What is Rate Limiter? How does it work

JAVINPAUL

JAN 14, 2024 • PAID

20 2 Share

In the dynamic landscape of computing, where applications and services interact with many users and systems, maintaining stability and preventing abuse are paramount concerns.

Enter the realm of **rate limiting—a strategic mechanism designed to regulate the flow of requests or operations**, ensuring that the delicate balance between accessibility and resource protection is preserved.

In this issue of the Newsletter, we will delve into the fundamental concept of rate limiting, exploring its significance in diverse computing environments. From safeguarding APIs against malicious attacks to preventing server overloads, rate limiters are crucial in enhancing system reliability and security.

Along the way, you will learn the inner workings of rate limiting, unraveling the mechanisms that empower it to handle the ebb and flow of digital interactions gracefully.

Whether you are a developer, system architect, or simply curious about the intricacies of computational control, this exploration promises insights into the art and science of managing access in the digital realm.

What is a Rate Limiter?

A rate limiter is a mechanism used in computing to control the rate at which certain operations or requests are allowed. It is commonly employed in various applications and systems to prevent abuse, protect resources, and maintain stability.

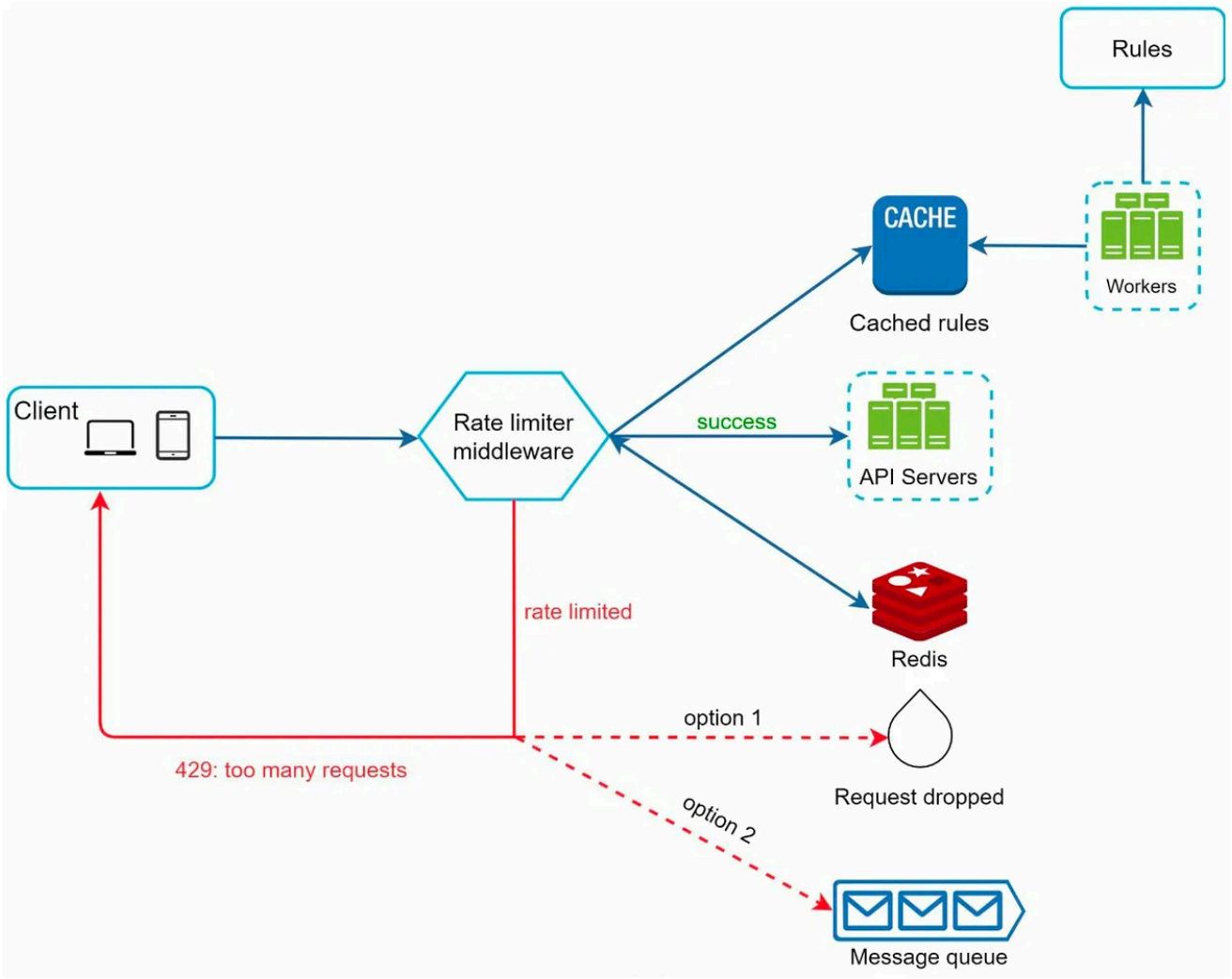
The basic idea behind a rate limiter is to restrict the number of requests or operations that a user or system can perform within a specified period. This is done to prevent overloading a service, avoid unnecessary resource consumption, and protect against potential malicious activities or unintentional misuse.

Here's a simple explanation of how a rate limiter works:

1. **Define Limits:** Specify the maximum number of requests or operations allowed within a certain time window. For example, you might allow only 100 requests per minute.

2. **Track Usage:** The rate limiter keeps track of the number of requests made by a user or system over time.
3. **Check Limits:** Before processing a new request, the rate limiter checks whether the user or system has exceeded the defined limits within the current time window.
4. **Decision Making:** If the user or system is within the allowed limits, the request is processed. Otherwise, the request might be delayed, rejected, or some other action taken, depending on the specific implementation.

And here is a nice diagram that shows the Rate Limiter in Action



Rate Limiter Algorithms

There are different algorithms and strategies for implementing rate limiting, depending on the requirements and characteristics of the system. Some common approaches include:

- **Token Bucket:** Requests are granted as long as there are tokens available in a bucket. Tokens are added to the bucket at a fixed rate, and each request consumes one or more tokens.
- **Leaky Bucket:** Similar to the token bucket, but instead of tokens, requests leak out of the bucket at a fixed rate. If the bucket overflows, excess requests may be delayed or discarded.

- **Sliding Window:** Counts the number of requests made within a sliding time window. If the count exceeds the allowed limit, further requests are delayed or rejected.

Rate limiting is widely used in web APIs, network protocols, web servers, and various distributed systems to ensure fair usage, prevent abuse, and maintain system stability. If you want to learn more then the Rate Limiter chapter on *ByteByteGo by Alex Xu* is a great resource, I learned most of my knowledge from there itself.

In the vast seas of digital interactions, rate limiting emerges as the captain steering the ship. Balancing accessibility and protection, it's the unsung hero preventing overloads and ensuring smooth sailing.

From token buckets to sliding windows, these mechanisms equip us to navigate the digital waves, fostering a reliable and secure online experience. In a world defined by connectivity, rate limiting proves indispensable—a guardian of equilibrium in our dynamic digital realm.

It's also one of the essential concepts to learn for a System design interview and if you are preparing for a System design interview, here are a few more resources you can check out:

- *System Design Interview Vol 2*
- *Grokking the System Design Interview*
- *Designing Data-Intensive Applications*
- *ByteByteGo* ***by Alex Xu***
- *https://www.infoq.cn/article/the-road-of-the-growth-weixin-background*
- *https://systemdesign.one/slack-architecture/*

20 Likes · 2 Restacks

← Previous

Next →

Comments

Write a comment...

© 2024 javinpaul · Privacy · Terms · Collection notice
Substack is the home for great culture