超时 - 重试 - 幂等

> 💡 本期精彩内容：
>
> - 介绍超时、重试的应用场景
> - 分别聊聊超时和重试的细节
> - 接口的幂等性是什么

超时 & 重试

应用场景

只要涉及网络调用、服务器宕机等问题，就需要设置超时和重试，例如：微服务内部的调用、对数据库、缓存、MQ、第三方接口、中间件等的调用。

超时机制

基本概念

当请求超过设置时间还没被处理，则直接被取消，抛出超时异常。

目的是，尽量不在服务端堆积请求连接，既会影响新请求的处理，也可能导致系统崩溃。

如果没有超时控制，可能会导致连接泄漏、线程泄漏。

链路超时控制

确定全链路的超时时长，在协议头的元数据上传递超时时间戳，就能实现整个链路的超时控制。

实现细节：设置超时时间

根据响应时间调整：TP99（或 TP999）都在 x 时间范围内，那么超时时间可以设置为 99 线。

- 接入可观测工具，加监控，确定 TP99
- 做压力测试，确定 TP 99

大厂实践：只要与第三方打交道，针对不同的调用下游（数据库、Redis、Kafka、第三方接口等），都设置不同的超时时间。

重试机制

基本概念

调用第三方接口时，搭配超时来用，多次发送相同的请求，避免网络抖动和偶然故障。

目的是，尽可能让请求被成功处理。由于偶然故障发生频率小，重试对服务器的资源消耗可以忽略不计。

实现细节：设置重试次数

设置重试次数、重试间隔。重试次数不宜过多，否则会给系统负载带来压力，造成系统雪崩。

幂等

基本概念

> f(x) = f(f(x))，例如求绝对值的函数

- 核心思想
  - 保证同一个请求不被多次执行。
- 发生场景
  - 请求的响应结果是超时，并不能确定是**服务端没处理**（前方请求堆积，排不上队），还是服务端处理了发送响应时，碰到了**网络抖动**导致超时。
- 重试困境
  - 执行重试，可能会出现请求多次执行的情况。
  - 不重试，可能出现请求一次都没被执行的情况。

设计幂等接口：幂等去重

> 非幂等操作 → 幂等操作

针对写请求的接口，对请求进行去重，确保同一个请求处理一次和多次的结果是相同的，即幂等接口。

去重逻辑：

- 请求方每次请求生成唯一的 ID，在首次调用和重试时，唯一的 ID 保持不变。
- 服务端收到请求时，查询 ID 是否被处理过，处理过则直接返回结果，不再重复执行业务逻辑。

延伸：幂等相关文章

- HTTP 的幂等性
- 分布式锁如何实现幂等性

实战！聊聊幂等设计 - 掘金

44 | 弹力设计篇之“幂等性设计”-极客时间