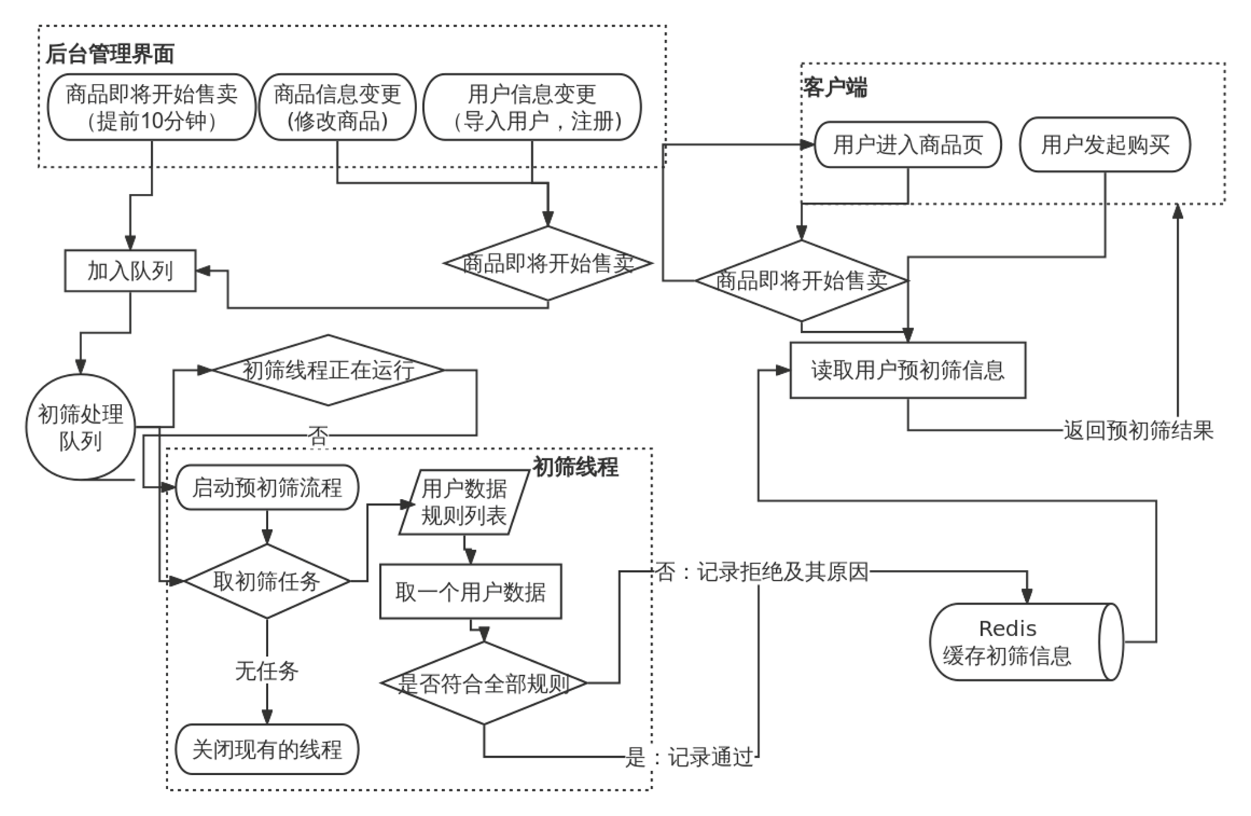


图6初筛流程图

当用户访问时，若商品已经开启初筛，则会读取已经处理的预初筛结果，并将其返回给用户，并向其展示初筛审核结果和原因（如果有）。初筛后被拒绝的用户无法购买该产品。初筛具体流程图如图6所示。

#### 6.2.5初筛反馈（邮件通知）

设置完初筛后，在商品开始售卖前会进行预处理。预处理时，如果程序遇到了下面的情况，则初筛将会发生错误：

* 用户的属性信息尚未被提供（需要通过Excel上传）
* 用户的属性信息中没有对应的驱动变量

发生错误后系统会向管理员发送一封邮件，其中说明了错误发生的商品，原因，

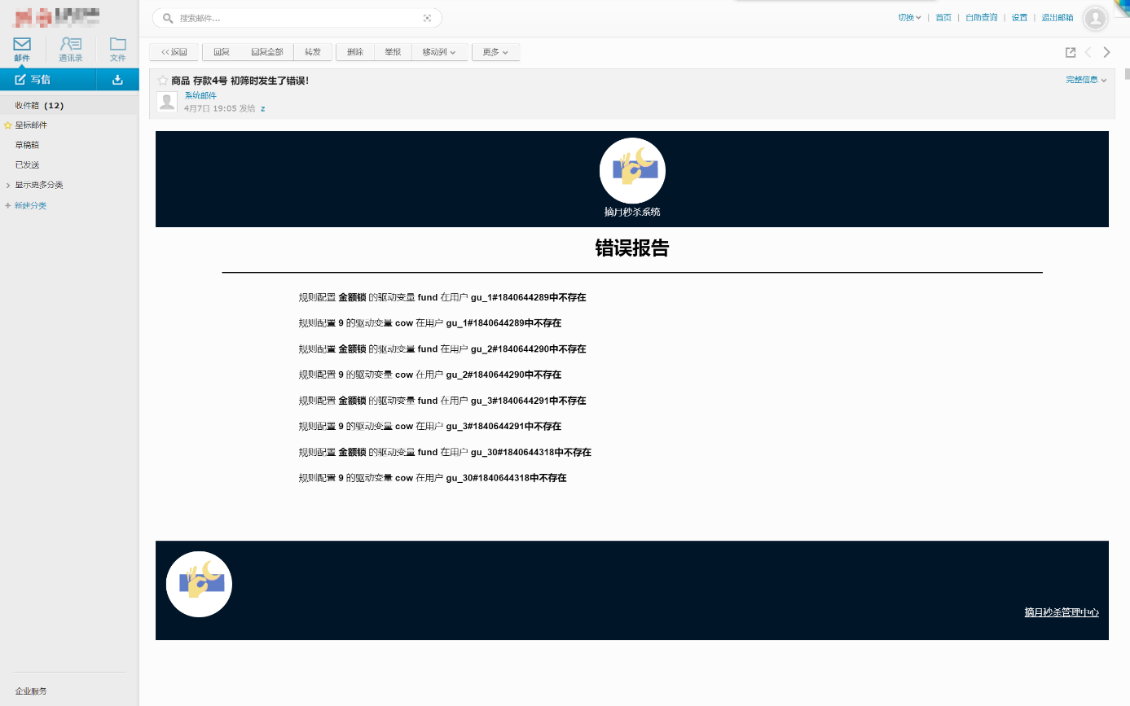
用户和驱动变量。管理员可以在商品正式开始售卖之前再次对信息进行更改以修正错误，保证商品售卖正常进行。邮件通知如图7所示。

图7 错误报告邮件通知图

### 6.3 百度云身份证识别与风险检测

#### 6.3.1使用场景

应赛题要求，客户注册时需输入客户姓名、客户身份证等核心信息。团队在开发本系统时，为客户的安全与方便考虑，团队采用了百度云的卡证识别接口，对客户上传的身份证进行信息识别。让客户检查无误后插入到数据库中。并且，若客户未经过身份证的信息认证，则无法进行秒杀活动。

#### 6.3.2风险检测与解决方案

支持对二代居民身份证正反面所有8个字段进行结构化识别，包括姓名、性别、民族、出生日期、住址、身份证号、签发机关、有效期限，识别准确率超过99%;同时，支持对用户上传的身份证图片进行图像质量和风险检测，可识别图片是否为复印件或临时身份证，是否被翻拍或编辑。出去对企业以及客户的安全性考虑，团队启用了对身份证的风险检测。

当开启对身份证的风险检查后，百度云将会对上传上来的身份证图像进行检查，并在返回参数中添加 risk\_type 字段，以下是对 risk\_type 的说明：

* normal：正常身份证
* copy：复印件
* temporary：临时身份证
* screen：翻拍
* unknown：其他未知情况

团队将会对copy（复印件）、screen（翻拍）、unknown（其他未知情况）进行拦截，阻止这些疑似非法途径进行身份证认证的客户进行认证信息。下面是身份证认证的流程：

1. 客户首次进入系统，会要求录入身份证信息。客户提交身份证正面照片后，将会提交身份证图像至百度云，百度云卡证识别接口在识别后会返回相应的json字符串。
2. 后台获取到json后进行解析，对copy（复印件）、screen（翻拍）、unknown（其他未知情况）进行拦截，使其无法添加信息。
3. 用户在检查完身份证信息是否正确后提交。

## 7.测试

#### 7.1.1测试目的

本次测试目的是为了给团队开发的银行产品秒杀系统进行压力测试，确定团队的银行产品秒杀系统在测试环境下是否能满足用户的并发量以及在相应并发数情况下平台响应情况。本次性能测试利用压力测试工具Jmeter模拟并发用户对系统进行压力测试，对其处理能力进行性能评估。

#### 7.1.2测试流程说明

本次测试的流程如下：

* 大批量用户登录秒杀系统客户端。
* 进入获取MD5密钥（详见4.3 MD5算法生成动态链接）。
* 用户购买相应商品（Jmeter中提前配置），进行下单操作。
* 订单查询轮询，查询成功后再进行支付
* 用户支付

测试流程如图7-1 测试流程图

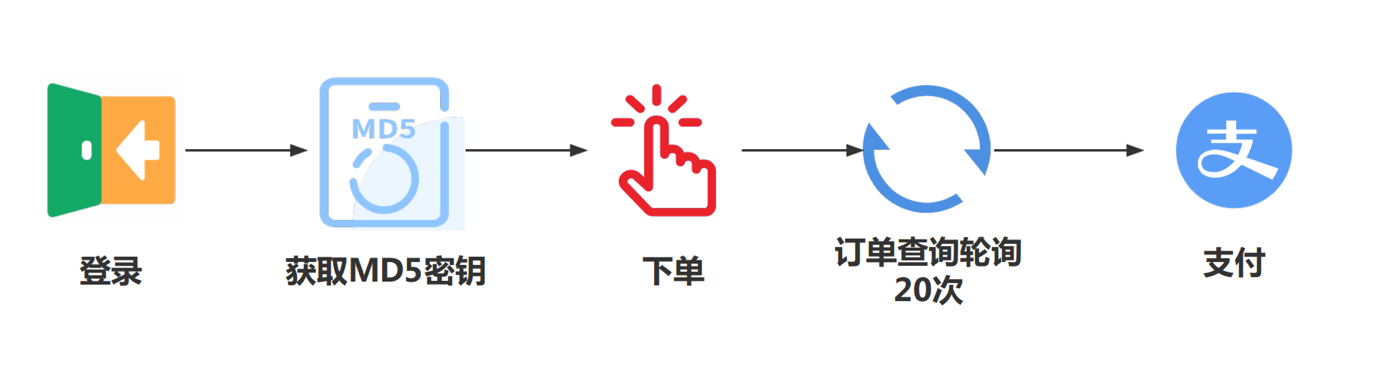


图8 测试流程图

表7-1 测试环境表

|  |  |
| --- | --- |
| ip | 配置信息 |
| 52.82.45.193 | Linux 2核4G |

在经过单机测试之后，团队为模拟更大的并发场景，借用了学校的机房进行分布式测试。测试场景如图7-3 分布式测试场景。为获取机房中的机器的ip地址，组员编写了易语言脚本，利用极域电子教室系统一键发布获取所有测试机器的ip地址。所用易语言脚本的伪代码如图7-2 易语言脚本伪代码。



图 9 易语言脚本伪代码

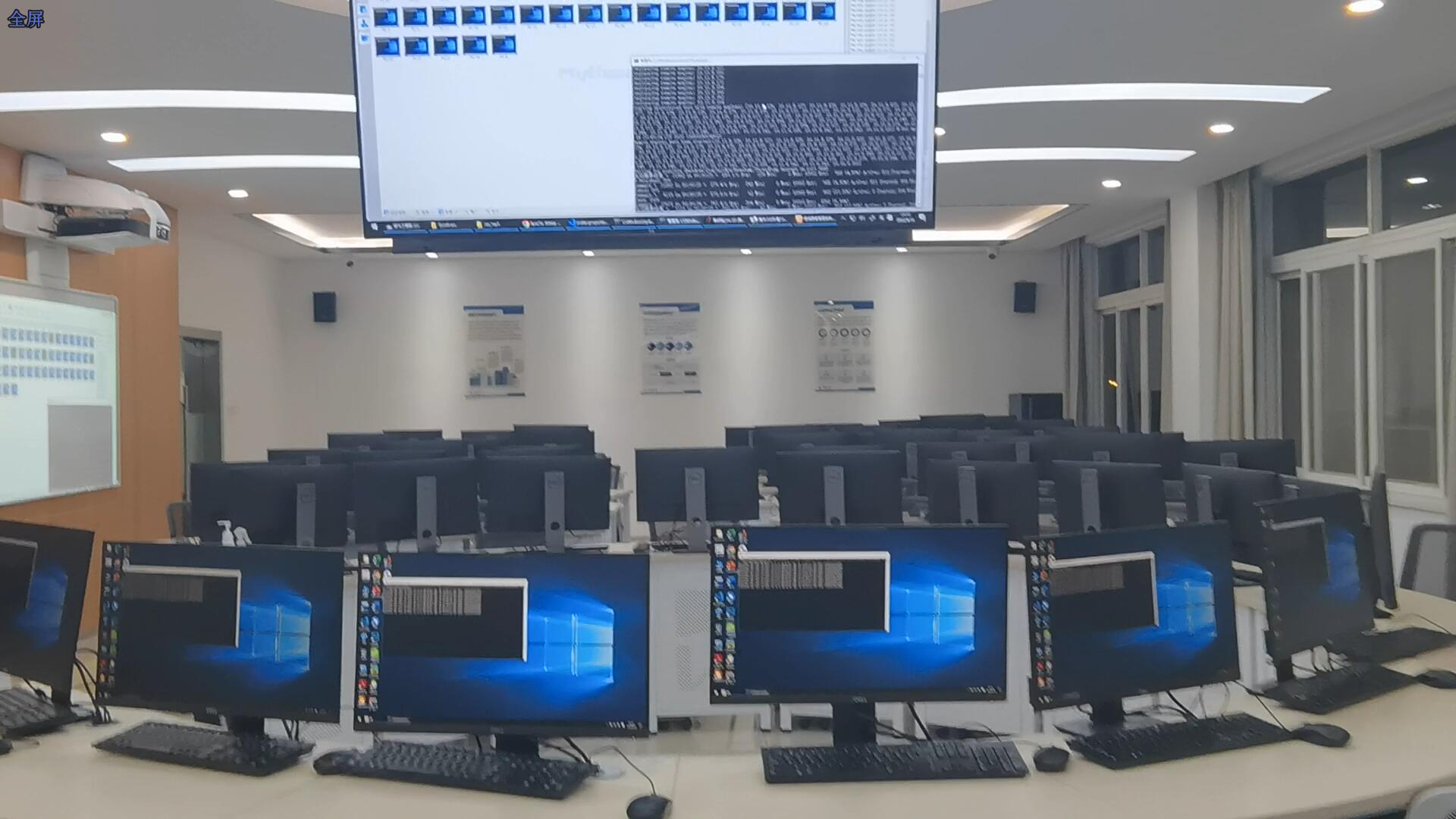


图 10 分布式测试场景

#### 7.1.3测试数据和结果分析

通过Jemter的GUI界面，团队提前配置了如下信息：

|  |  |  |
| --- | --- | --- |
| 属性名 | 属性值 | 备注 |
| 线程数 | 1000 | 无 |
| 线程启动间隔 | 1s | 每两次线程启动之间的间隔 |
| 循环次数 | 100次 | 无 |
| 每位用户购买数 | 1份 | 无 |
| 商品总份数 | 10000份 | 无 |
| 最大重试次数 | 11次 | 轮询订单接口最大次数 |

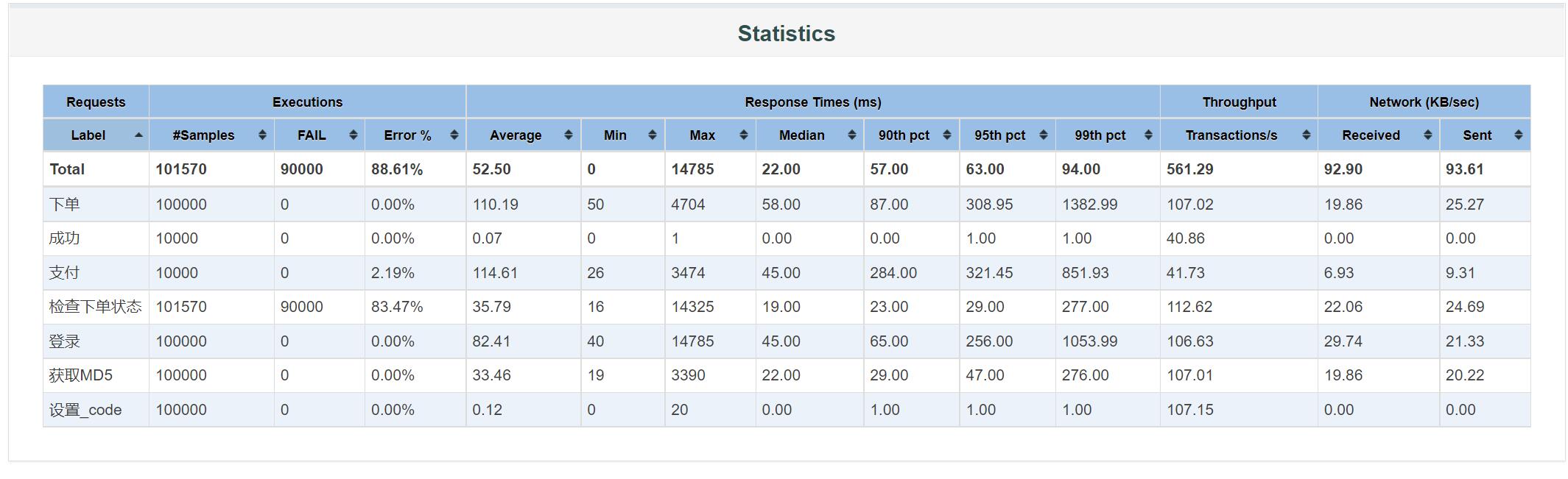
在高并发环境下，团队采用终端命令进行压力测试。测试结果如下。

图 11 测试结果状况图

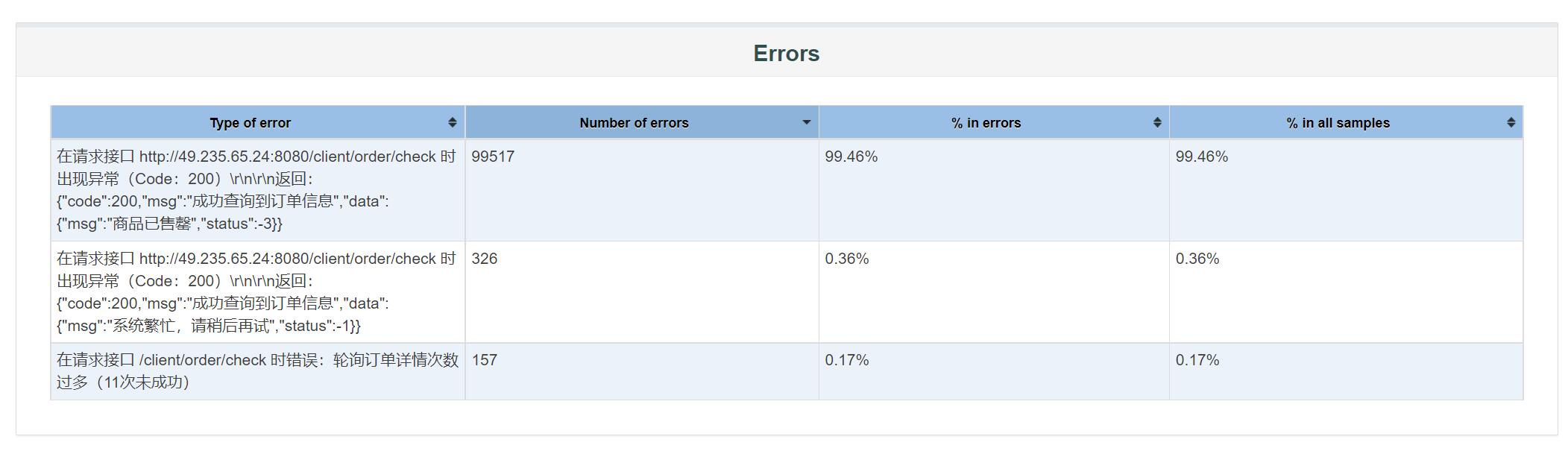


图12 测试错误数据图

由图11与图12所示，本团队开发的秒杀系统能够在高强度的分布式压测环境下平稳运行，登录、下单等并无出现异常。在请求检查下单状态这一接口时虽然有错误提示，但是99.48%的情况时因为商品已售罄，属于正常情况，另外的0.17%为轮询订单详情次数过多，发生概率比较低，并不影响用户的使用。

另外在Jmeter的报告中，还有整个测试持续的时间与商品售罄的时间的报告,如下图所示：

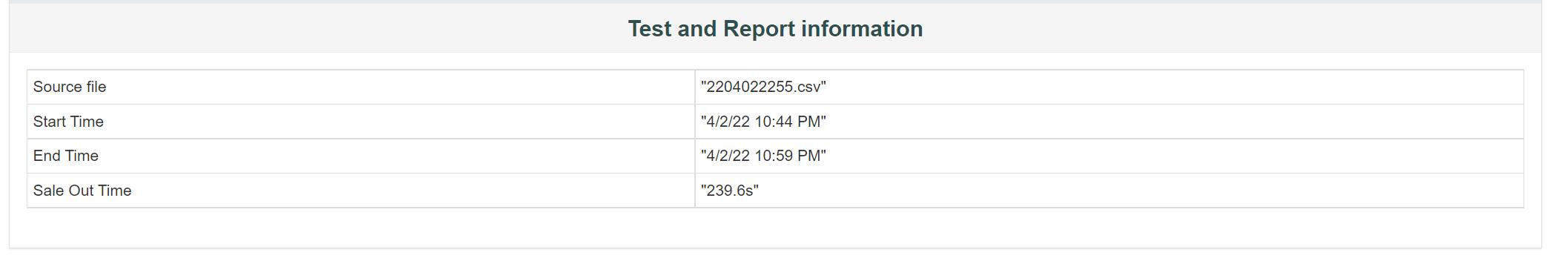


图 13 测试与报告信息图

从图8-8这个报告可以看出，整个测试持续的时间延续了15分钟左右，而商品售罄时间为秒杀活动开始后的239.6s。