19.流量控制与熔断利器：Sentinel介绍_V20240120

本篇聊聊流量控制与熔断利器Sentinel，背后的原理，适用的场景及存在的不足。不涉及具体的配置，具体配置请参考官方文档。

1. 前言

在流量控制系列文章中的前五篇，分别介绍了固定时间窗口算法、滑动时间窗口算法、漏桶原理、令牌桶、消息中间件如何应用到分布式环境下流量与并发控制。

我们做个简单回顾：

固定窗口：算法简单，对突然流量响应不够灵活。超过流量的会直接拒绝，通常用于限流。

滑动窗口： 算法简单，对突然流量响应比固定窗口灵活。超过流量的会直接拒绝，通常用于限流。

漏桶算法：在固定窗口的基础之上，使用队列缓冲流量。提供了稳定的流量输出，适用于对流量平滑性有严格要求的场景。

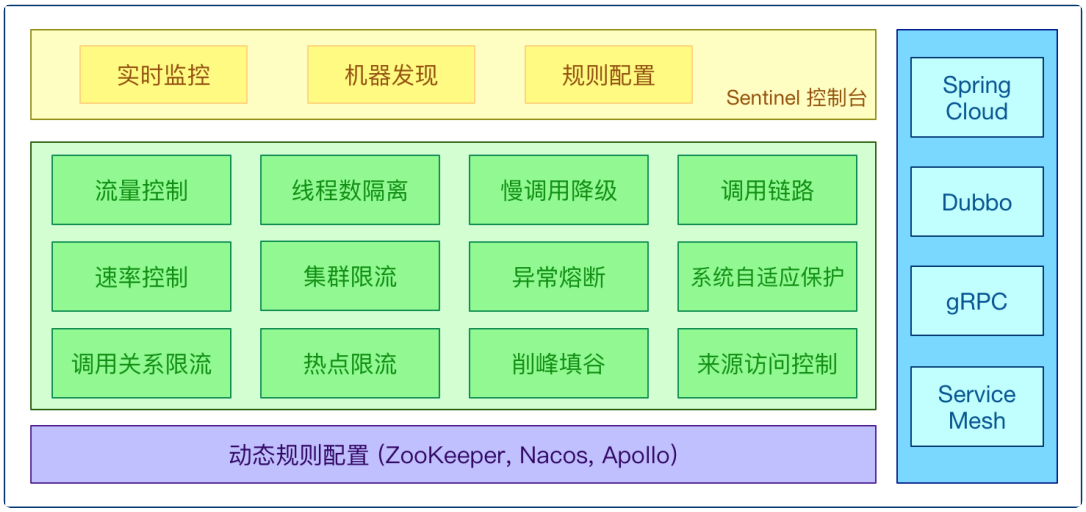
令牌桶算法：在滑动窗口的基础之上，使用队列缓冲流量。提供了稳定的流量输出，且能应对突发流量。

分布式消息中间件：在支付场景的削峰填谷用得比较多，且对精度没有那么苛刻的场景。

今天介绍另一个流量控制和熔断机制利器：Sentinel。

需要说明的是，这里只是做简单介绍，更具体的使用，建议参考官方文档。

2. Sentinel简单介绍



图片来自网络。

Sentinel 是由阿里巴巴开源的一个轻量级的、高性能的流量控制、熔断降级的 Java 库。主要用于在分布式系统中保护服务的稳定性和可靠性，通过实现流量控制、熔断降级、系统负载保护等功能，来防止应用级别的故障和服务级别的雪崩效应。

主要特点

1. **丰富的流量控制策略**：Sentinel 提供了多种流量控制策略，如 QPS、线程数、响应时间等。
2. **熔断降级机制**：当资源的运行指标超过阈值时，Sentinel 可以自动进行熔断降级处理，防止系统过载。
3. **系统负载保护**：能够根据系统的负载情况，如 CPU 使用率、平均负载等，来自动调整流量入口。
4. **实时监控和开放的指标**：提供实时监控和丰富的指标数据，方便用户进行实时的流量监控和调整。
5. **多维度规则授权**：支持多种维度的规则配置，如调用关系、调用来源、API 级别等。
6. **高可用性和可扩展性**：Sentinel 的设计考虑了高可用性，易于扩展，支持与其他组件和服务的集成。
7. **轻量级和高性能**：其对系统的性能影响很小，非常适合高并发的场景。

3. 流量控制

Sentinel 的流量控制基于资源的定义、流量控制规则的设定以及运行时的流量控制处理。在 Sentinel 中，流量控制主要是通过对服务调用或资源访问的限制来实现的。

1. 资源的定义

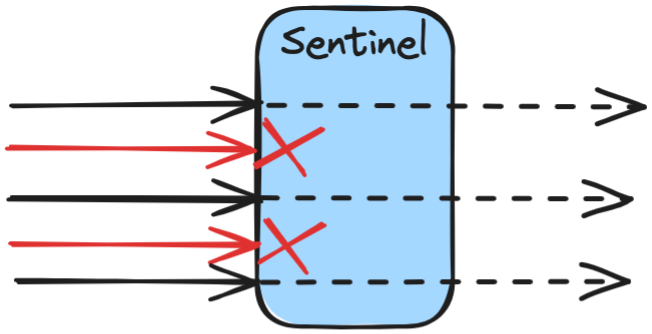
在 Sentinel 中，资源通常是指需要被保护的服务调用点或关键代码段。例如，一个 HTTP 接口、一个内部定义的服务调用或一个数据库查询等都可以被定义为资源。

2. 流量控制规则

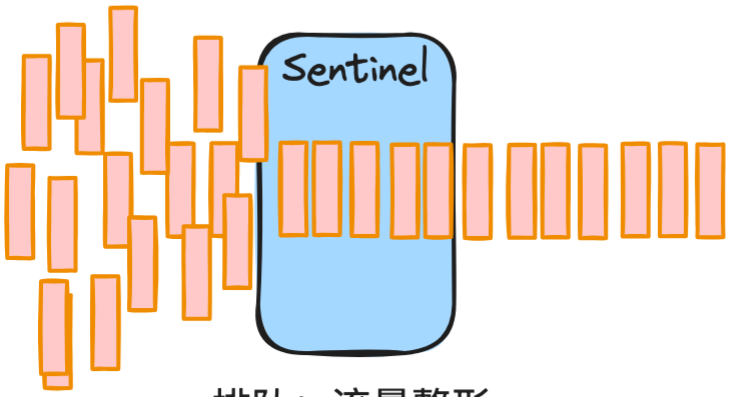
Sentinel 允许对每个资源设置流量控制规则。这些规则可以基于多种不同的标准，例如：

- QPS（每秒查询次数）：限制资源每秒可以处理的请求数。
- 并发线程数：限制资源同时处理的最大并发请求数。
- 响应时间：当资源的响应时间超过设定阈值时，可以触发流量控制措施。

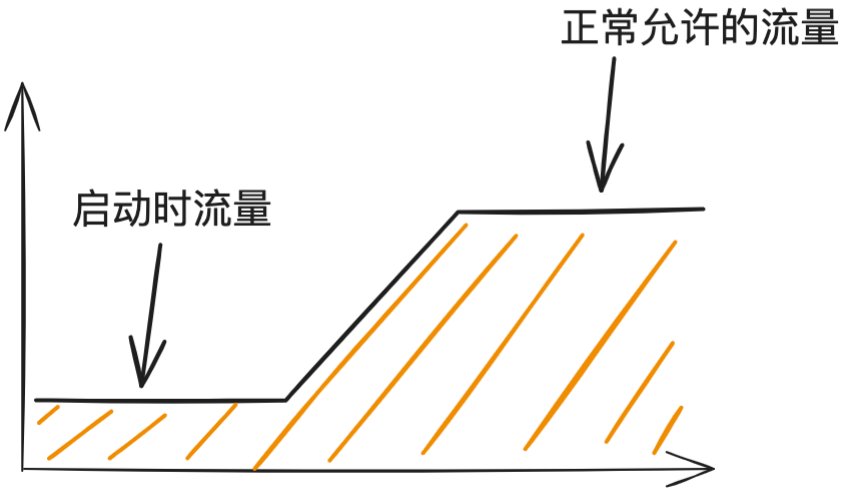
3. 流量控制处理



直接拒绝：快速失败



排队：流量整形



预热：启动阶段保护后台资源

一旦流量达到设定的阈值，Sentinel 会根据配置的流量控制效果来处理额外的流量。主要包括：

- 直接拒绝（Fast Fail）：立即拒绝访问请求，通常用于防止系统过载。

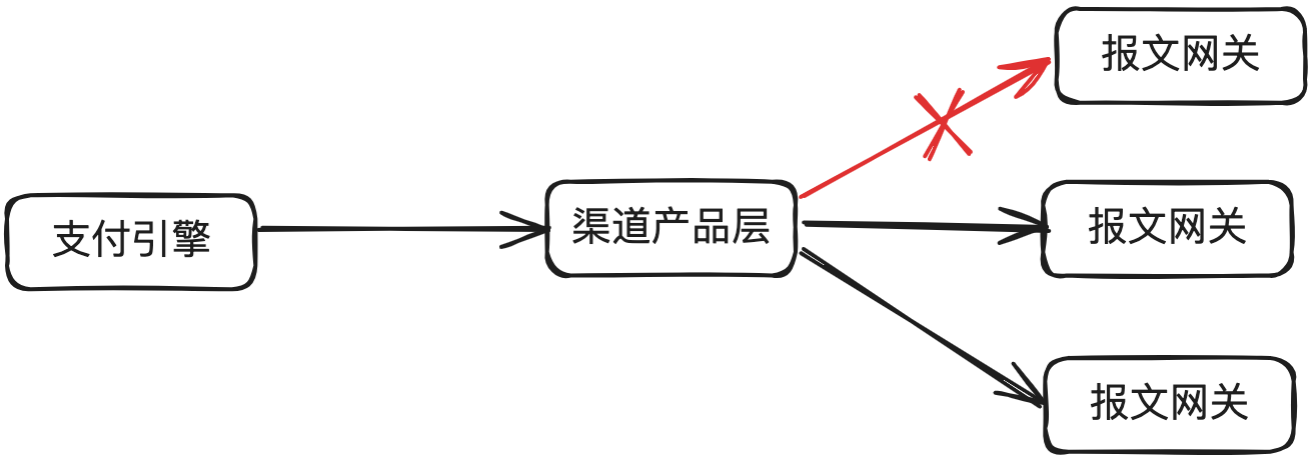
- 预热（Warm Up）：通过逐渐增加流量的方式来预热服务，适用于系统刚启动时。
- 排队等待（Rate Limiter）：使请求排队并逐个处理，保证系统稳定。排队还可以设定超时时间。

4.流量计算与排队

Sentinel 内部也是使用滑动时间窗口算法来计算资源的实时流量。有兴趣的同学可以自己去翻一下源代码。

排队通常使用先进先出（FIFO）的队列实现。

4. 熔断保护与自动恢复



熔断是一种保护系统的策略，当检测到某个微服务不稳定或者响应时间过长时，Sentinel会自动切断对该服务的请求，防止系统雪崩。Sentinel的熔断策略是基于RT（响应时间）、异常比例和异常数等不同的指标来实现的。一旦触发熔断规则，Sentinel会暂时阻断请求，直到服务恢复正常。

1. 熔断策略

Sentinel 提供多种熔断策略，典型的包括：

- 慢调用比例：如果资源的响应时间超过阈值的调用比例超过设定值，则触发熔断。
- 异常比例：如果资源的异常调用比例超过设定值，则触发熔断。
- 异常数：如果在一个统计窗口内，资源的异常数超过设定值，则触发熔断。

2. 统计窗口

- 滑动窗口：Sentinel 通常使用滑动窗口来统计资源的性能指标（如响应时间、调用异常等）。
- 时间间隔：Sentinel 允许配置统计窗口的时间间隔，以便根据应用的实际需求调整熔断的灵敏度。

3. 熔断状态与自动化恢复

Sentinel 的熔断机制有三种状态：

- 关闭：正常状态，请求正常处理。
- 打开：熔断状态，所有对资源的请求都会被立即拒绝。
- 半开：一段时间后，Sentinel 会自动将熔断器置为半开状态，允许部分请求通过以检测资源的健康状态。如果这些请求成功，则关闭熔断器；如果失败，则再次打开熔断器。

5. 支付系统应用场景

Sentinel广泛应用于微服务架构中，可以应对突发流量、分布式服务之间的依赖保护、系统负载过高等情况。在电商、支付、金融等行业随处可见。

我们主要用在各子域之间的限流。每年大促，就会梳理业务请求量，算出流量分布图，根据流量分布图，设置每个子应用各接口的限流值。

6. 无法适用的场景

Sentinel的限流有两种模式：1）总量除以机器数，然后做单机限流。2）拿出一台机器做集群结点，然后做集群限流。

但无论哪种模式都无法解决跨服务单元的限流。比如部署了两个机房，两个机房合用限流到1TPS，Sentinel是无法做到的。这个时候就需要用到我们前面几篇文章中讲到的自己实现的限流方案，比如漏桶，令牌桶等。

7. 结束语

在当前微服务架构的盛行的时代，Sentinel作为一个高效的流量控制与熔断工具，为确保系统的稳定性和可用性提供了强有力的支持。通过使用Sentinel，我们可以大大提高在线支付系统等复杂应用的稳定性和用户体验。

但另一方面，我们也需要知道不同的工具有不同的应用场景，Sentinel解决的是限流和熔断，消息中间件解决削峰填谷和应用间解耦，自己手撸一个漏桶或令牌桶解决极低TPS这种特殊场景。

这是《百图解码支付系统设计与实现》专栏系列文章中的第（19）篇。和墨哥（隐墨星辰）一起深入解码支付系统的方方面面。

欢迎转载。

Github（PDF文档全集，不定时更新）：https://github.com/yinmo-sc/Decoding-Payment-System-Book

公众号：隐墨星辰。

有个小群不定时解答一些问题或知识点，有兴趣的同学可先加微信（yinmo_sc）后进入，添加微信请备注：加支付系统设计与实现讨论群。

隐墨星辰

扫一扫上面的二维码图案，加我为朋友。