上一课时我们分析了限流的常用策略，下面来看一下，高可用的另外两大撒手锏：降级和熔断，关于这两种技术手段如何实施，又有哪些区别呢？

高可用之降级

我们在第 39 课时提过服务降级是电商大促等高并发场景的常见稳定性手段，那你有没有想过，为什么在大促时要开启降级，平时不去应用呢？

在大促场景下，请求量剧增，可我们的系统资源是有限的，服务器资源是企业的固定成本，这个成本不可能无限扩张，所以说，**降级是解决系统资源不足和海量业务请求之间的矛盾。**

降级的具体实现手段是，在暴增的流量请求下，对一些非核心流程业务、非关键业务，进行有策略的放弃，以此来释放系统资源，保证核心业务的正常运行。我们在第 34 课时中提过二八策略，换一个角度，服务降级就是尽量避免这种系统资源分配的不平衡，打破二八策略，让更多的机器资源，承载主要的业务请求。

就如同我们之前的例子中，电商大促时限制退款，但平时并不会限制，所以服务降级不是一个常态策略，而是应对非正常情况下的应急策略。服务降级的结果，通常是对一些业务请求，返回一个统一的结果，你可以理解为是一种 FailOver 快速失败的策略。

举个例子，我们都有在 12306 网站购票的经历，在早期春运抢票时，会有大量的购票者进入请求，如果火车票服务不能支撑，你想一想，是直接失败好呢，还是返回一个空的信息好呢？一般都会返回一个空的信息，这其实是一种限流后的策略，我们从一个广义的角度去理解，限流也是一种服务降级手段，是针对部分请求的降级。

一般来说，降级针对的目标，一般是业务闭环中的一些次要功能，比如大促时的评论、退款功能，从一致性的角度，因为强一致性的保证需要很多系统资源，降级可能会降低某些业务场景的一致性。

具体在进行服务降级操作时，要注意哪些点呢？首先需要注意梳理核心流程，知道哪些业务是可以被牺牲的，比如双十一大家都忙着抢购，这时候一些订单评论之类的边缘功能，就很少有人去使用。另外，要明确开启时间，在系统水位到达一定程度时开启。还记得我们在第 16 课时提到的分布式配置中心吗？降级一般是通过配置的形式，做成一个开关，在高并发的场景中打开开关，开启降级。

高可用之熔断

不知道你有没有股票投资的经验，在很多证券市场上，在大盘发生非常大幅度的波动时，为了保护投资者的利益，维护正常的市场秩序，会采取自动停盘机制，也就是我们常说的**股市熔断。**

在高可用设计中，也有熔断的技术手段，熔断模式保护的是业务系统不被外部大流量或者下游系统的异常而拖垮。

通过添加合理的熔断策略，可以防止系统不断地去请求可能超时和失败的下游业务，跳过下游服务的异常场景，防止被拖垮，也就是防止出现服务雪崩的情况。

熔断策略其实是一种熔断器模式，你可以想象一下家里应用的电路过载保护器，不过熔断器的设计要更复杂，一个设计完善的熔断策略，可以在下游服务异常时关闭调用，在下游服务恢复正常时，逐渐恢复流量。

下面我举一个例子，假设你开发了一个电商的订单服务，你的服务要依赖下游很多其他模块的服务，比如评论服务。现在有一个订单查询的场景，QPS 非常高，但是恰好评论服务因为某些原因部分机器宕机，出现大量调用失败的情况。如果没有熔断机制，订单系统可能会在失败后多次重试，最终导致大量请求阻塞，产生级联的失败，并且影响订单系统的上游服务，出现类似服务雪崩的问题，导致整个系统的响应变慢，可用性降低。

如果开启了熔断，订单服务可以在下游调用出现部分异常时，调节流量请求，比如在出现 10% 的失败后，减少 50% 的流量请求，如果继续出现 50% 的异常，则减少 80% 的流量请求；相应的，在检测的下游服务正常后，首先恢复 30% 的流量，然后是 50% 的流量，接下来是全部流量。

对于熔断策略的具体实现，我建议你查看 Alibaba Sentinel 或者 Netflix Hystrix 的设计，熔断器的实现其实是数据结构中有限状态机（Finite-state Machines，FSM）的一种应用，关于 FSM 的具体分析和应用，不是本课时的目标，因为 FSM 不光在算法领域有应用，在复杂系统设计时，为了更好的标识状态流转，用有限状态机来描述会特别清晰。

熔断器的恢复时间，也就是平均故障恢复时间，称为 MTTR，在稳定性设计中是一个常见的指标，在 Hystrix 的断路器设计中，有以下几个状态。

- Closed：熔断器关闭状态，比如系统检测到下游失败到了 50% 的阈值，会开启熔断。
- Open：熔断器打开状态，此时对下游的调用在内部直接返回错误，不发出请求，但是在一定的时间周期以后，会进入下一个半熔断状态。
- Half-Open：半熔断状态，允许少量的服务请求，如果调用都成功（或一定比例）则认为恢复了，关闭熔断器，否则认为还没好，又回到熔断器打开状态。

在系统具体实现中，降级和熔断推荐使用成熟的中间件，包括 Sentinel 和 Hystrix，以及 resilience4j，关于这几种组件的应用细节，这里暂不做展开分析，我一直觉得，授人以鱼不如授人以渔，在解决了原理层面以后，如何实现就变得简单很多。

我在工作中应用 Sentinel 比较多，你可以在Sentinel 官网看到详细的介绍，下面是对这几种组件的对比，来自阿里巴巴 Sentinel 开发团队的分享，作为补充资料：

	Sentinel	**Hystrix**	**resilience4j**
隔离策略	信号量隔离（并发线程数限流）	线程池隔离/信号量隔离	信号量隔离
熔断降级策略	基于响应时间、异常比率、异常数	基于异常比率	基于异常比率、响应时间
实时统计实现	滑动窗口（LeapArray）	滑动窗口（基于 RxJava）	Ring Bit Buffer
动态规则配置	支持多种数据源	支持多种数据源	有限支持
扩展性	多个扩展点	插件的形式	接口的形式
基于注解的支持	支持	支持	支持

	Sentinel	Hystrix	resilience4j
限流	基于 QPS，支持基于调用关系的限流	有限的支持	Rate Limiter
流量整形	支持预热模式、匀速器模式、预热排队模式	不支持	简单的 Rate Limiter 模式
系统自适应保护	支持	不支持	不支持
控制台	提供开箱即用的控制台，可配置规则、查看秒级监控、机器发现等	简单的监控查看	不提供控制台，可对接其他监控系统

总结

以上就是这一课时的内容，和大家总结了降级和熔断的概念，应用场景和实现手段，通过一些应用实例进行了对比。

不知道你有没有发现，在系统设计中，特别是高可用模块，和生活里的一些博弈策略息息相关，不是一个纯技术领域的工作。比如在中国象棋策略中，有个成语叫作丢车保帅，和服务降级有异曲同工之妙，敌人已经攻打过来了，这时候是保护元帅不被将军，还是丢弃一些军备，下次还能卷土重来呢？而服务降级就是放弃一些非关键功能，保证整体系统的运行。

这一课时中，我也列举了股票熔断、漏电保护器等生活实例，希望大家可以扩展思考一下，高可用设计中的博弈在生活中有哪些体现，也欢迎留言分享你的观点。

精选评论