HiLL PUBLISHING

Advances in Computer and Communications, 2022, 3(2), 77-80
https://www.hillpublisher.com/journals/acc/
ISSN Online: 2767-2875

Design and Implementation of a High Performance Red Packet Rushing System

Lei Liu

Guangdong Open University, Guangzhou, Guangdong 510000, China.

How to cite this paper: Lei Liu. (2022) Design and Implementation of a High Performance Red Packet Rushing System. *Advances in Computer and Communication*, 3(2), 77-80.
DOI: 10.26855/acc.2022.12.005

Received: October 28, 2022
Accepted: November 25, 2022
Published: December 30, 2022

***Corresponding author:** Lei Liu, Guangdong Open University, Guangzhou, Guangdong 510000, China.

Abstract

In view of the "short time, high concurrency and high performance" requirements of the red packet grabbing system, this paper designs a solution, which uses the industry popular SSM framework to achieve high scalability, Bootstrap framework to be compatible with multiple terminals, Redis cache to achieve high performance, and queues to deal with high concurrency. It focuses on the analysis and design of red packet sending and red packet grabbing processes and red packet allocation algorithms; Finally, the test and application practice show that the design of high-performance red packet application based on memory is effective. This scheme has a certain reference value for solving the requirements of instantaneous high concurrency.

Keywords

Red envelope grabbing system, High performance, Redis, Atomic operation, Red packet algorithm, Concurrent current limiting, Cluster deployment

1. Business analysis

The red packet grabbing system involves red packet sending users and red packet grabbing users. A typical red packet grabbing business is: a red packet sending user fills in the number of red packets, the total amount and other information to send the red packet, the red packet data is stored in the database, and the system pushes the message to the red packet grabbing user; After receiving the red packet message, multiple red packet users click Start Red Packet Robbing and send the request to the server. The server generates a small amount from the total amount to the user who has grabbed the red packet according to the red packet allocation algorithm. The total amount decreases, and the number of red packets decreases by one. When the big red packet is robbed by the user, the number of red packets and the total amount decrease to zero, and the details of the red packet are saved in the database, thus completing the red packet grabbing process. It is generally stipulated that the red packet is valid for 24 hours, and the same user can only rob the same red packet once. It can be seen from the analysis that a red packet sent out is often snatched up instantaneously. If the red packet exceeds 24 hours, it is invalid. The system is characterized by "short time"; The number of people captured in a red packet snatching activity is equal to the number of red packets. This number can be determined when red packets are sent, but it is not too large. However, the number of people involved in red packets snatching is usually greater than the number of red packets. If 100 million people send red packets at the same time, and the ratio of sending and robbing is 1:9, 900 million people will almost participate in red packet snatching at the same time. The system will carry a large number of "high concurrency" requests; A user sends a red packet and multiple users snatch it. In essence, it is an instantaneous competition for limited resources. It is a way to generate a large amount of money into a number of small amounts. Data access and amount calculation take time. Whether a

user sends a request to snatch a red packet successfully, or has snatched it, or has snatched it, the system must respond as quickly as possible. The system has the requirement of "high performance". To sum up, designing a robust red packet grabbing system should have the capability of grabbing and sending red packets functionally and meet the requirements of high performance and high concurrency in performance [2-3].

2. Red packet allocation algorith

A large number of users rob a limited amount of red packets. In essence, it is a process of decomposing a large number m into n decimals according to a certain algorithm. The range of decimals is [min, max]. Considering the high concurrency of red packet grabbing, the simpler the red packet allocation algorithm is, the more efficient it is. The faster it runs, the shorter the "resource competition queuing time" is, and the higher the system performance and concurrency processing capability are. Different red packet algorithms are designed from different angles. This scheme designs an algorithm to quickly generate random red packets.

Algorithm principle:

1) As the minimum amount of RMB is 1 point, the minimum amount of red envelope generated is 0.01 yuan;

2) If the number of red packets is one, the total amount is directly used to generate the red packet amount;

3) If the number of red packets is multiple, the random number between 0.01 and 2 times the average value of red packets is taken. The average value of red packets=the total remaining amount/the number of remaining red packets, that is, the minimum amount of random red packets generated in each round is 0.01, and the maximum amount is 2 times the average value of red packets. Using 2 times the average value of red packets as the maximum value can ensure that the amount of red packets generated in each round will not vary too much;

4) Each time a red packet is generated, the total amount minus the generated amount, and the number of red packets is reduced by one until the number of red packets is one, that is, the last red packet. The amount uses the remaining amount and is no longer generated randomly.

Assuming that the total amount of the remaining red packets is m, the number of remaining red packets is n, and the amount of red packets generated in each round is x, the algorithm formula is as follows:

$$x = \begin{cases} m & n = 1 \\ Rand\left(0.01, 2 * \frac{m}{n}\right) & n > 1 \end{cases}$$

$$m = m - x$$

$$n = n - 1$$

3. Red packet structure design

The action of the red packet sending user (the contractee for short) is to seal a red packet and throw it out. The key data of the red packet encapsulated includes the contractee ID, the red packet ID, the number of red packets, the total amount, etc. In order to cope with the concurrent red packet sending request exceeding the system's concurrent processing capacity, queue technology can be used to mitigate the impact on the server at the peak flow stage. The implementation structure of red packet sending is designed as follows: 1) The contractee fills in the red packet data in the front end and submits it to the server, To ensure the global uniqueness of a single red packet data in the cache module, a globally unique ID can be generated in advance through Random as the red packet primary key.

4. Structural design of red packet grabbing

A large number of users vie for a red packet. In essence, it is the preemption of limited resources under high concurrency. There is a problem of high concurrency competition. There is a limit on the number of red packets when they are sent, that is, the number of users who can effectively seize red packets can be estimated when they are sent. It is meaningless to compete for red packets after the red packets have been allocated for user access requests that exceed the number of red packets. Therefore, when the concurrency of red packets is relatively large, you can set a layer of request filtering queue. According to the counter statistics, the requests within the number of red packets are directly put in. Once the red packets have been robbed, the number of red packets is reduced to zero, and the subsequent requests are not put in any more. The direct return is "red packets have been robbed". The process of robbing red packets is completed in memory, and the red packet data and the robbed user list are temporarily cached, so as to ensure the fastest response performance and wait for the red packets to be robbed, the user ID and amount of red

packet will be warehoused in batch. The implementation structure of red packet grabbing is designed as follows: 1) the packet grabber sends a request for red packet grabbing; 2) when the concurrency is large, the request is first put into the queue, and the number of red packets exceeds the number of red packets directly returned through the counter statistics; 3) a valid red packet grabbing request is put in, and the red packet grabbing process is completed in memory. The remaining number and amount of red packets are obtained from the cache according to the red packet ID, and a random amount is generated to the packet grabber through the red packet allocation algorithm, this calculation is completed in real time in memory. Update the remaining amount of red packets, reduce the remaining amount of red packets by 1, and put the updated red packet data back into the cache for the next red packet grab. At the same time, put the user ID and amount of the red packet grab into the cache, which can maintain a data array of the user grab, and record the ID and amount of all the users grab. 4) When the red packet is robbed, Put the user data array that catches the red packet into the receipt queue. 5) The data in the receipt queue is stored in the database asynchronously in batches.

5. System test

In order to test the performance of the red packet grabbing system designed in this paper, the author carried out two groups of simulation experiments on a PC, the hardware configuration is: CPU Core i5 3.3GHz, memory 8GB, and the software configuration is: Tomcat 7.0, JDK 1.7, Redis 3.0, MySQL 5.5.

In order to test the fairness of the red packet allocation algorithm designed in this paper, the author simulated 10 people grabbing 100 yuan red packets, generating 10 small red packet data in each round, executing 1000000 rounds, and generating 1000000 red packet data per person. Calculate the average value and standard deviation of each person's red packet data, and the data obtained is shown in Table 1.

Table 1. Simulation of 10 people grabbing 100 yuan (1000000 times)

Bag snatcher	average value	Dimension difference
The first person	10.00	5.78
The second person	9.99	5.82
The third person	10.00	5.88
The fourth person	9.99	5.95
The 5th person	10.00	6.07
The 6th person	10.00	6.22
The 7th person	10.00	6.44
The 8th person	10.01	6.84
The 9th person	10.01	7.69
The 10th person	10.00	7.68

It can be seen from the experimental data that: (1) the average values of 10 people are roughly equal and fluctuate around 100/10=10. From a statistical point of view, it shows that the expected value of red packets grabbed by each person is roughly equal, that is, the face value of red packets grabbed by everyone is roughly evenly distributed in probability and close to the theoretical average (total amount/total number); (2) There is a difference in the standard deviation of 10 people, which gradually increases from the first person to the tenth person. This shows that the red envelope data of Mr. Cheng has a small fluctuation, while the later generated red envelope data has a large fluctuation. That is, the later robbers have a greater chance to get "the best luck" or "the worst luck", which increases the fun of robbing red envelopes.

6. Summary

The red packet grabbing solution designed and implemented in this paper uses the SSM framework to achieve high scalability, implements high-performance red packet allocation algorithm based on Redis cache, and uses clustered deployment to enhance the system's concurrent processing capability. Experiments show that this solution is typical and has a general reference value for solving the requirements of "instant high concurrent resource grabbing" such as

red packet grabbing and second killing. This paper does not discuss the security of the red packet grabbing system more, further research is needed. Of course, there are more than one online implementation scheme of the red packet grabbing system, and different red packet allocation algorithms can be designed from different perspectives. There is no optimal and one-time scheme, only the scheme that best meets the business needs, and readers can choose flexibly according to the scenario.

References

[1] Li Jichao. The application of "robbing red packets" in online marketing [J]. Youth reporter. 2015/23.

[2] Xu Jingzhou. Design and Implementation of Redis based High Concurrency Red Packet Robbing Application [D]. Hunan University. 2016.

[3] Li Junfeng, He Mingxin. Design and Implementation of High Concurrency Web Ticketing Seckill System [J]. Computer Engineering and Design. 2013 (03).

[4] Wang Yanqing, Chen Hong. Research and Design of Intelligent Web System Based on SSM Framework [J]. Computer Engineering and Design. December 2012.

[5] Zhang Zijie, Zhuang Yufei. Design and Implementation of Job Recruitment System Based on Bootstrap and SSH [J]. Software Guide. 2016 (10).

[6] Yang Yongbin, Tang Lianggui. Research on Application of Queue Scheduling Algorithm in Network [J]. Computer Science. 2005/07.

[7] Liu Junlong, Liu Guangming, Zhang Dai, Yu Jie. Research on Real time Storage and Indexing Strategy of Massive Internet Small Files Based on Redis [J]. Computer Research and Development. 2015/S2.

[8] Ma Yuxing. Analysis of Redis Database Characteristics [J]. Internet of Things Technology, 2015,03:105-106.

[9] Yang Yan, Li Wei, Wang Chun. Application of memory database in cache [J]. Modern Telecom Technology, 2011, 12: 59-64.

[10] Li Shiyun. Performance optimization of crowdsourcing system based on memory database Redis [D]. Zhejiang University. 2016.

[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] Li Jun. Design and Optimization of High Concurrency Web System [D]. Beijing Jiaotong University. 2009.

[13] Bao Lihui, Huang Yanfei. Research on the architecture of high concurrency websites and solutions [J]. Computer Science. 2012 (S2).

[14] Mei Huawei, Zhang Mingquan, Li Tian. Research on High Concurrency and High Load Website System Architecture [J]. Computer and Network. 2009 (14).