31 | 如何应对接口级的故障？

2018-07-07 李运华

讲述：黄洲君
时长 10:17 大小 4.71M

异地多活方案主要应对系统级的故障，例如，机器宕机、机房故障、网络故障等问题，这些系统级的故障虽然影响很大，但发生概率较小。在实际业务运行过程中，还有另外一种故障影响可能没有系统级那么大，但发生的概率较高，这就是今天我要与你聊的如何应对接口级的故障。

接口级故障的典型表现就是系统并没有宕机，网络也没有中断，但业务却出现问题了。例如，业务响应缓慢、大量访问超时、大量访问出现异常（给用户弹出提示“无法连接数据库”），这类问题的主要原因在于系统压力太大、负载太高，导致无法快速处理业务请求，由此引发更多的后续问题。例如，最常见的数据库慢查询将数据库的服务器资源耗尽，导致读写超时，业务读写数据库时要么无法连接数据库、要么超时，最终用户看到的现象就是访问很慢，一会访问抛出异常，一会访问又是正常结果。

导致接口级故障的原因一般有下面几种：

内部原因

程序 bug 导致死循环，某个接口导致数据库慢查询，程序逻辑不完善导致耗尽内存等。

外部原因

黑客攻击、促销或者抢购引入了超出平时几倍甚至几十倍的用户，第三方系统大量请求，第三方系统响应缓慢等。

解决接口级故障的核心思想和异地多活基本类似：**优先保证核心业务**和**优先保证绝大部分用户**。

降级

降级指系统将某些业务或者接口的功能降低，可以是只提供部分功能，也可以是完全停掉所有功能。例如，论坛可以降级为只能看帖子，不能发帖子；也可以降级为只能看帖子和评论，不能发评论；而 App 的日志上传接口，可以完全停掉一段时间，这段时间内 App 都不能上传日志。

降级的核心思想就是丢车保帅，优先保证核心业务。例如，对于论坛来说，90% 的流量是看帖子，那我们就优先保证看帖的功能；对于一个 App 来说，日志上传接口只是一个辅助的功能，故障时完全可以停掉。

常见的实现降级的方式有：

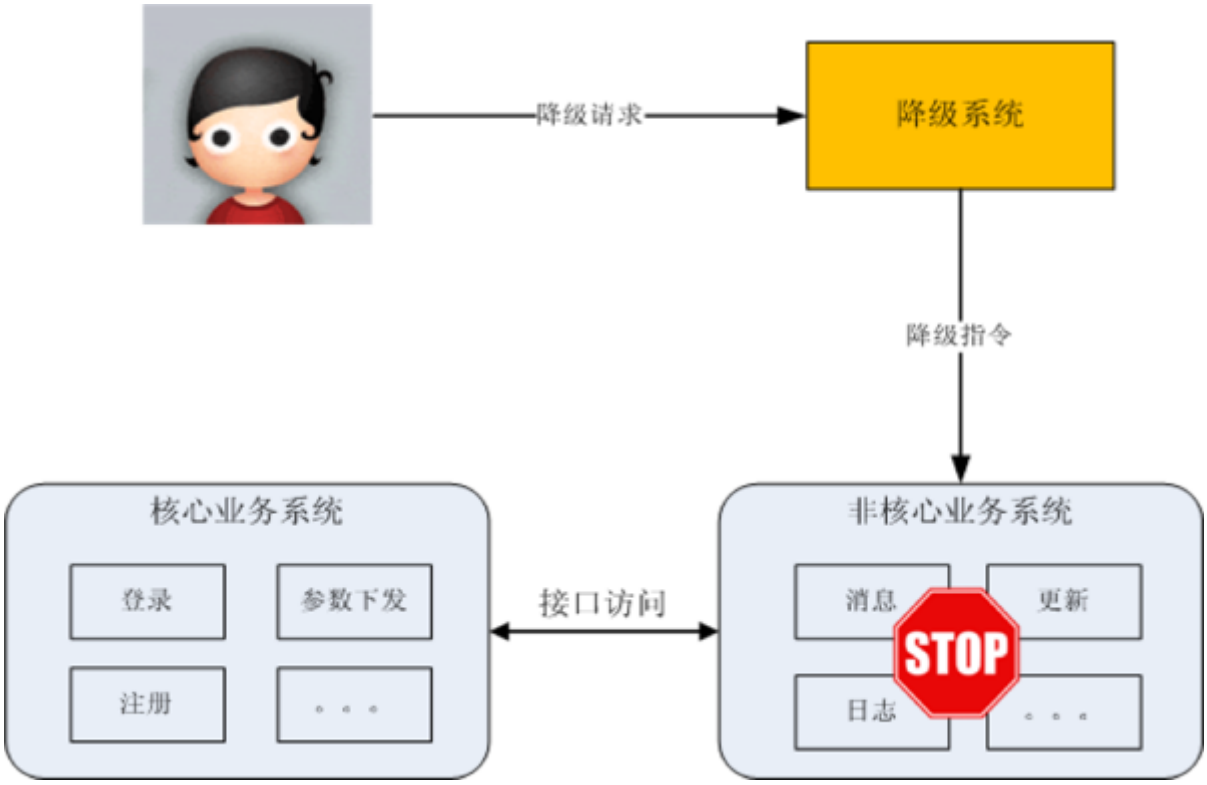
系统后门降级

简单来说，就是系统预留了后门用于降级操作。例如，系统提供一个降级 URL，当访问这个 URL 时，就相当于执行降级指令，具体的降级指令通过 URL 的参数传入即可。这种方案有一定的安全隐患，所以也会在 URL 中加入密码这类安全措施。

系统后门降级的方式实现成本低，但主要缺点是如果服务器数量多，需要一台一台去操作，效率比较低，这在故障处理争分夺秒的场景下是比较浪费时间的。

独立降级系统

为了解决系统后门降级方式的缺点，将降级操作独立到一个单独的系统中，可以实现复杂的权限管理、批量操作等功能。其基本架构如下：



熔断

熔断和降级是两个比较容易混淆的概念，因为单纯从名字上看好像都有禁止某个功能的意思，但其实内在含义是不同的，原因在于降级的目的是应对系统自身的故障，而熔断的目的是应对依赖的外部系统故障的情况。

假设一个这样的场景：A 服务的 X 功能依赖 B 服务的某个接口，当 B 服务的接口响应很慢的时候，A 服务的 X 功能响应肯定也会被拖慢，进一步导致 A 服务的线程都被卡在 X 功能处理上，此时 A 服务的其他功能都会被卡住或者响应非常慢。这时就需要熔断机制了，即：A 服务不再请求 B 服务的这个接口，A 服务内部只要发现是请求 B 服务的这个接口就立即返回错误，从而避免 A 服务整个被拖慢甚至拖死。

熔断机制实现的关键是需要有一个统一的 API 调用层，由 API 调用层来进行采样或者统计，如果接口调用散落在代码各处就没法进行统一处理了。

熔断机制实现的另外一个关键是阈值的设计，例如 1 分钟内 30% 的请求响应时间超过 1 秒就熔断，这个策略中的“1 分钟”“30%”“1 秒”都对最终的熔断效果有影响。实践中

一般都是先根据分析确定阈值，然后上线观察效果，再进行调优。

限流

降级是从系统功能优先级的角度考虑如何应对故障，而限流则是从用户访问压力的角度来考虑如何应对故障。限流指只允许系统能够承受的访问量进来，超出系统访问能力的请求将被丢弃。

虽然“丢弃”这个词听起来让人不太舒服，但保证一部分请求能够正常响应，总比全部请求都不能响应要好得多。

限流一般都是系统内实现的，常见的限流方式可以分为两类：基于请求限流和基于资源限流。

基于请求限流

基于请求限流指从外部访问的请求角度考虑限流，常见的方式有：限制总量、限制时间量。

限制总量的方式是限制**某个指标的累积上限**，常见的是限制当前系统服务的用户总量，例如某个直播间限制总用户数上限为 100 万，超过 100 万后新的用户无法进入；某个抢购活动商品数量只有 100 个，限制参与抢购的用户上限为 1 万个，1 万以后的用户直接拒绝。限制时间量指限制**一段时间内某个指标的上限**，例如，1 分钟内只允许 10000 个用户访问，每秒请求峰值最高为 10 万。

无论是限制总量还是限制时间量，共同的特点都是实现简单，但在实践中面临的主要问题是比较难以找到合适的阈值，例如系统设定了 1 分钟 10000 个用户，但实际上 6000 个用户的时候系统就扛不住了；也可能达到 1 分钟 10000 用户后，其实系统压力还不大，但此时已经开始丢弃用户访问了。

即使找到了合适的阈值，基于请求限流还面临硬件相关的问题。例如一台 32 核的机器和 64 核的机器处理能力差别很大，阈值是不同的，可能有的技术人员以为简单根据硬件指标进行数学运算就可以得出来，实际上这样是不可行的，64 核的机器比 32 核的机器，业务处理性能并不是 2 倍的关系，可能是 1.5 倍，甚至可能是 1.1 倍。

为了找到合理的阈值，通常情况下可以采用性能压测来确定阈值，但性能压测也存在覆盖场景有限的问题，可能出现某个性能压测没有覆盖的功能导致系统压力很大；另外一种方式是

逐步优化，即：先设定一个阈值然后上线观察运行情况，发现不合理就调整阈值。

基于上述的分析，根据阈值来限制访问量的方式更多的适应于业务功能比较简单的系统，例如负载均衡系统、网关系统、抢购系统等。

基于资源限流

基于请求限流是从系统外部考虑的，而基于资源限流是从系统内部考虑的，即：找到系统内部影响性能的关键资源，对其使用上限进行限制。常见的内部资源有：连接数、文件句柄、线程数、请求队列等。

例如，采用 Netty 来实现服务器，每个进来的请求都先放入一个队列，业务线程再从队列读取请求进行处理，队列长度最大值为 10000，队列满了就拒绝后面的请求；也可以根据 CPU 的负载或者占用率进行限流，当 CPU 的占用率超过 80% 的时候就开始拒绝新的请求。

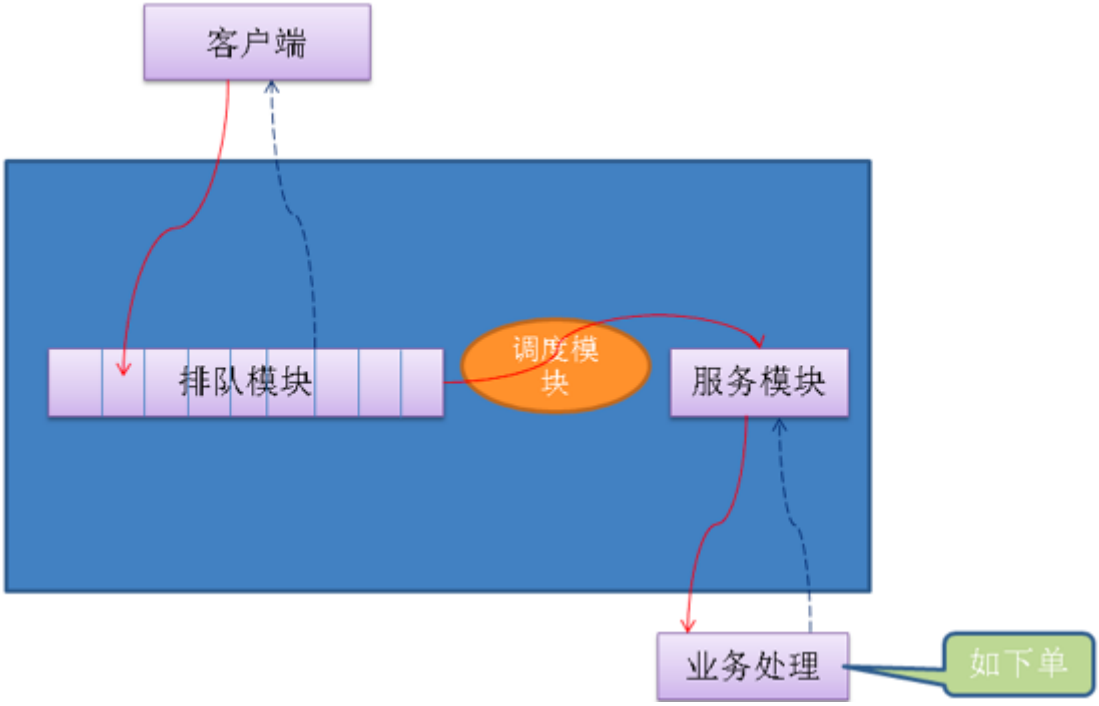
基于资源限流相比基于请求限流能够更加有效地反映当前系统的压力，但实践中设计也面临两个主要的难点：如何确定关键资源，如何确定关键资源的阈值。通常情况下，这也是一个逐步调优的过程，即：设计的时候先根据推断选择某个关键资源和阈值，然后测试验证，再上线观察，如果发现不合理，再进行优化。

排队

排队实际上是限流的一个变种，限流是直接拒绝用户，排队是让用户等待一段时间，全世界最有名的排队当属 12306 网站排队了。排队虽然没有直接拒绝用户，但用户等了很长时间后进入系统，体验并不一定比限流好。

由于排队需要临时缓存大量的业务请求，单个系统内部无法缓存这么多数据，一般情况下，排队需要用独立的系统去实现，例如使用 Kafka 这类消息队列来缓存用户请求。

下面是 1 号店的“双 11”秒杀排队系统架构



其基本实现摘录如下：

> 【排队模块】
> 负责接收用户的抢购请求，将请求以先入先出的方式保存下来。每一个参加秒杀活动的商品保存一个队列，队列的大小可以根据参与秒杀的商品数量（或加点余量）自行定义。
> 【调度模块】
> 负责排队模块到服务模块的动态调度，不断检查服务模块，一旦处理能力有空闲，就从排队队列头上把用户访问请求调入服务模块，并负责向服务模块分发请求。这里调度模块扮演一个中介的角色，但不只是传递请求而已，它还担负着调节系统处理能力的重任。我们可以根据服务模块的实际处理能力，动态调节向排队系统拉取请求的速度。
> 【服务模块】
> 负责调用真正业务来处理服务，并返回处理结果，调用排队模块的接口回写业务处理结果。

小结

今天我为你讲了接口级故障的四种应对方法，分别是降级、熔断、限流和排队，希望对你有所帮助。

这就是今天的全部内容，留一道思考题给你吧，如果你来设计一个整点限量秒杀系统，包括登录、抢购、支付（依赖支付宝）等功能，你会如何设计接口级的故障应对手段？

欢迎你把答案写到留言区，和我一起讨论。相信经过深度思考的回答，也会让你对知识的理解更加深刻。（编辑乱入：精彩的留言有机会获得丰厚福利哦！）

上一篇 30 | 异地多活设计4步走

下一篇 32 | 可扩展架构的基本思想和模式

精选留言 (33)

写留言

空档滑行
2018-07-08

46

1.对于用户服务，在抢购期间可以准备降级策略，压力过大时保证用户登录的可用，注册和修改信息可以做降级处理
2.抢购下单涉及到订单，库存，和商品查询。可通过请求排队来限流，超出库存的请求直接返回。
为了应对库存和商品服务可能发生的故障，可以提前对商品数据和库存数据做缓存，如...
展开

作者回复: 分析全面👍👍

climber
2018-07-08

13

1.抢购页面最大程度静态化，一般用户开始前会尝试刷新页面，查询压力要比下单压力大很多
2.抢购页面要求登陆后访问，一般人不会抢购开始那一刻才进来，错开登陆压力
3.活动未开始，不允许点击抢购按钮。对请求做轻量分析，对于请求过频繁或者可疑ua等做拉黑，为防误杀要求输验证码...

展开

作者回复: 分析的很好，支付如果依赖第三方的话，需要考虑熔断，然后做补偿或者容错措施，例如提示用户10分钟后再来支付

凡凡
2018-07-08

7

1.登录接口在流量特别大的时候，可以适当降级，较少参与人数，另外一点是登录一般有有效期（尤其对于web客户端），可以适当延长登录有效期。2.抢购接口采用队列方式，无法支撑时，也可以适当限流。另外一点，秒杀一般还会有一个秒杀结果查询的接口，也可以降级或者减小请求频次。3.支付接口，一般第三方支付对接入方有流量限制，可以提前申请扩大限制，同时做好降级准备。

展开

作者回复: 分析更好，支付接口依赖第三方应该是熔断，然后做好容错措施，例如提示用户10分钟后来支付

张浩
2018-09-14

6

我发现了,系列文章越到后面,人越少.还好我坚持看到了这里...

作者回复: 😄😄被你发现了，一定要坚持

Tom
2018-07-09

3

功能：登录、抢购、支付。

接口故障应对方法：降级、熔断、限流和排队。

三个功能是有依赖关系的，登录了才能抢购，抢购了才能支付。因此如果从依赖关系考...
展开

作者回复: 假如降级时优先保证登录，但是用户登录进来后发现抢购不了，其实体验也不好，而且已经抢购了的用户可能无法支付，这样体验更不好，甚至会引起投诉，因此抢购类降级是优先降登录会好些，保留抢购和支付，保证进来的用户能够完成业务流程。

支付失败真没什么好办法了，因为这是核心链路的核心功能。

何磊
2018-08-10

2

老师你好，关于排队方式请教个问题：
比如用户从pc发一个请求过来，我们将这个请求放到队列，这个时候就直接返回客户端，客户端定时轮训排队状况，还是说是其它处理方案呢？

作者回复: 常见的有：1. 轮询，2. Long polling，3. HTTP/2推送

qpm
2018-07-07

2

1、在抢购开始前，可以通过降级来保证登陆服务。例如提前半小时限制注册、修改用户信息的服务。
2、整个流量的洪峰应该在于下单。下单可以同时基于限流和排队两种方案，这个需要大概估计到。譬如说100W人抢购100件商品，可以在整点时对下单操作做99%的限流，把1000K的访问问题改写为10k，10k里面再进行排队。...
展开

作者回复: 基本思路就是这样的，另外，登录也可以降级

小狮子辛巴
2018-11-20

1

第一版（不看评论的思考回答）

假如有100种商品，每个商品1k件。首先设计100个队列。
对于已经登录的用户...

展开

作者回复: 认真思考，点赞

今天
2018-10-06

1

感谢老师的专栏，写的很专业，学到了很多，物超所值，尤其是一些方法类的东西，受益终生，以前学习过程中的疑问，也从老师这找到了答案。也期待老师的其他专栏。

作者回复: 没有其它专栏的计划，写个专栏要几年

奋斗心
2018-09-27

1

访问的流量在每个环节可能逐步递减（登陆例外）
1、引导部分用户提前登陆；
2、秒杀价系统独立部署（感觉和其他系统部署在一起才需要降级）；
3、抢购使用排队方式。有界队列，队列大小可以预估较大长度，队列外的拒绝；
4、（如果要求以支付成功为准）通过队列和熔断。...

展开

作者回复: 正确

孙振超
2018-09-09

1

登录大体可分为免密登录和非免密登陆（包含登陆密码、人脸登陆、短信登录等），95%左右的都是免密登录，内部的实现相对也比较简单，采用限流方式就可以了；对于非免密

登录，内部实现相对会复杂些，并且会有风控策略，因而面对大流量需要同时采取限流和内部降级两种策略了。
对于抢购，优先采用排队策略，可以避免限流策略下前面的用户因限流导致抢购失败而...
展开

作者回复: 正确

诗坤
2018-08-15

1

我认为核心需要保证的业务的是抢购，抢购业务可能并发用户较多，对于抢购可以采取排队策略，对参与抢购的用户进行排队。对于登录服务，当发生接口故障的时候，为了保证抢购业务顺利，可以采取降级策略，可以让部分用户登录失败，可以停掉注册等接口，当大量调用支付系统，而支付系统反应慢而影响整个系统的时候，可以对支付系统进行容错处理，可以让用户过段时间再支付。对于熔断的阈值可以根据具体的业务来判断。
展开

作者回复: 分析正确👍👍

食指可爱多
2018-07-26

1

“每一个参加秒杀活动的商品保存一个队列，队列的大小可以根据参与秒杀的商品数量（或加点余量）自行定义。”我之前也思考过类似于淘宝秒杀或12306这种高并发系统，应该不会所以资源在一个队列里排队，这样整个系统并发度无法水平伸缩，那么请问老师一个技术细节问题，每个商品一个队列这个有啥推荐的实现方案吗？
展开

作者回复: 用kafka，用商品id做队列名称就可以了

100kg
2018-07-08

1

老师，我有个双机那节的问题，如果采用了mysql 双主架构，对于库存这样的字段，在并发较高时，如果两个mysql同时 update 了库存（采用乐观锁，库存本身作为版本号），这样会不会导致货物被两个人购买但库存却只减了1呢？该怎么解决呢？求赐教

展开

作者回复: 会的，库存不能用双主

衣申人

2018-07-08

1

整点前降级其他非核心功能，登录采用限流，抢购采用排队加限流，支付对支付宝要有熔断，可加限流。完毕！

展开

作者回复: 没问题就降级，有点过分了呀，产品经理要打你🤓

另外，支付最好也不要限流，支付限流的话，用户体验很不好，还不如前面限流

星火燎原

2018-07-07

1

整点限量抢购核心业务应该是登录和抢购，抢购完了放入购物车不一定马上支付，所以在系统负载较高的时候可以让支付接口做降级处理，过了整点再恢复。抢购接口一般采取商品对应一个抢购队列，队列到上限之后拒绝流量进来，系统根据自身负载情况去消息队列进行流量的拉取，大致思路如上所述，还有什么遗漏还请老师指教

展开

作者回复: 一般不建议对支付做降级，用户体验很不好，还不如登录和抢购阶段限流，这是有心理学理论支撑的，用户没抢到前，如果抢不到他会认为自己运气不好，但如果用户抢到了没法支付，他会觉得自己损失了，会触发“损失厌恶”心理😄

梁中华

2019-05-10

为啥没提供一些工具组建供参考

展开

作者回复: 不同语言不同行业都不同，JAVA推荐spring全家桶

Andy

2019-05-09

请教老师一个技术之外的问题，解决方案架构师和系统架构师的区别是什么？各有什么侧重点吗

作者回复: 解决方案架构师偏业务，系统架构师偏技术，例如来了一个地铁项目，先由解决方案架构师分析需求，业务，然后再交给系统架构师设计方案架构

gkb111

2019-04-15

降级熔断限流排队等方式，来应对接口级故障

展开

vwzf@sina...

2019-03-07

降级：优先保证登录与购买

熔断：图片js脚本等通过静态资源服下载，与功能业务接口分离

限流：用性能测试工具确定系统可承受访问量，超过后拒绝任务执行

排队：业务接收请求用消息队列工具存储

展开