**Frontiers in Computing and Intelligent Systems**
ISSN: 2832-6024 | Vol. 2, No. 2, 2022

# Design and optimization of an order killing system

**Lang Cheng, Changzhong Chen, Xin Liu, Qi Luo**

College of Automation and Information Engineering, Sichuan University of Science & Engineering, Zigong, Sichuan 643000, China.

**Abstract:** With the rapid development of the Internet industry, merchants ， second killing activities are increasingly frequent. Second kill goods are usually sold at prices far below the market, often sold out in a few seconds of activity. Therefore, it has the characteristics of large instantaneous flow and high concurrency. In this regard, this paper designs an order killing system based on java. The system uses SpringBoot framework to build, and uses Mysql database, with redis cache. At the same time, the second killing business is optimized and analyzed, using page static, Rabbitmq asynchronous order, and Nginx, Lvs to load balance the system to further improve the throughput of the system. Finally, the Jmeter pressure measurement tool is used to measure the system before and after optimization. The results show that the optimization has a certain effect.

**Keywords:** Second kill; High concurrency; Middleware; Load balancing.

## 1. Introduction

In recent years, with the rapid development of the global Internet, consumers ' awareness of online shopping has gradually become popular, and the habit of online shopping has gradually formed. The number of users and transactions of the e-commerce system has also ushered in a rapid improvement [1]-[3]. Now the electronic merchant system has occupied an important place in people 's work and life. The second killing activity is fast to attract the attention of consumers at low prices, increasing consumer desire to buy. Consumers can snap up their favorite goods, and stores get a huge turnover in the second killing activity. With the arrival of opportunities, challenges also follow.

Second kill requests usually have the characteristics of short duration, high concurrent access, large number of users, and small inventory [4]-[8]. Therefore, the usual architecture system often cannot meet the needs of second kill requests. Therefore, the design of a highly concurrent, stable, scalable, and developable seckill system plays a key role in the development of e-commerce.

## 2. Technology introduction

### 2.1. SpringBoot framework

Spring Boot is a new open source lightweight framework that was first released by the Pivotal team in April 2014. It is based on Spring4.0 design, not only inherits the original excellent features of Spring framework, but also further simplifies the whole construction and development process of Spring application by simplifying the configuration.

### 2.2. Redis

Redis (Remote Dictionary Service) is an open source log type, Key-value database written in ANSI and C language, which supports network and can be persistent basedon memory, and provides APIs in many languages.

### 2.3. Rabbitmq

Rabbitmq is an open source message queue tool that can help store and forward messages. Because of the asynchronous nature of message queues, it is usually used for message delivery in distributed systems.

### 2.4. Nginx

Nginx is a powerful high-performance Web and reverse proxy service. It has the advantages of high performance, good stability, strong concurrency, low resource consumption and simple configuration. It can run on multiple operating systems and is widely used because of its lightweight, modular, friendly configuration format and powerful reverse proxy capabilities.

### 2.5. Jmeter

Apache JMeter is a Java-based performance testing tool developed by Apache organization. It is an open source desktop application software that can be used to simulate user load to complete performance testing. JMeter can test Web applications. It can also test Java requests, JMS, EJB, Web Service, JDBC, FTP, JSR223, Socket and other protocols. In addition, it can meet specific test requirements by extending JMeter functions.

## 3. System design

### 3.1. Design objective

On the one hand, the system needs to meet the needs of merchants and suppliers for background management, on the other hand, it also needs to meet the business needs of customers. Therefore, the system integrates the functions of login module, order module, order detail module, security module, pressure measurement and so on.

Through a comprehensive system requirements consideration, the following design goals:

(1) Using the SpringBoot framework to develop the system to meet the business needs;

(2) Support high concurrency, high availability, the page is simple and easy to understand, to bring a better user experience;

(3) Background to achieve configurable, scalable distributed structure, easy to operate, easy to maintain.

## 3.2. System process analysis

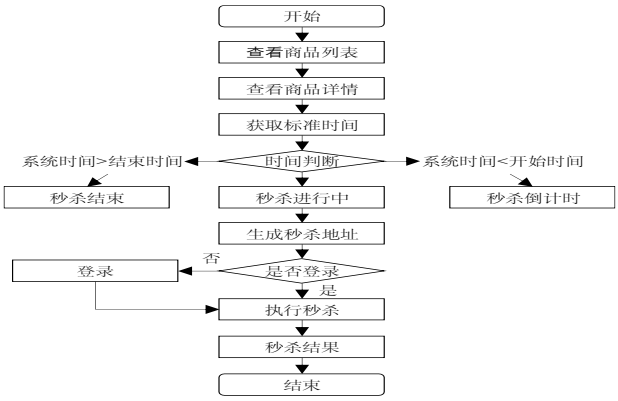


**Figure 1.** system flowchart

## 3.3. System functions design

(1) System login: user name password verification and other functions.

(2) Implementation and background management: commodity display list, order details, second kill countdown and other functions.

(3) Page-level high concurrency optimization: the realization of the product list page cache, goods, order details static, solve oversold and other functions.

(4) Service-level high concurrent second killing: Redis pre-reduction inventory, Rabbitmq asynchronous order, stress test comparison and other functions.

(5) Graphical verification code and malicious anti-brush: hide seconds to kill the interface address, graphical verification code, interface limit brush.

(6) Parameter optimization: Tomcat parameter optimization, Nginx reverse proxy, Nginx, Lvs load balancing.

The overall system function diagram is as follows.

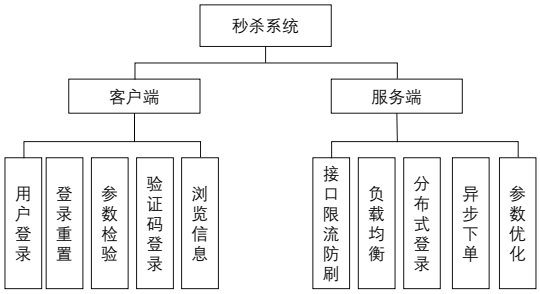


**Figure 2.** System overall function diagram

## 3.4. System database design

Through the analysis of the system data information and consumer demand for system function, and the future demand for high availability of the system, we can get the following data items:

(1) product information: product ID, product name, product unit price, product pictures, etc.;

(2) seconds to kill product information: product ID, inventory, seconds to kill price, seconds to kill the end time after the start;

(3) second to kill user information: user ID, mobile phone number, password, login times, etc.;

(4) order information: user ID, product ID, quantity, unit price, order status, etc.;

(5) Order information: user ID, order ID, product ID;

(6) User information: ID, user name.

The overall E-R diagram of the system is as follows:

**Figure 3.** E-R diagram

## 3.5. System implementation

(1) Login verification login, which is also the user entrance of the entire system, see below.


**Figure 4.** Login interface diagram

(2) Users can see product details by clicking on the Details button, second kill start time, second kill countdown, second kill success.

**Figure 5.** Login interface diagram

**Figure 6.** Shot success diagram

(3) Skip to order page after successful

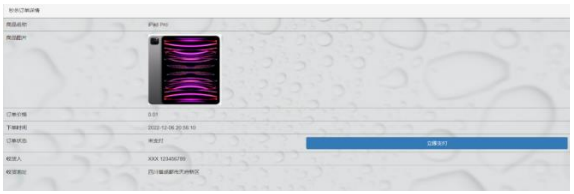

**Figure 7.** Order Chart

(4) Parameter optimization and load balancing configuration

```
#! /bin/bash
echo 1 > /proc/sys/net/ipv4/ip_forward
ipv=/sbin/ipvsadm
vip=192.168.88.150
rs1=192.168.88.130
rs2=192.168.88.131
case $1 in
start)
   echo "Start LVS"
   ifconfig eth160:0 $vip broadcast $vip netmask 255.255.255.255 up
   route add -host $vip dev eth160:0
   $ipv -A -t $vip:80 -s lc
   $ipv -a -t $vip:80 -r $rs1:80 -g -w 1
   $ipv -a -t $vip:80 -r $rs2:80 -g -w 1
;;
stop)
   echo "Stop LVS"
```

**Figure 8.** load balancing

(5) The optimized pressure test comparison the optimized system QPS meets our requirements as shown in the following figure.

| Requests | Executions | | | Response Times (ms) | | | | | | | Throughput | Network (KB/sec) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Label | #Samples | FAIL | Error % | Average | Min | Max | Median | 90th pct | 95th pct | 99th pct | Transactions/s | Received | Sent |
| Total | 1000 | 0 | 0.00% | 128.20 | 3 | 693 | 128.00 | 152.90 | 184.00 | 394.89 | 732.60 | 2534.05 | 114.47 |
| HTTP Request | 1000 | 0 | 0.00% | 128.20 | 3 | 693 | 128.00 | 152.90 | 184.00 | 394.89 | 732.60 | 2534.05 | 114.47 |

**Figure 9.** QPS diagram

# 4. Conclusion

In this paper, the problem of slow response and low QPS of the traditional second killing system is re-designed and optimized. By using the buffer and load balancing deployment of middleware, the problem of slow response and low QPS is optimized to a certain extent. It provides a reference technical solution for the industry and has broad dawn prospects.

# References


[1] Huang Zhilong, Xu Lisha, Qu Shaocheng. Design and optimization of high concurrency Web e-commerce system [J]. Computer and digital engineering, 2019,47 ( 07 ) : 1719-1724 + 1775. (In Chinese)

[2] Profiling and analyzing the I/O performance of NOSQL DBS [J] . Jiri Schindler. ACM SIGMETRICS Performance Evaluation Review . 2013 (1).

[3] NoSQL databases: a step to database scalability in web environment [J]. Jaroslav Pokorny. International Journal of Web Information Systems . 2013 (1).

[4] Research on high availability architecture of SQL and No SQL . Wang Z, wei Z, Liu H. AIP Conference Proceedings . 2017.

[5] Geng Xiaoli, Zhang Mang, Yin Yonghong. Research on distributed e-commerce platform architecture with high concurrency and high availability [J]. Computer technology and development, 2021,31 ( 02 ) : 111-115 + 121. (In Chinese)

[6] FFLtool:a web server for transcription factor and miRNA feed forward loop analysis in human.XIE Guiyan,XIA Mengxuan,MIAO Yaru,et al. Bioinformatics.2019

[7] Formulas and web application for designing a biospecimen pooling study to compare group means. VAN DOMELENDR,LYLESR H. Epidemiology.2020.

[8] Chen Rui. High concurrency Java Web development model based on Springboot [J]. Computer programming techniques and maintenance, 2019 ( 04 ) : 27-30.