主讲老师：Fox

> 有道笔记地址：https://note.youdao.com/s/dkZGfI5l

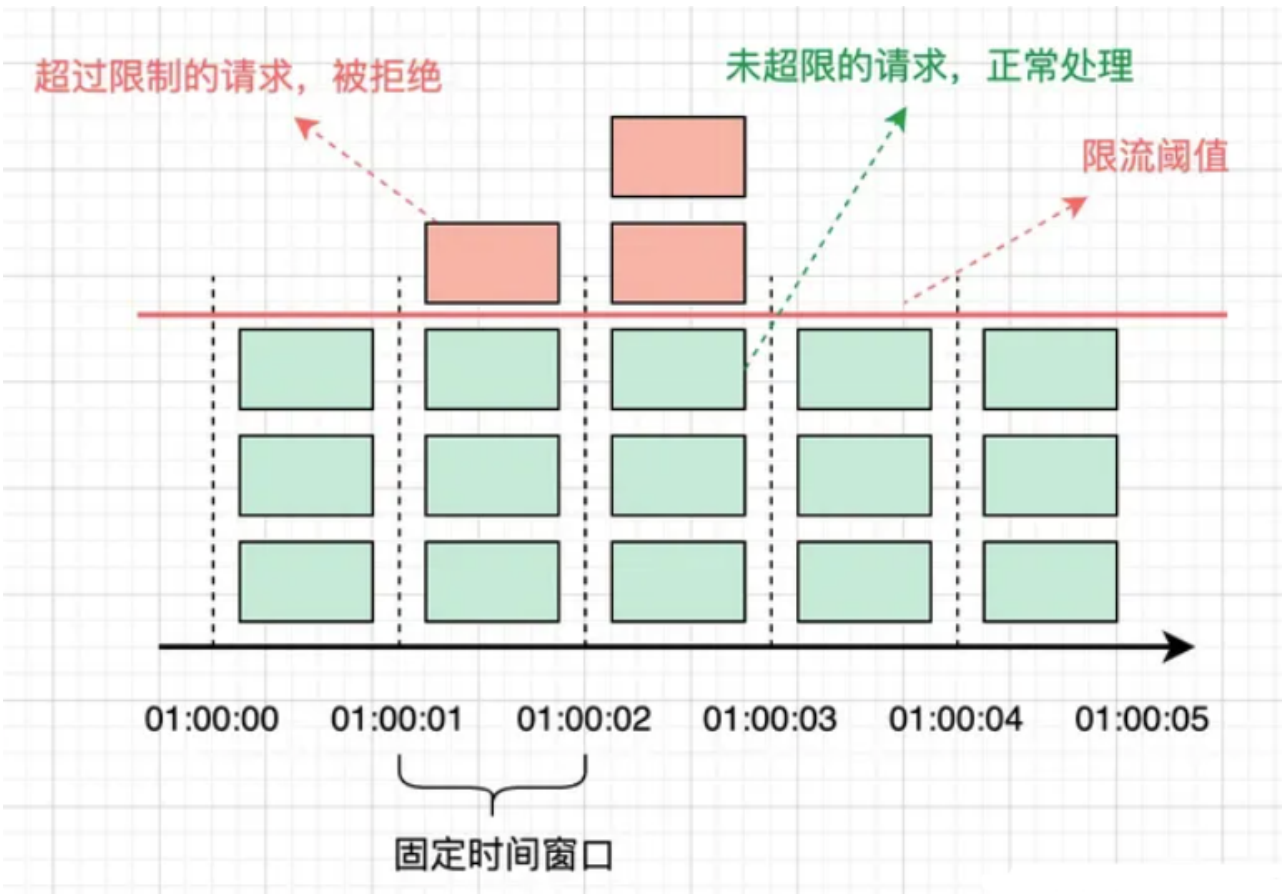
1.常见的限流算法

在分布式系统中，高并发场景下，为了防止系统因突然的流量激增而导致的崩溃，同时保证服务的高可用性和稳定性，限流是最常用的手段。

限流算法也是面试中必考题，今天我们学习一下常见的四种限流算法，分别是：**固定窗口算法、滑动窗口算法、漏桶算法、令牌桶算法。**

1.1 固定窗口算法

固定窗口限流算法（Fixed Window Rate Limiting Algorithm）也叫计数器限流算法，是一种最简单的限流算法。在一个固定长度的时间窗口内限制请求数量，每来一个请求，请求次数加一，如果请求数量超过最大限制，就拒绝该请求。



优点： 实现简单，容易理解。

缺点：

- 限流不够平滑。例如：限流是每秒3个，在第一毫秒发送了3个请求，达到限流，窗口剩余时间的请求都将会被拒绝，体验不好。
- 无法处理窗口边界问题。因为是在某个时间窗口内进行流量控制，所以可能会出现窗口边界效应，即在时间窗口

的边界处可能会有大量的请求被允许通过，从而导致突发流量。

例如：限流是每秒3个，在第一秒的最后一毫秒发送了3个请求，在第二秒的第一毫秒又发送了3个请求。在这两毫米内处理了6个请求，但是并没有触发限流。如果出现突发流量，可能会压垮服务器。

1.2 滑动窗口算法

滑动窗口算法是对固定窗口算法的一种改进。它将单位时间周期分为n个小周期，分别记录每个小周期内接口的访问次数，并且根据时间滑动删除过期的小周期。它可以解决固定窗口临界值的问题。当滑动窗口的格子周期划分的越多，那么滑动窗口的滚动就越平滑，限流的统计就会越精确。

假设单位时间还是1s，滑动窗口算法把它划分为5个小周期，也就是滑动窗口（单位时间）被划分为5个小格子。每格表示0.2s。每过0.2s，时间窗口就会往右滑动一格。然后呢，每个小周期，都有自己独立的计数器，如果请求是0.83s到达的，0.8~1.0s对应的计数器就会加1。

假设我们1s内的限流阀值是5个请求，0.8~1.0s内（比如0.9s的时候）来了5个请求，落在黄色格子里。时间过了1.0s这个点之后，又来5个请求，落在紫色格子里。如果是固定窗口算法，是不会被限流的，但是滑动窗口的话，每过一个小周期，它会右移一个小格。过了1.0s这个点后，会右移一小格，当前的单位时间段是0.2~1.2s，这个区域的请求已经超过限定的5了，已触发限流啦，实际上，紫色格子的请求都被拒绝啦。

优点：

- 精度高（通过调整时间窗口的大小来实现不同的限流效果）
- 可扩展性强（可以非常容易地与其他限流算法结合使用）

缺点：

突发流量无法处理（无法应对短时间内的大量请求，但是一旦到达限流后，请求都会直接暴力被拒绝），需要合理调整时间窗口大小。

1.3 漏桶算法

漏桶限流算法是一种常用的流量整形（Traffic Shaping）和流量控制（Traffic Policing）的算法，它可以有效地控制数据的传输速率以及防止网络拥塞。

实现原理是：

1. 一个固定容量的漏桶，按照固定速率出水（处理请求）；
2. 当流入水（请求数量）的速度过大会直接溢出（请求数量超过限制则直接拒绝）。
3. 桶里的水（请求）不够则无法出水（桶内没有请求则不处理）。

当请求流量正常或者较小的时候，请求能够得到正常的处理。当请求流量过大时，漏桶限流算法可以通过丢弃部分请求来防止系统过载。

这种算法的一个重要特性是，输出数据的速率始终是稳定的，无论输入的数据流量如何变化。这就确保了系统的负载不会超过预设的阈值。但是，由于漏桶的出口速度是固定的，所以无法处理突发流量。此外，如果入口流量过大，漏桶可能会溢出，导致数据丢失。

优点：

- 平滑流量。由于漏桶算法以固定的速率处理请求，可以有效地平滑和整形流量，避免流量的突发和波动（类似于消息队列的削峰填谷的作用）。
- 防止过载。当流入的请求超过桶的容量时，可以直接丢弃请求，防止系统过载。

缺点：

- 无法处理突发流量：由于漏桶的出口速度是固定的，无法处理突发流量。例如，即使在流量较小的时候，也无法以更快的速度处理请求。
- 可能会丢失数据：如果入口流量过大，超过了桶的容量，那么就需要丢弃部分请求。在一些不能接受丢失请求的场景中，这可能是一个问题。
- 不适合速率变化大的场景：如果速率变化大，或者需要动态调整速率，那么漏桶算法就无法满足需求。

1.4 令牌桶算法

令牌桶限流算法是一种常用的流量整形和速率限制算法。

实现原理：

1. 系统以固定的速率向桶中添加令牌；
2. 当有请求到来时，会尝试从桶中移除一个令牌，如果桶中有足够的令牌，则请求可以被处理或数据包可以被发送；
3. 如果桶中没有令牌，那么请求将被拒绝；
4. 桶中的令牌数不能超过桶的容量，如果新生成的令牌超过了桶的容量，那么新的令牌会被丢弃。

令牌桶算法的一个重要特性是，它能够应对突发流量。当桶中有足够的令牌时，可以一次性处理多个请求，这对于需要处理突发流量的应用场景非常有用。但是又不会无限制的增加处理速率导致压垮服务器，因为桶内令牌数量是有限制的。

优点：

- 可以处理突发流量：令牌桶算法可以处理突发流量。当桶满时，能够以最大速度处理请求。这对于需要处理突发流量的应用场景非常有用。
- 限制平均速率：在长期运行中，数据的传输率会被限制在预定义的平均速率（即生成令牌的速率）。
- 灵活性：与漏桶算法相比，令牌桶算法提供了更大的灵活性。例如，可以动态地调整生成令牌的速率。

缺点：

- 可能导致过载：如果令牌产生的速度过快，可能会导致大量的突发流量，这可能会使网络或服务过载。
- 需要存储空间：令牌桶需要一定的存储空间来保存令牌，可能会导致内存资源的浪费。
- 实现稍复杂：相比于计数器算法，令牌桶算法的实现稍微复杂一些。

1.5 适用场景总结

前面我们介绍了四种常用的限流算法：固定窗口算法、滑动窗口算法、漏桶算法和令牌桶算法。每种算法都有其特点和适用场景：

- 固定窗口算法实现简单，但是限流不够平滑，存在窗口边界问题，适用于需要简单实现限流的场景。

    - 电商秒杀活动：在秒杀活动开始时，系统需要限制同时访问的用户数量，以避免服务器过载。固定窗口算法可以根据预设的阈值，在活动开始时拒绝后续的请求，确保系统的稳定性和可用性。
    - API接口限流：对于高流量的API接口，为了避免单个请求对系统造成过大压力，可以使用固定窗口算法来限制同时访问的请求数量。
- 滑动窗口算法解决了窗口边界问题，但是还是存在限流不够平滑的问题，适用于需要控制平均请求速率的场景。
    - 网络流量控制：在保证网络稳定性和可靠性的场景中，可以使用滑动窗口算法来动态调整发送数据的速率和数量。例如，在视频流媒体服务中，可以根据网络带宽和流量负载等参数，动态调整视频流的传输速率，以保证网络的稳定性和流畅性。
    - 服务器资源管理：在服务器资源有限的情况下，可以使用滑动窗口算法来控制请求的处理速率和资源分配。例如，在游戏服务器中，可以根据玩家的数量和请求的频率，动态调整服务器的处理能力和资源分配，以保证游戏的稳定性和流畅性。
- 漏桶算法的优点是流量处理更平滑，但是无法应对突发流量，适用于需要平滑流量的场景。
    - 电商抢购活动：在电商抢购活动中，由于用户数量众多且请求量巨大，可以使用漏桶算法来控制请求的速率和数量。例如，在抢购开始时，系统可以根据预设的阈值和漏桶算法的规则，限制每个用户的请求频率和数量，以避免服务器过载和崩溃。
    - 网络流量整形：在网络流量整形场景中，可以使用漏桶算法来保护系统资源不被打满。例如，在CDN服务中，可以使用漏桶算法来控制流量的速率和数量，以保证系统的稳定性和可用性。
- 令牌桶算法既能平滑流量，又能处理突发流量，适用于需要处理突发流量的场景。
    - 网络流量控制：在需要应对突发流量的场景中，可以使用令牌桶算法来控制网络流量的速率和数量。例如，在视频直播服务中，可以根据用户的数量和请求的频率，动态调整视频流的传输速率和码率，以保证网络的稳定性和流畅性。
    - API调用限制：对于高流量的API接口，可以使用令牌桶算法来限制同时访问的请求数量。例如，在API网关中，可以根据API接口的调用频率和令牌桶算法的规则，限制每个用户的请求频率和数量，以避免服务器过载和崩溃。

2. 拓展：微服务治理标准OpenSergo实战