An Improved Token Bucket Algorithm for Service Gateway Traffic Limiting

Lang Li[1], Xinming Tan[1], Chao Deng[1]
1. School of Computer Science &Technology, Wuhan University of Technology
Wuhan, Hubei

Keywords: Micro-service, Zuul, Token bucket algorithm, URI configuration file.

Abstract: In the Spring Cloud micro-service system, the gateway Zuul, as the entrance to all internal services, must carry out the traffic limiting operation to guarantee the stability of the service in the case of high concurrency. The traditional token bucket traffic shaping algorithm cannot guarantee the stability of core services. In order to solve this problem, this paper designed an overload protection strategy based on URI configuration file in combination with gateway Zuul, which can filter the request before acquiring the tokens. It improved the token bucket algorithm, realized the traffic limiting function and guaranteed the stability and immediacy of the core service process.

1 INTRODUCTION

With the development of computer technology, the traditional single application architecture has been unable to meet the rapidly growing customer demand. Componentization based on micro-service has gradually become a new choice for system design, and has been rapidly developed and applied (Zhang Jing, et al, 2016). In the micro-service architecture, service consumers and service providers interact through a service gateway between the two middle tiers. The API gateway is the entry point of external services, hiding the implementation of the internal architecture. Therefore, when all API calls are routed through the service gateway, the gateway must guarantee the stability of the service in the case of high concurrency. At the upper limit of the service gateway, a certain traffic shaping strategy must be in place to ensure the stability of the service.

Currently, in the Spring Cloud micro-service architecture, the token bucket algorithm is used for the current limiting operation mainly through the gateway component Zuul. One of the problems is that regardless of the current request URI, as long as the load threshold set by the system is exceeded, the request will be directly discarded or re-forwarded. This flow control method is too simple and may result in the core API like payment being discarded, while other non-core API are frequently accessed. This affects the user's sense of experience and affects the interaction between core services in the overall micro-service architecture.

Based on the traditional token bucket algorithm and the gateway Zuul, this paper proposes an overload protection strategy based on URI configuration file. The basic idea is that when each request arrives at the gateway, it intercepts and filters the current URI before requesting the token, and determines whether it needs to a current-limit operation that processes the request. This strategy can guarantee the stability of the micro-service system under high concurrency. And the feasibility and performance of the improved method are verified by experiments.

2 RELATED WORK

2.1 Research on Gateway Zuul

The core component of the micro-service system's current-limit operation is the gateway Zuul (Li Conglei, 2017). Zuul is Netflix's open source micro-service gateway, a server load balancing device based on JVM routing (Zhou Yongsheng, et al, 2018). The core of Zuul is a series of filters that allow users to implement authentication, security, scheduling, and traffic limiting functions through custom filters (Zhang Jie, et al, 2018).

In the literature (Zhu Rongxin, 2017) the Zuul component is used to implement the API gateway system of the game mall server based on the

Li, L., Tan, X. and Deng, C.
An Improved Token Bucket Algorithm for Service Gateway Traffic Limiting.
DOI: 10.5220/0008098802220226
In *Proceedings of the International Conference on Advances in Computer Technology, Information Science and Communications (CTISC 2019)*, pages 222-226
ISBN: 978-989-758-357-5

micro-service architecture. In the literature (Zhao Qing, 2018), the Zuul gateway is applied to the design of the news data service platform, and they customize each stage filter to implement the interception and filtering of requests through the service gateway. Thus providing functions such as service routing, service filtering, and rights management. But they don't consider the stability of the gateway component Zuul in the case of high concurrent access.

2.2 Research on Related Traffic Limiting Strategies

In order to ensure the stability of the service of the gateway component in the case of high concurrent access, a certain traffic limiting policy is needed. In the literature (Liu Yitian, et al, 2018), it uses the Zuul component to implement unified micro-service access routing and forwarding, considering that the single-node Zuul service will become a bottleneck of the whole system operation, it is proposed that multiple Zuul nodes can be deployed and registered in the service registry. That is, the Zuul clustering method is implemented to achieve high availability of the gateway service, but this will result in more scheduling and management capabilities of the deployment architecture. In the design of service security access control framework, literature (Lu Liangwei, et al, 2018) is proposed that when the request is high concurrency, the API gateway uses the token bucket current limiting algorithm to limit the traffic, ensure the availability and performance of the system.

However, the traditional token bucket current limiting algorithm is oriented to all requests to the gateway, and cannot selectively carry out current limiting operation according to different levels of API requests. Therefore, combined with the gateway component Zuul, this paper designs an overload protection strategy based on URI configuration file, which improves the token bucket algorithm. Under high concurrency condition, it can guarantee the stability of the core API service and limit the operation of URI which is not included in the configuration file.

3 IMPROVEMENT AND IMPLEMENTATION OF TOKEN BUCKET ALGORITHM BASED ON SERVICE GATEWAY

3.1 Traditional Token Bucket Algorithm

At present, the classic traffic shaping algorithm includes the leaky bucket algorithm and the token bucket algorithm (Zhong Sihui, 2016). In Google's open source toolkit Guava, a current limiting tool class RateLimiter (Ali A, et al, 2013) is implemented based on the token bucket algorithm. Since the module is already integrated in the Spring Cloud, the Zuul gateway can use the token bucket algorithm for traffic limiting operations. The token bucket algorithm is one of the most commonly used algorithms for network traffic shaping and rate limiting (Liu Yuanfeng, 2008). The basic idea of the traditional token bucket algorithm is as follows:

1) Put the tokens into the bucket at an average rate of r per second.
2) A maximum of b tokens can be stored in the bucket. If the bucket is full, the newly placed token will be discarded.
3) When an n-byte packet arrives, it consumes n tokens and forwards the packet to the network.
4) If the available tokens in the bucket are less than the n tokens required by the packet, the tokens in the bucket are not deleted, and the packet is identified as being outside the traffic restriction.

The idea of the traditional token bucket algorithm is shown in Figure 1.

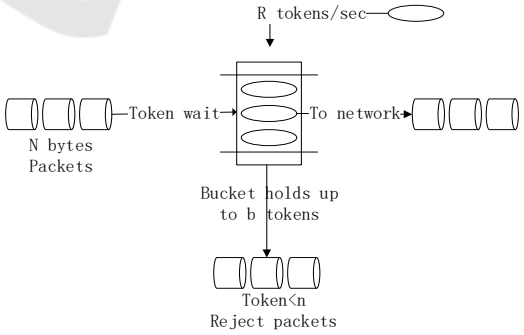

Figure 1: Traditional token bucket algorithm.

The traditional token bucket algorithm can also allow a certain degree of burst transmission (Li Wei, et al, 2011) in addition to limiting the data transmission rate. As long as there is a token in the bucket, it can burst data in the configured threshold,

so it is suitable for the sudden traffic access in the gateway (Liu Zhenyu, 2012).

3.2 Improved Token Bucket Algorithm

The traditional token bucket algorithm does not filter the request when it allocates tokens on the request packet. It just judges whether there are enough tokens in the bucket for allocation. If the number of tokens in the bucket is insufficient, all requests that satisfy the condition are discarded, which will include some core API such as order payment requests, which will affect the quality and stability of the service.

This paper improves the traditional token bucket algorithm based on gateway Zuul. First, gateway Zuul intercepts all requests, and inserts a URI configuration file interception strategy in the step of waiting for the tokens which intercepts the request before blocking the token. The configuration file contains all the core URIs. The improved token bucket algorithm is as follows in Figure 2.

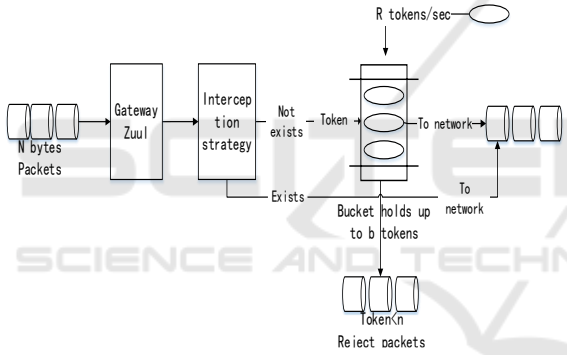

Figure 2: Improved token bucket algorithm.

As can be seen from the above figure, after the request packet arrives at the gateway Zuul, the interception policy determines whether the request URI is in a customized configuration file. If it exists, it does not need to obtain a token for traffic limiting, and directly allows access to the service. If it does not exist, it needs to request the tokens. When the threshold is exceeded, the current limiting operation is performed.

3.3 Gateway Zuul Implements An Improved Token Bucket Algorithm

In this paper, the interception strategy based on URI configuration file is designed in the token bucket algorithm. The custom URI configuration file is needed to read the core API. At the same time, in the gateway Zuul, the ZuulFilter abstract class is implemented by the custom filter class which contains four abstract functions defined by ZuulFilter. The four abstract functions in the ZuulFilter abstract class and the concrete implementation are as follows:

1) filterType: Returns a string representing the type of the filter. Since the current-limit operation processes the request before it is routed, the LimitFilter class is of type "pre".
2) filterOrder: Defines the execution order of the filter by the integer value. The LimitFilter class is defined as the first executed filter, so it returns 0.
3) shouldFilter: Returns a Boolean type to determine whether the filter is to be executed, so this function can be used to implement the filter switch. Because the LimitFilter class must be executed, so it returns true.
4) Run: The specific logic of the filter, the custom code logic is implemented within this method.

The specific flow chart is shown in Figure 3.

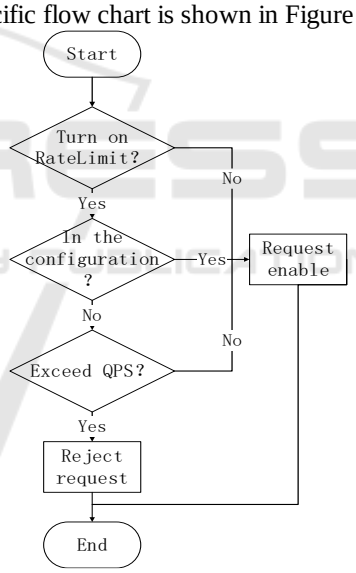

Figure 3: Zuul traffic limiting flow chart.

1) Create a RateLimiter object and set the initial rate value.
2) Obtain the current request URI, read the configuration file, and determine whether it is in the URI configuration list.
3) If it is in the configuration file, it is not subject to the rate value and allows direct access.
4) If not, you need to be bound by the rate value. And obtain the tokens from RateLimit. If the tokens are not obtained within the

10ms timeout period, the threshold has been exceeded, so the request is discarded.

5) If the tokens are obtained within the 10ms timeout period, the current rate value is still within the security range, so the request is allowed to access.

4. EXPERIMENTS AND RESULTS

In order to verify the feasibility of the improved design of the token bucket algorithm, this paper conducts experiments to verify whether the improvement measures are successful and effective by performing functional tests and performance tests in the micro-service system.

4.1. Functional Test

First, the whole project is divided into four modules: service registry module EurekaServer, service gateway module Eureka-Zuul, service consumer EurekaClient, the path of service external interface is /eurekaclient/test1, service consumer EurekaClient1, service external interface path is /eurekaclient1/test2. After the four modules are started, the results of service registration are shown in Figure 4.

Figure 4: Service Registry Center Diagram.

It can be seen from Figure 4 that the four modules are successfully started and registered. At the same time, through the analysis of the log output, it proves that the /eurekaclient/test1 interface path exists in the URI configuration file, and /eurekaclient1/test2 does not exist. This verifies the request URI was successfully intercepted and filtered by the custom gateway filter LimitFilter.

4.2. Performance Test

This article uses the Apache ab command to simulate multi-threaded concurrent requests. The token bucket algorithm's current-limit threshold is set to 10. The test scenario simulates the same number of requests and the amount of concurrently. The two interfaces are tested by the traditional token bucket algorithm and the improved token bucket algorithm respectively. In the case of the different algorithm, a stress test is performed to obtain the response time of each interface path under different conditions, and the result is recorded in following Table 1.

Table 1: Response time of each interface.

T-R	C-C	eurekaclient/ test1		eurekaclient1/ test2	
		T-A	I-A	T-A	I-A
100	3	9.024s	0.511s	9.656s	8.966s
100	10	15.458s	2.139s	14.658s	15.602s
200	10	20.416s	0.392s	22.215s	22.012s

In the table, T-R represents the total number of requests, C-C represents the current concurrency, T-A represents the traditional token bucket algorithm, and I-A represents the improved token bucket algorithm. As can be seen from the above table, in the case of the same concurrent request and current limiting parameters, the /eurekaclient/test1 interface acts as the core API, and the response time in the improved token bucket algorithm is much smaller than in the traditional token bucket algorithm. And the interface /eurekaclient1/test2 is not the core API, so the traffic limit operation has to be performed under both algorithms, so the response time is very close. It can be seen from the above results that with the improved token bucket algorithm, the traffic-limit interception strategy based on the URI configuration file can effectively process the core interface, ensure the stability of the core service, and ensure the current limit operation of the non-core API at the time of high concurrent request.

5 CONCLUSION

The traditional token bucket current limiting algorithm does not implement filtering of requests when assigning tokens. Through the core filter of the custom gateway Zuul, this paper designs and implements a traffic limiting interception strategy based on URI configuration file. At the same time, the experimental results show that compared with the traditional token bucket algorithm, the improved token bucket algorithm can guarantee the stability and immediacy of the core interface and service in the high concurrency situation of the micro-service system, as well as the traffic limiting operation for the non-core API. It is important to avoid abnormal traffic and malicious attacks.

REFERENCES

Zhang Jing, Huang Xiaofeng. An application framework based on micro-services [j]. Computer System Application, 2016, 25(09): 265-270.

Li Conglei. Design and implementation of an open platform for public service information services for microservices [d]. Southwest University of Science and Technology, 2017.

Zhou Yongsheng, Hou Fengyu, Sun Wen, Yang Lei, Zhang Xiaobei, Design and Implementation of Invoicing Management System Based on Spring Cloud micro-services Architecture[J]. Industrial Control Computer,2018,31(11):129-130+133.

Zhang Jie, Si Weichao, Wang Lina, Shi Chunling. Design and application of a general assessment system for micro-services [j]. Computer and Digital Engineering, 2018, 46(12): 2463-2467+2533.

Zhu Rongxin. Design and implementation of game mall server based on micro-service architecture [d]. Nanjing University, 2017.

Zhao Qing. Application of micro-service architecture in news data platform [a]. China Federation of Journalists and Technicians, Xinjiang Radio and Television Bureau. Proceedings of the 2018 Academic Annual Meeting of the China Association of Journalists and Technicians (Academic Papers) [c]. China Federation of Journalists and Technicians, Xinjiang Radio and Television Bureau: China Federation of Journalists and Technicians, 2018: 6.

Liu Yitian, Lin Tingjun, Liu Shijin. Flexible Microservice Security Access Control Framework [j]. Computer System Application, 2018, 27(10): 70-74.

Lu Liangwei, Huang Xiaofang.Design of Open Platform Framework for Secure and Scalable SaaS Services[J]. Computer measurement and Control,2018, 26(12):244-248.

Zhong Sihui. Design and implementation of flow control service based on token bucket algorithm [d]. Dalian University of Technology, 2016.

Ali A, Hutchison D, Angelov P, et al. Towards an autonomous resilience strategy for the implementation of a self-evolving rate limiter[C]//Computational Intelligence(UKCI), 2013 13th UK Workshop on. Guildford: IEEE, 2013:275-279.

Liu Yuanfeng. Research and implementation of network traffic shaping strategy based on leaky bucket theory and token bucket algorithm [d]. Northeast Normal University, 2008.

Li Wei. Research on key technologies of multimedia cloud computing platform [D]. University of Science and Technology of China, 2011s.

Liu Zhenyu. Research and implementation of network flow control technology based on token bucket algorithm [d]. Inner Mongolia University, 2012.