17 | Sentinel 体系结构：什么是服务容错（降级熔断、流量整形）？

2022-01-19 姚秋辰

《Spring Cloud 微服务项目实战》 课程介绍 >

讲述：姚秋辰

时长 17:26 大小 15.97M

你好，我是姚秋辰。

今天我们来学习大型微服务系统中高可用性的重要一环：服务容错。通过这节课的内容，你可以了解什么是降级熔断和流量整形，以及它们和服务高可用之间的联系。最后我再带你从架构层面去了解 Sentinel 服务容错的工作流程，为后面的实战课程做一些理论知识的铺垫。

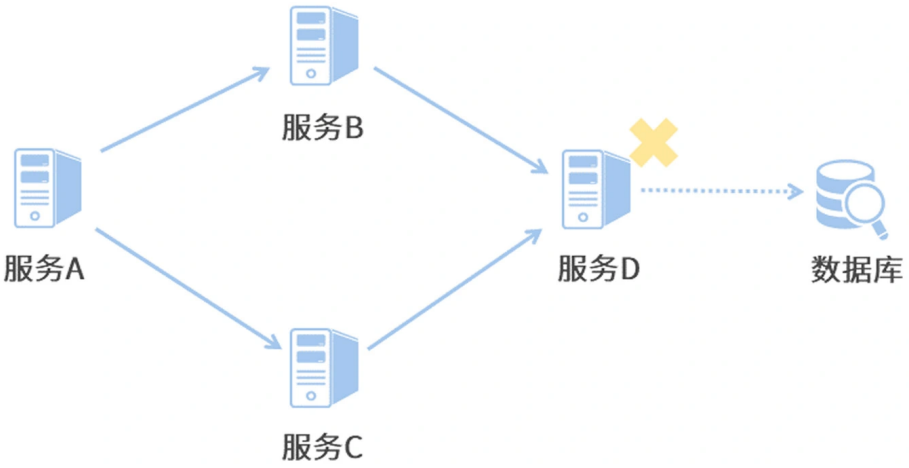
说到高可用，你也许会不由自主地想到“集群化”。没错，通过搭建服务集群来避免单点故障确实是高可用性保障的常规操作。但是，仅仅搭建集群就能高枕无忧了吗？当面对真正的高可用杀手“服务雪崩”时，即便是集群也会显得脆弱无力。

那么服务雪崩在微服务系统中能引起多大的故障呢？在学习什么是服务容错之前，我们先来了解一下服务容错所要解决的实际问题。

什么是服务雪崩？

我来用一个模拟场景带你感受一下服务雪崩的厉害之处。

假设我有一个微服务系统，这个系统内包含了 ABCD 四个微服务，这四个服务都是以集群模式构建的。我画了一张图用来表示各个服务之间的调用关系。



从上面的图中我们可以看出，服务 A 会向服务 B 和服务 C 发起调用，而服务 B 和服务 C 都会去调用服务 D。也就是说，服务 ABC 都直接或间接地依赖服务 D 完成自己的业务逻辑。

由于服务 D 底层有数据读写的需求，所以它会对数据库执行 CRUD 操作。如果开发服务 D 的程序员学艺不精，写了一段性能比较差的 SQL 语句，那么一次 DB 操作的执行时间就会比较长。这在小并发访问量的情况下没有什么问题，不过，一旦并发量堆积了起来，这种性能问题就会被放大。

在这种情况下，数据库的连接资源会被服务 D 迅速吃光，进而导致服务 D 的接口响应时间逐渐拉长，接口超时的情况越来越多。由于上游服务的请求还在源源不断抵达服务 D，所以接口超时会迅速传导到服务 B 和服务 C，进而又影响到 A，导致整个集群的服务变为不可用状态。

来自服务 D 的底层故障，如果不能得到有效处理，那么故障就像滚雪球一样在集群中被迅速放大，进而形成了一场“服务雪崩”，团灭了整个系统的上下游服务。我想，现在你一定理解了服务雪崩的危害。

那么我们如何使用 Sentinel 来防范服务雪崩呢？

Sentinel 服务容错的思路

Sentinel 是 Spring Cloud Alibaba 的一款服务容错组件，我们也经常把它叫做“防流量哨兵”。它是阿里巴巴双十一大促核心场景的保护神，内置了丰富的服务容错应用场景。它以流量作为切入点，通过各种内外防控手段达到维持服务稳定性的目的。

阿里系的 Sentinel 解决服务稳定性的思路是什么呢？

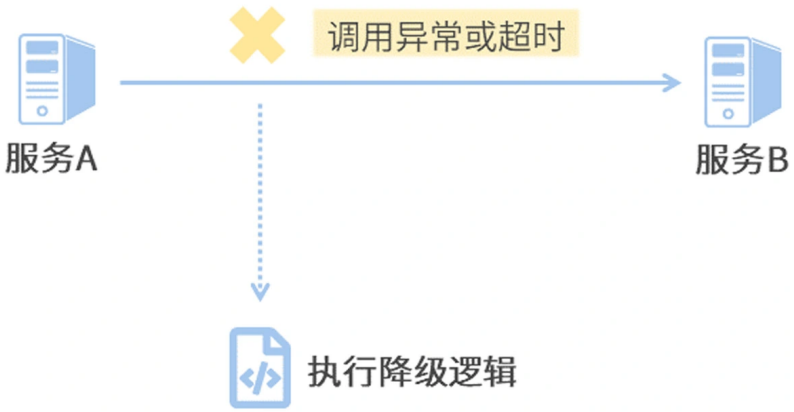
就拿服务雪崩这种稳定性崩坏的场景来说，这种极端场景的发生有两个主要因素，一个是**外部的高并发流量导致的请求数量增多**，超过了集群的吞吐量，另一个是**内部各种未知异常导致的接口响应异常超时**。

我们可以采用“**内外兼修**”的思路来摆平服务雪崩的两个主要因素。所谓“内”，是指内部的异常治理，所谓“外”，则是外部用户流量的疏导。内外兼修的根本目的，是从内部和外部双管齐下对集群访问量进行减压，下面我们深入分析一下 Sentinel 是如何通过内外兼修的手段做服务容错的。

内部异常治理

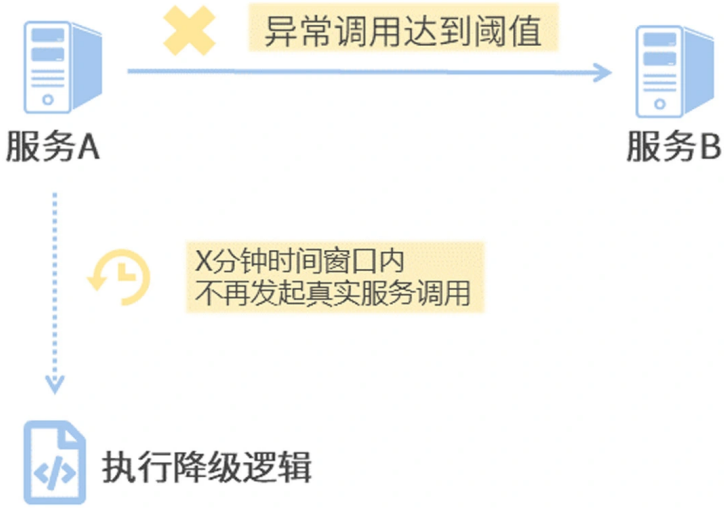
在 Sentinel 中，我们可以采用**降级**和**熔断**的方式处理内部的异常。

所谓降级，是指当服务调用发生了响应超时、服务异常等情况时，我们在服务内部可以执行一段“降级逻辑”。



在**降级**逻辑中，你可以选择静默处理，即忽略掉异常继续执行后续逻辑；或者你可以返回一个让业务可以继续执行下去的默认结果；又或者，在降级逻辑中尝试重试、或者恢复异常服务。从这里我们可以看出，降级是针对“单次服务调用异常”而执行的处理逻辑。

而所谓**熔断**，是指当异常调用量达到一定的判定条件，比如在异常降级和慢调用请求的比例达到一个阈值、窗口时间内降级请求达到一定数量的情况下，微服务在一段时间内停止对目标服务发起调用，所有来访请求直接执行降级逻辑。所以，熔断是“多次服务调用异常”累积的结果。



我们可以看出，当服务进入到了“熔断状态”以后，当前服务对下游目标服务的调用行为也就停止了，这样一来就大大降低了下游服务的访问压力。

关于降级熔断，我的经验是主链路服务（也就是核心业务链路的重要服务）一定要设置降级预案，防止服务雪崩在核心业务上的传播。除此之外，对于非核心链路的服务，应该设置手动降级开关，在大促等高并发场景下做主动降级，将额外的计算资源通过弹性方案匀给主链路服务。

关于降级的判定条件，需要结合全链路压测的结果来判定。通常我们在双 11 这类大促场景下，会组建一个稳定性团队专门负责全链路压测，并根据多轮次的压测结果来调整各个服务集群的水位，并对降级和限流判定条件进行微调，以期达到最佳的主链路吞吐量。

以上是内部异常治理的方法，那么外部流量疏导又是如何降低集群访问压力的呢？

外部流量控制

提到外部流量控制，你一定会想到“**限流**”。没错，限流是流量整形 / 流控方案的一种。在 Sentinel 中我们可以根据集群的处理能力，为每个服务设置一个限流规则，从 QPS 维度或者并发线程数的维度控制外部的访问流量。一旦访问量超过阈值，后续的请求就会被“fast fail”，这是最为常用的一种限流手段。但 Sentinel 所支持的方案可不止这一种。

从流量整形的效果来看，除了“快速失败”以外，我们还可以在 Sentinel 中选择预热模型和排队模型。

顾名思义，**预热模型**就是在一段规定的预热时间窗口内，由低到高逐渐拉高流量阈值，直到达到预设的最高阈值为止。

而对于**排队模型**来说，如果访问量超过了设定的阈值，服务请求不会被立即失败，而是被放入一个队列内等待处理，如果服务请求在预设的超时时间内仍然未被处理，那么就会被移出队列。

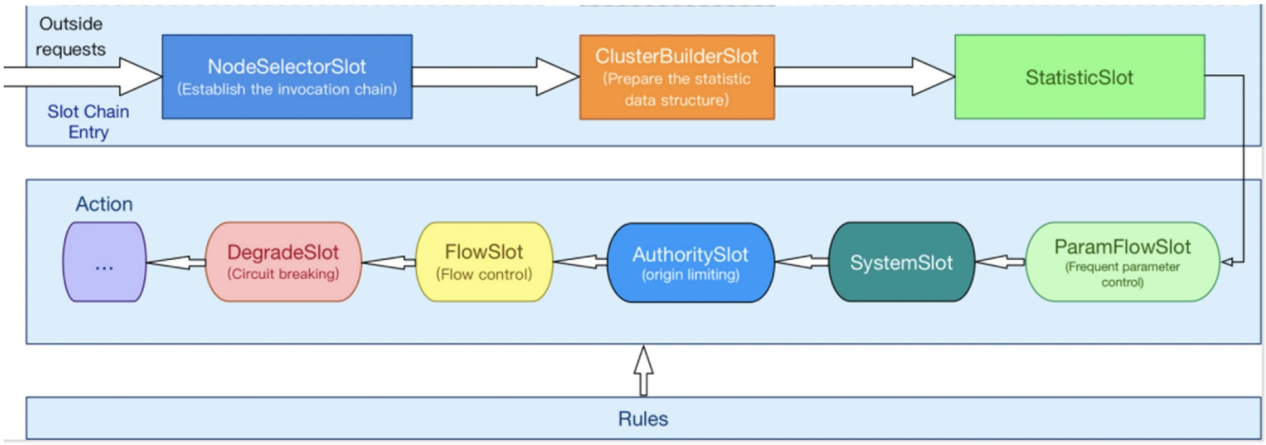
限流是挡在降级熔断之前的一道关卡，它是投入产出比最高的防护措施，为什么这样说呢？我们将限流和降级做一个比对：在熔断阶段，用户流量已经打到了服务器，尽管我们对下游服务的调用不会真实发起，但上游服务的计算资源实打实的被占用了；而限流则不同，如果用户流量在入口处就被限制，那么它并不会占用服务器的资源来处理这个请求。

我会在接下来的两节课中和你详细讲解如何在 Sentinel 中设置熔断规则、流控规则、热点规则等服务容错设置。

到这里，我想你已经了解了 Sentinel “内外兼修”的治理之道。接下来，我们去简单了解一下 Sentinel 的工作原理。

Sentinel 工作原理

我放了一张 Sentinel 官方的流程图，你可以一目了然地看到 Sentinel 对服务请求的处理流程。



在 Sentinel 的世界中，万物都是可以被保护的“资源”，当一个外部请求想要访问 Sentinel 的资源时，便会创建一个 Entry 对象。每一个 Entry 对象都要过五关斩六将，经过 Slot 链路的层层考验最终完成自己的业务，你可以把 Slot 当成是一类完成特定任务的“Filter”，**这是一种典型的职责链设计模式。**

分布在 SlotChain 里面的各个 Slot 各司其职，为了保护 Sentinel 背后的“资源”，这些 Slot 通过互相配合的方式执行了各项检查任务。比如有的 Slot 用来统计数据，而有的 Slot 负责做限流降级检查。我这里挑选几个重要的 Slot 节点讲解一下，让你明白这个 Slot 职责链的核心功能。

在这些 Slot 中，有几个是被专门用来**收集数据**的。比如，**NodeSelectorSlot** 被用来构建当前请求的访问路径，它将上下游调用链串联起来，形成了一个服务调用关系的树状结构。而 **ClusterBuilderSlot** 和 **StatisticSlot** 这两个 Slot 会从多个维度统计一些运行期信息，比如接口响应时间、服务 QPS、当前线程数等等。

由这几个 Slot 统计出来的结果，会为后续的限流降级等 Sentinel 策略提供数据支持。比如说我想为指定的链路定义 QPS 维度的限流策略，那么这个限流策略在执行阶段就需要获取到这些统计数据，作为决策依据。

除了上面介绍的三个 Slot 以外，Sentinel 还有很多被用作“**规则判断**”的 Slot。比如 **FlowSlot** 被用来做流控规则的判定，**DegradeSlot** 被用来做降级熔断判定，这两个 Slot 是我们平时在项目中使用频率最高的服务容错功能。**ParamFlowSlot** 可以根据请求参数做精细粒度的流控，它经常被用来在大型应用中控制热点数据所带来的突发流量。**AuthoritySlot** 可以针对特定资源设置黑白名单，限制某些应用对资源的访问。

除此之外，Sentinel 的 Slot 机制也具备一定的扩展性，如果你想要添加一个自定义的 Slot，我们可以通过实现 ProcessorSlot 接口来完成，而且你还可以通过优先级调整各个 Slot 之间的执行顺序。

到这里，相信你已经对服务容错有了比较清晰的认识，并且了解了 Sentinel 实现服务容错的路子。现在，我们来回顾一下这节课的重点内容。

总结

今天我带你了解了提高服务稳定性的思路，那就是**“内外兼修”，通过降级熔断解决内部异常治理，再通过外部流控削减外部访问压力**。然后我们学习了 Sentinel 的工作原理，了解了它是如何使用职责链的方式做资源访问检查的。

在本节课的 Sentinel 工作原理部分，我列举了很多 Sentinel 规则类的类名，如果你对 Sentinel 的底层原理感兴趣的话，我推荐你从 Sentinel 的 GitHub 主页将源代码下载到本地（建议你下载 1.8.2 版本），然后导入项目到 IDE 中直接打开这几个类，通过源代码来了解一下它的底层实现。这样一来，你就会对 Sentinel 的 SlotChain 职责链有一个更为清晰的认识。

在后面的课程中，我将带你进入到 Sentinel 的实战环节。我会使用两节课将 Sentinel 中的服务容错规则应用到优惠券实战项目中，最后再用一节课对 Sentinel 的源码做个二次改造，将 Sentinel 中的限流规则持久化到 Nacos Config。我建议你在几个关键 Slot 处打上断点，通过 debug 的方式跟一遍代码，这样你就能快速拎清楚 Sentinel 的底层运作原理了。

思考题

通过这节课介绍的内容，如果我想要在 Sentinel 的职责链中做一个小扩展，添加一个自定义的 Slot 节点，并指定它的执行顺序，你知道该如何做吗？你可以试着深入学习我前面提到的几个 Slot 类的源码，我相信你可以从中轻松找到答案。

好啦，这节课就结束啦。欢迎你把这节课分享给更多对 Spring Cloud 感兴趣的朋友。我是姚秋辰，我们下节课再见！

上一篇 16 | 如何集成 Nacos Config 实现配置项动态刷新？

下一篇 18 | Sentinel 实战：如何实施流量整形与限流策略？

精选留言 (1)

写留言

peter
2022-01-19

A “降级”和“熔断”都会执行降级逻辑。但对于“降级”，下一次A服务还会调用B服务，只不过失败后再执行降级逻辑，对吗？
B 我对“降级”的理解是“下线某些不重要的服务”，和本篇所讲的“降级”不一样。是不同层次的“降级”吗？
Q2：预热模型两个问题？...

展开

共 1 条评论 >