# 常见容错方案

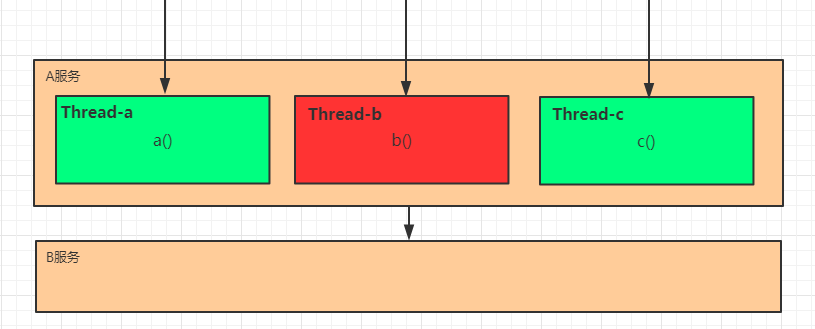
要防止雪崩的扩散，我们就要做好服务的容错，容错说白了就是保护自己不被猪队友拖垮的一些措 施, 下面介绍常见的服务容错思路和组件。

常见的容错思路

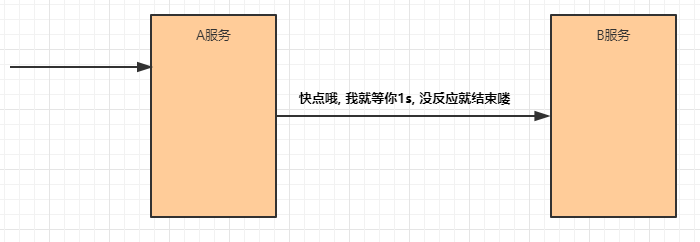
常见的容错思路有隔离、超时、限流、熔断、降级这几种，下面分别介绍一下。

## 隔离

它是指将系统按照一定的原则划分为若干个服务模块，各个模块之间相对独立，无强依赖。当有故 障发生时，能将问题和影响隔离在某个模块内部，而不扩散风险，不波及其它模块，不影响整体的 系统服务。常见的隔离方式有：线程池隔离和信号量隔离．

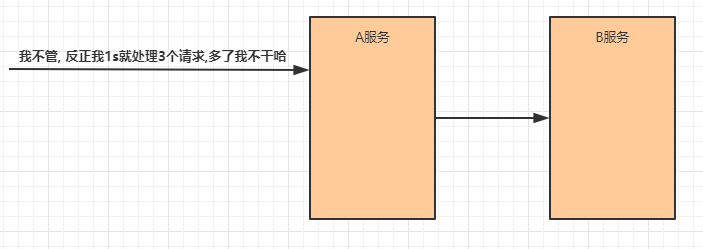


## 超时

在上游服务调用下游服务的时候，设置一个最大响应时间，如果超过这个时间，下游未作出反应， 就断开请求，释放掉线程。

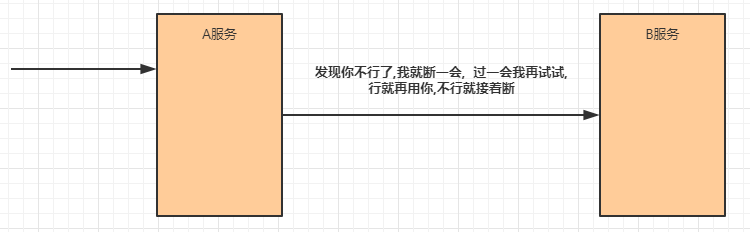
## 限流

限流就是限制系统的输入和输出流量已达到保护系统的目的。为了保证系统的稳固运行,一旦达到 的需要限制的阈值,就需要限制流量并采取少量措施以完成限制流量的目的。



## 熔断

在互联网系统中，当下游服务因访问压力过大而响应变慢或失败，上游服务为了保护系统整体的可用性，可以暂时切断对下游服务的调用。这种牺牲局部，保全整体的措施就叫做熔断。



服务熔断一般有三种状态：

熔断关闭状态（Closed）

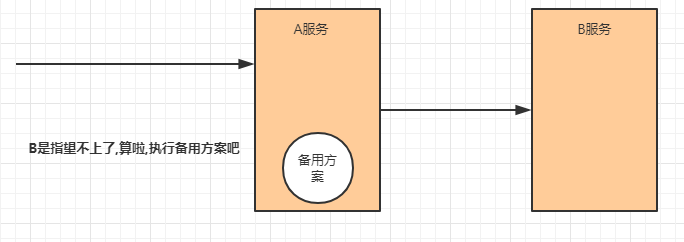
服务没有故障时，熔断器所处的状态，对调用方的调用不做任何限制熔断开启状态（Open）

后续对该服务接口的调用不再经过网络，直接执行本地的fallback方法

半熔断状态（Half-Open）

尝试恢复服务调用，允许有限的流量调用该服务，并监控调用成功率。如果成功率达到预 期，则说明服务已恢复，进入熔断关闭状态；如果成功率仍旧很低，则重新进入熔断关闭状 态。

## 降级

降级其实就是为服务提供一个托底方案，一旦服务无法正常调用，就使用托底方案。

# 常见的容错组件

## Hystrix

Hystrix是由Netﬂix开源的一个延迟和容错库，用于隔离访问远程系统、服务或者第三方库，防止 级联失败，从而提升系统的可用性与容错性。

## Resilience4J

Resilicence4J一款非常轻量、简单，并且文档非常清晰、丰富的熔断工具，这也是Hystrix官方推 荐的替代产品。不仅如此，Resilicence4j还原生支持Spring Boot 1.x/2.x，而且监控也支持和prometheus等多款主流产品进行整合。

## Sentinel

Sentinel 是阿里巴巴开源的一款断路器实现，本身在阿里内部已经被大规模采用，非常稳定。

## 下面是三个组件在各方面的对比：

|  |  |  |  |
| --- | --- | --- | --- |
|  | **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 |
| 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，可对接其它监控系统 |

