高可用性这个topic，然后咱们会用几讲的时间来讲解一下如何用hystrix，来构建高可用的服务的架构

咱们会用一个真实的项目背景，作为业务场景，来带出来在这个特定的业务场景下，可能会产生哪些各种各样的可用性的一些问题

针对这些问题，我们用hystrix的解决方案和原理是什么

带着大家，纯手工将所有的服务的高可用架构的代码，全部纯手工自己敲出来

形成高可用服务架构的项目实战的一个课程

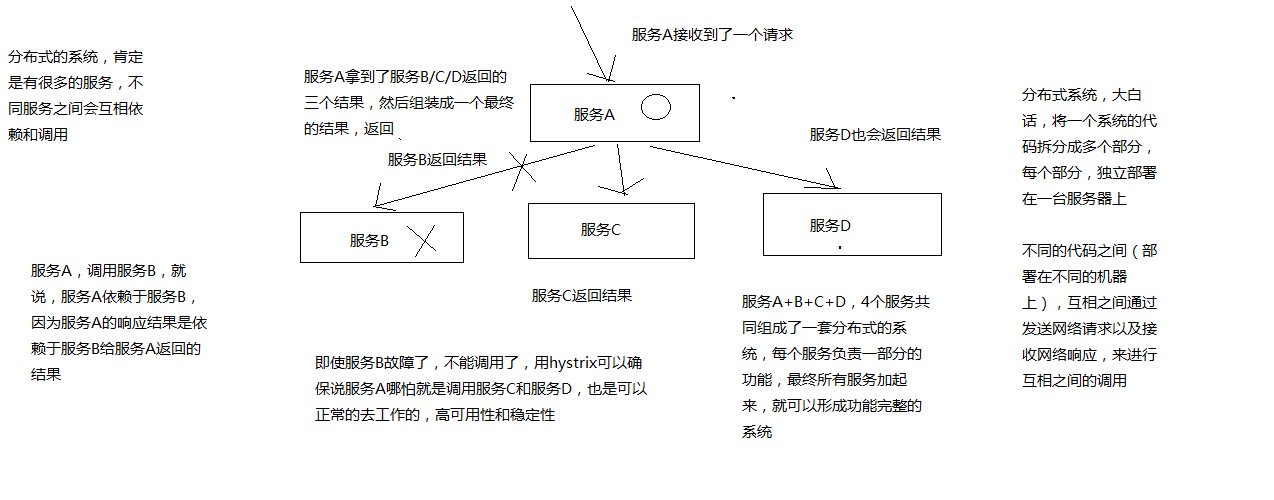
1、Hystrix是什么？

在分布式系统中，每个服务都可能会调用很多其他服务，被调用的那些服务就是依赖服务，有的时候某些依赖服务出现故障也是很正常的。

Hystrix可以让我们在分布式系统中对服务间的调用进行控制，加入一些调用延迟或者依赖故障的容错机制。

Hystrix通过将依赖服务进行资源隔离，进而组织某个依赖服务出现故障的时候，这种故障在整个系统所有的依赖服务调用中进行蔓延，同时Hystrix还提供故障时的fallback降级机制

总而言之，Hystrix通过这些方法帮助我们提升分布式系统的可用性和稳定性



什么是分布式系统及其中的故障

2、Hystrix的历史

hystrix，就是一种高可用保障的一个框架，类似于spring（ioc，mvc），mybatis，activiti，lucene，框架，预先封装好的为了解决某个特定领域的特定问题的一套代码库

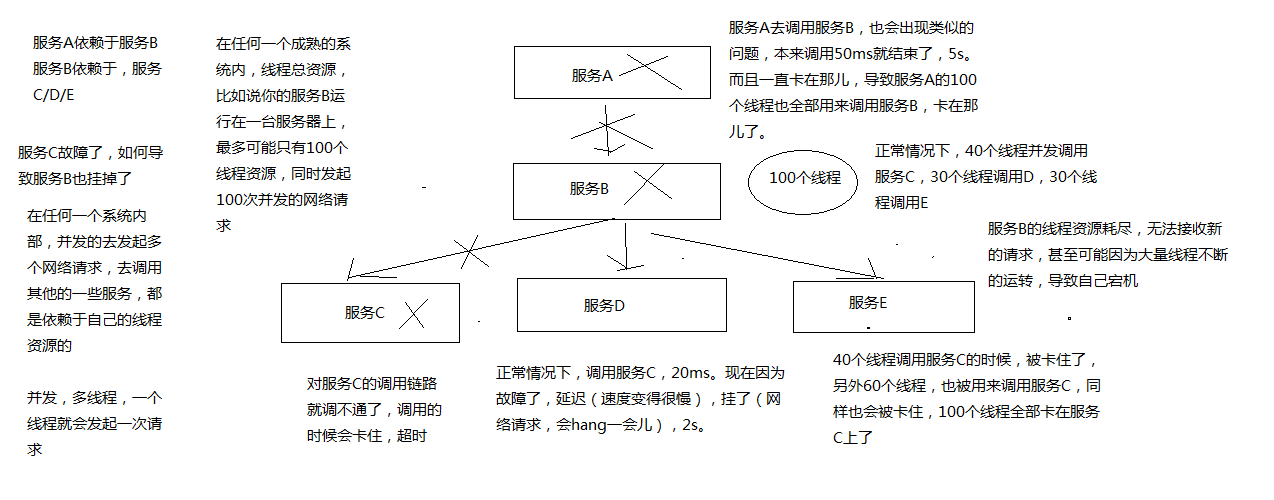
框架，用了框架之后，来解决这个领域的特定的问题，就可以大大减少我们的工作量，提升我们的工作质量和工作效率，框架

hystrix，高可用性保障的一个框架

Netflix（可以认为是国外的优酷或者爱奇艺之类的视频网站），API团队从2011年开始做一些提升系统可用性和稳定性的工作，Hystrix就是从那时候开始发展出来的。

在2012年的时候，Hystrix就变得比较成熟和稳定了，Netflix中，除了API团队以外，很多其他的团队都开始使用Hystrix。

时至今日，Netflix中每天都有数十亿次的服务间调用，通过Hystrix框架在进行，而Hystrix也帮助Netflix网站提升了整体的可用性和稳定性



依赖服务的故障导致服务被拖垮以及故障的蔓延

3、初步看一看Hystrix的设计原则是什么？

hystrix为了实现高可用性的架构，设计hystrix的时候，一些设计原则是什么？？？

（1）对依赖服务调用时出现的调用延迟和调用失败进行控制和容错保护

（2）在复杂的分布式系统中，阻止某一个依赖服务的故障在整个系统中蔓延，服务A->服务B->服务C，服务C故障了，服务B也故障了，服务A故障了，整套分布式系统全部故障，整体宕机

（3）提供fail-fast（快速失败）和快速恢复的支持

（4）提供fallback优雅降级的支持

（5）支持近实时的监控、报警以及运维操作

调用延迟+失败，提供容错

阻止故障蔓延

快速失败+快速恢复

降级

监控+报警+运维

完全描述了hystrix的功能，提供整个分布式系统的高可用的架构

4、Hystrix要解决的问题是什么？

在复杂的分布式系统架构中，每个服务都有很多的依赖服务，而每个依赖服务都可能会故障

如果服务没有和自己的依赖服务进行隔离，那么可能某一个依赖服务的故障就会拖垮当前这个服务

举例来说，某个服务有30个依赖服务，每个依赖服务的可用性非常高，已经达到了99.99%的高可用性

那么该服务的可用性就是99.99%的30次方，也就是99.7%的可用性

99.7%的可用性就意味着3%的请求可能会失败，因为3%的时间内系统可能出现了故障不可用了

对于1亿次访问来说，3%的请求失败，也就意味着300万次请求会失败，也意味着每个月有2个小时的时间系统是不可用的

在真实生产环境中，可能更加糟糕

上面也就是说，即使你每个依赖服务都是99.99%高可用性，但是一旦你有几十个依赖服务，还是会导致你每个月都有几个小时是不可用的

画图分析说，当某一个依赖服务出现了调用延迟或者调用失败时，为什么会拖垮当前这个服务？以及在分布式系统中，故障是如何快速蔓延的？

5、再看Hystrix的更加细节的设计原则是什么？

（1）阻止任何一个依赖服务耗尽所有的资源，比如tomcat中的所有线程资源

（2）避免请求排队和积压，采用限流和fail fast来控制故障

（3）提供fallback降级机制来应对故障

（4）使用资源隔离技术，比如bulkhead（舱壁隔离技术），swimlane（泳道技术），circuit breaker（短路技术），来限制任何一个依赖服务的故障的影响

（5）通过近实时的统计/监控/报警功能，来提高故障发现的速度

（6）通过近实时的属性和配置热修改功能，来提高故障处理和恢复的速度

（7）保护依赖服务调用的所有故障情况，而不仅仅只是网络故障情况

调用这个依赖服务的时候，client调用包有bug，阻塞，等等，依赖服务的各种各样的调用的故障，都可以处理

6、Hystrix是如何实现它的目标的？

（1）通过HystrixCommand或者HystrixObservableCommand来封装对外部依赖的访问请求，这个访问请求一般会运行在独立的线程中，资源隔离

（2）对于超出我们设定阈值的服务调用，直接进行超时，不允许其耗费过长时间阻塞住。这个超时时间默认是99.5%的访问时间，但是一般我们可以自己设置一下

（3）为每一个依赖服务维护一个独立的线程池，或者是semaphore，当线程池已满时，直接拒绝对这个服务的调用

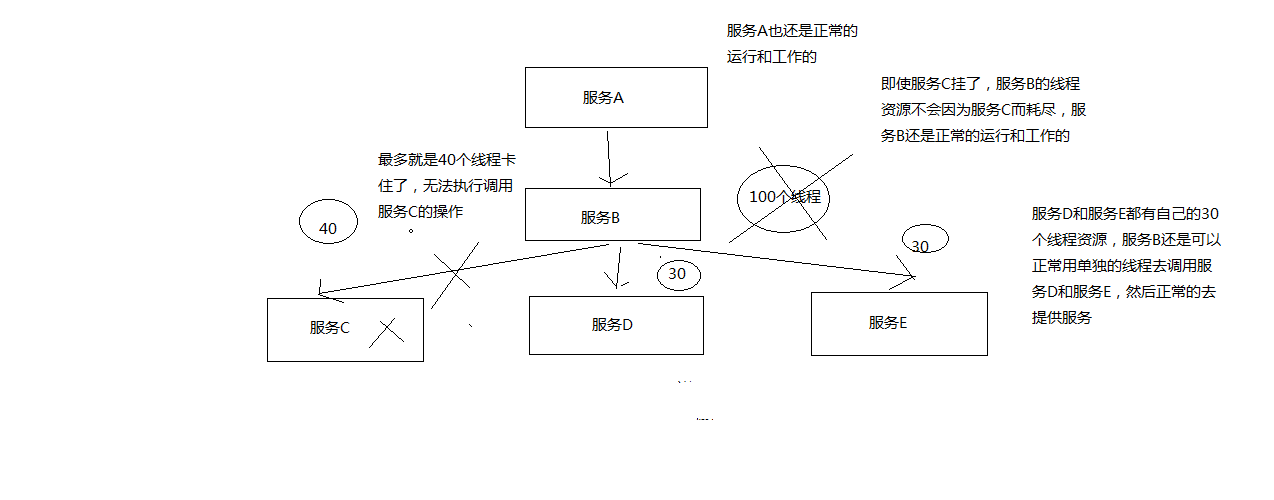
（4）对依赖服务的调用的成功次数，失败次数，拒绝次数，超时次数，进行统计

（5）如果对一个依赖服务的调用失败次数超过了一定的阈值，自动进行熔断，在一定时间内对该服务的调用直接降级，一段时间后再自动尝试恢复

（6）当一个服务调用出现失败，被拒绝，超时，短路等异常情况时，自动调用fallback降级机制

（7）对属性和配置的修改提供近实时的支持

画图分析，对依赖进行资源隔离后，如何避免依赖服务调用延迟或失败导致当前服务的故障



资源隔离如何保护依赖服务的故障不要拖垮整个系统

《亿级流量电商详情页系统的大型高并发与高可用缓存架构实战》

1、亿级流量的电商网站的商品详情页系统架构

2、大型的缓存架构，支撑高并发与高可用

3、几十万QPS的高并发+99.99%高可用+1T以上的海量数据+绝对数据安全的redis集群架构

4、高并发场景下的数据库+缓存双写一致性保障方案

5、大缓存的维度化拆分方案

6、基于双层nginx部署架构的缓存命中率提升方案

7、基于kafka+spring boot+ehcache+redis+nginx+lua的多级缓存架构

8、基于zookeeper的缓存并发更新安全保障方案

9、基于storm+zookeeper的大规模缓存预热解决方案

10、基于storm+zookeeper+nginx+lua的热点缓存自动降级与恢复解决方案

11、基于hystrix的高可用缓存服务架构

12、hystrix的进阶高可用架构方案、架构性能优化以及监控运维

13、基于hystrix的大规模缓存雪崩解决方案

14、高并发场景下的缓存穿透解决方案

15、高并发场景下的缓存失效解决方案