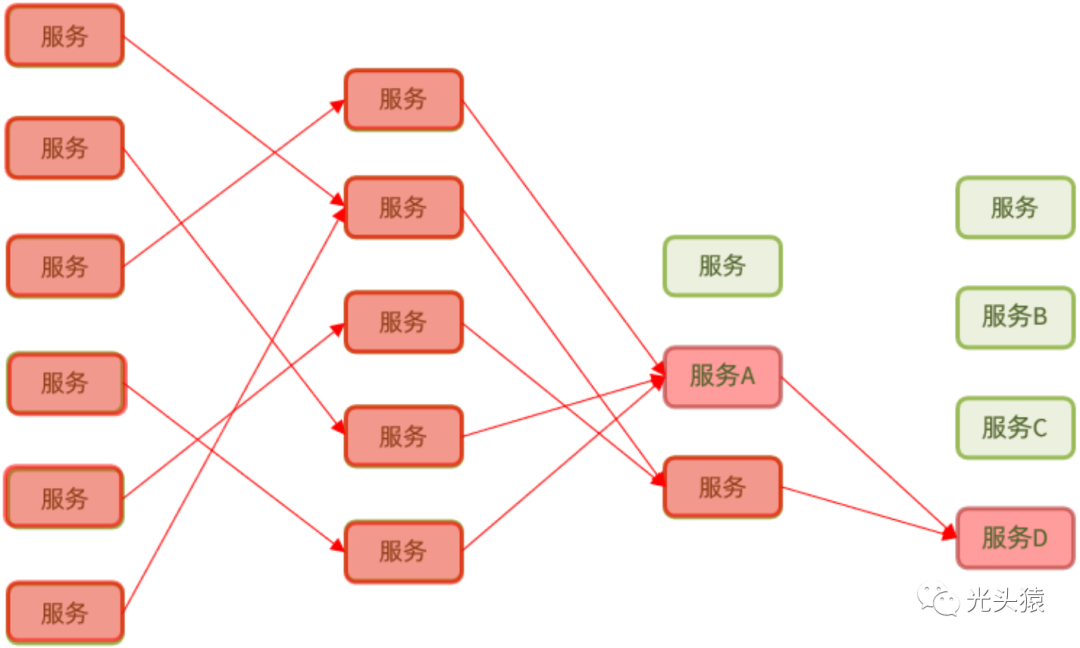
# 1、什么是雪崩效应以及常见的解决方案有哪些？

雪崩效应：一个服务的不可用导致整个系统出现不可用的现象。如下图所示：



常见的解决方案：

1、超时处理：设定超时时间，请求超过一定时间没有响应就返回错误信息，不会无休止等待。

2、线程隔离：给每一个服务分配指定数量的线程，当这个服务使用完这些线程以后，该服务就不能再次被访问，而对其他服务的访问不受影响，将故障控制到了一个小的范围。

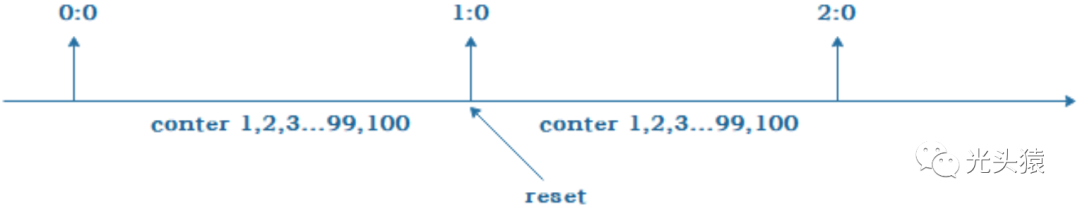
3、熔断降级：由断路器统计业务执行的异常比例，如果超出阈值则会熔断该业务，拦截访问该业务的一切请求，并且可以给失败的调用返回一个降级方案(兜底方案)。

4、流量控制：限制业务访问的QPS，避免服务因流量的突增而故障。

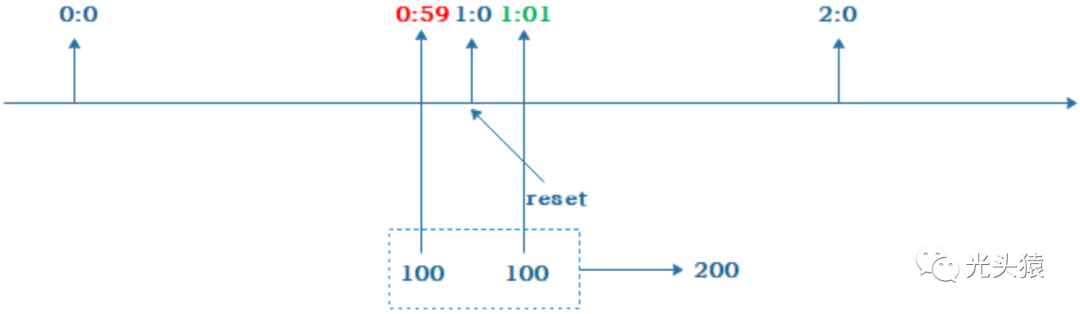
# 2、常见的限流算法都有哪些？

常见的限流算法：计数器、漏桶算法、令牌桶算法

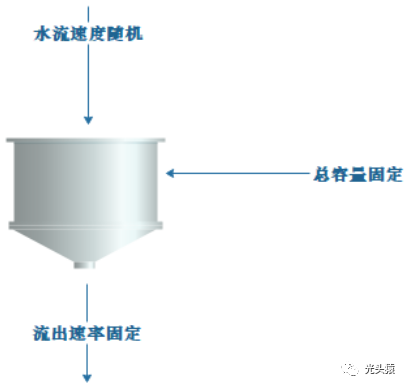
计数器：计数器法是限流算法里最简单也是最容易实现的一种算法。比如我们规定，对于A接口来说，我们1分钟的访问次数不能超过100个。那么我们我们可以设置一个计数器counter，其有效时间为1分钟（即每分钟计数器会被重置为0），每当一个请求过来的时候，counter就加1，如果counter的值大于100，就说明请求数过多；如下图所示：



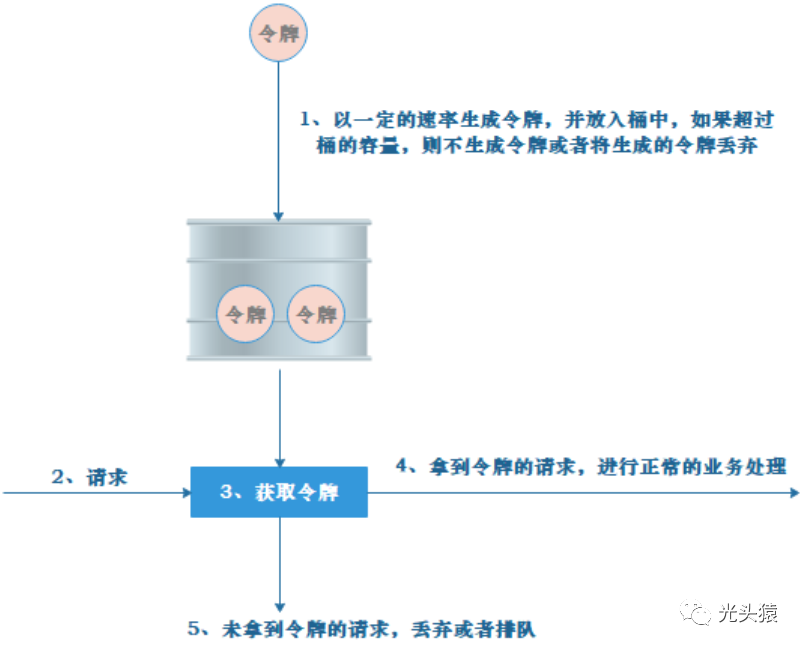
这个算法虽然简单，但是有一个十分致命的问题，那就是临界问题。如下图所示，在1:00前一刻到达100个请求，1:00计数器被重置，1:00后一刻又到达100个请求，显然计数器不会超过100，所有请求都不会被拦截；然而这一时间段内请求数已经达到200，远超100。



漏桶算法：漏桶算法其实很简单，可以粗略的认为就是注水漏水过程，往桶中以一定速率流出水，以任意速率流入水，当水超过桶流量则丢弃，因为桶容量是不变的，保证了整体的速率。如下图所示：



令牌桶算法：令牌桶是一个存放固定容量令牌的桶，按照固定速率r往桶里添加令牌；桶中最多存放b个令牌，当桶满时，新添加的令牌被丢弃；当一个请求达到时，会尝试从桶中获取令牌；如果有，则继续处理请求；如果没有则排队等待或者直接丢弃；可以发现，漏桶算法的流出速率恒定，而令牌桶算法的流出速率却有可能大于r；如下图所示：



从作用上来说，漏桶和令牌桶算法最明显的区别就是是否允许突发流量(burst)的处理，漏桶算法能够强行限制数据的实时传输（处理）速率，对突发流量不做额外处理；而令牌桶算法能够在限制数据的平均传输速率的同时允许某种程度的突发传输。

# 3、nginx中如何实现限流？

nginx的限流主要是两种方式：限制访问频率和限制并发连接数。

nginx按请求速率限速模块使用的是漏桶算法，即能够强行保证请求的实时处理速度不会超过设置的阈值。

nginx官方版本限制IP的连接和并发分别有两个模块：

1、limit\_req\_zone：用来限制单位时间内的请求数，即速率限制 , 采用的漏桶算法 "leaky bucket"。

2、limit\_conn\_zone：用来限制同一时间连接数，即并发限制。

limit\_req\_zone限流配置：

http { *# 定义限流策略,$binary\_remote\_addr对客户端的ip进行限流，zone：定义共享内存区来存储访问信息， rateLimit:10m 表示一个大小为10M，名字为rateLimit的内 # 存区域。1M能存储16000 IP地址的访问信息，10M可以存储16W IP地址访问信息。rate定义了最大访问频率，1s最多允许1个请求访问。* limit\_req\_zone $binary\_remote\_addr zone=rateLimit:10m rate=1r/s ; *# 搜索服务的虚拟主机*    server { location / { *# 使用限流策略，burst=5，重点说明一下这个配置，burst爆发的意思，这个配置的意思是设置一个大小为5的缓冲区（队列）当有大量请求（爆发）过来时，* *# 超过了访问频次限制的请求可以先放到这个缓冲区内。nodelay，如果设置，超过访问频次而且缓冲区也满了的时候就会直接返回503，如果没有设置，则所* *# 有请求会等待排队。* limit\_req zone=rateLimit burst=5 nodelay; proxy\_pass http://train-manager-search ;         }    } }

limit\_conn\_zone限流配置：

http { *# 定义限流策略,$binary\_remote\_addr对客户端的ip进行限流、$server\_name对虚拟主机支持的最大连接数进行限流* limit\_conn\_zone $binary\_remote\_addr zone=perip:10m; limit\_conn\_zone $server\_name zone=perserver:10m; *# 搜索服务的虚拟主机*    server {            location / {             *# 对应的key是 $binary\_remote\_addr，表示限制单个IP同时最多能持有1个连接。*         limit\_conn perip 1;              *# 对应的key是 $server\_name，表示虚拟主机(server) 同时能处理并发连接的总数。* *# 注意，只有当 request header 被后端server处理后，这个连接才进行计数。* limit\_conn perserver 10 ; proxy\_pass http://train-manager-search ;         }    } }

# 4、Sentinel中提供了哪些流控模式分别表示什么意思？

常见的流控模式：

1、直接：统计当前资源的请求，触发阈值时对当前资源直接限流，也是默认的模式

2、关联：统计与当前资源相关的另一个资源，触发阈值时，对当前资源限流。如下流控模式：



表示的意思：当/write资源访问量触发阈值时，就会对/read资源限流，避免影响/write资源。

3、链路：统计从指定链路访问到本资源的请求，触发阈值时，对指定链路限流

例如有两条请求链路：

* /test1 --> /common
* /test2 --> /common

如果只希望统计从/test2进入到/common的请求，则可以这样配置：



# 5、Sentinel中提供了哪些流控效果分别表示什么意思？

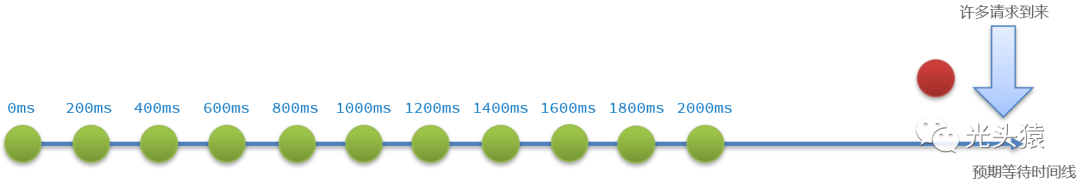
常见的流控效果：

1、快速失败：达到阈值后，新的请求会被立即拒绝并抛出FlowException异常。是默认的处理方式。

2、warm up：预热模式，对超出阈值的请求同样是拒绝并抛出异常。但这种模式阈值会动态变化，从一个较小值逐渐增加到最大阈值。这种模式主要应用于服务的冷启动，请求阈值初始值是 maxThreshold / coldFactor，持续指定时长后，逐渐提高到maxThreshold值。而coldFactor的默认值是3。例如，我设置QPS的maxThreshold为10，预热时间为5秒，那么初始阈值就是 10 / 3 ，也就是3，然后在5秒后逐渐增长到10。如下图所示：



3、排队等待：让所有的请求按照先后次序排队执行，两个请求的间隔不能小于指定时长。后来的请求必须等待前面执行完成，如果请求预期的等待时间超出最大时长，则会被拒绝。



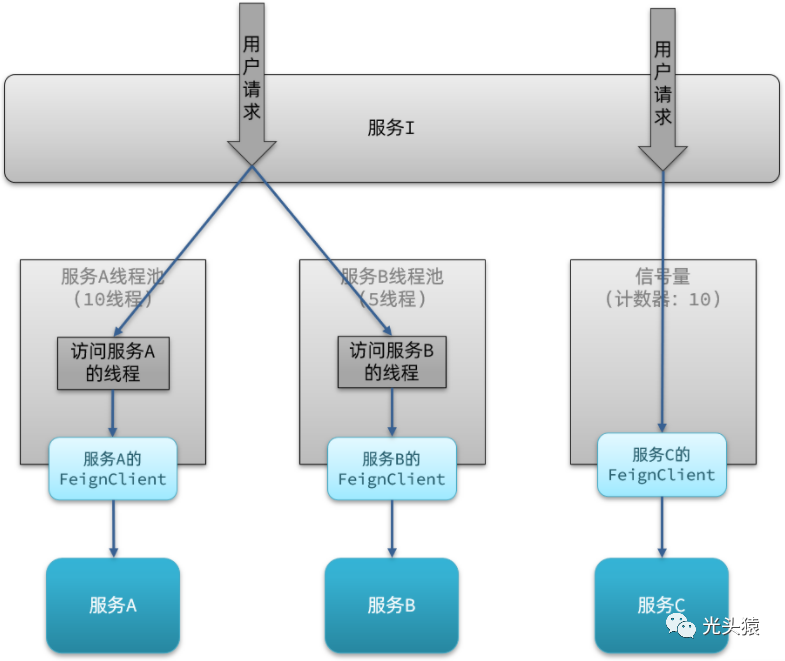
# 6、实现线程隔离有几种方式？Sentinel中使用的是哪一种方式？

线程隔离方式：

1、线程池隔离：有额外开销，但隔离控制更强

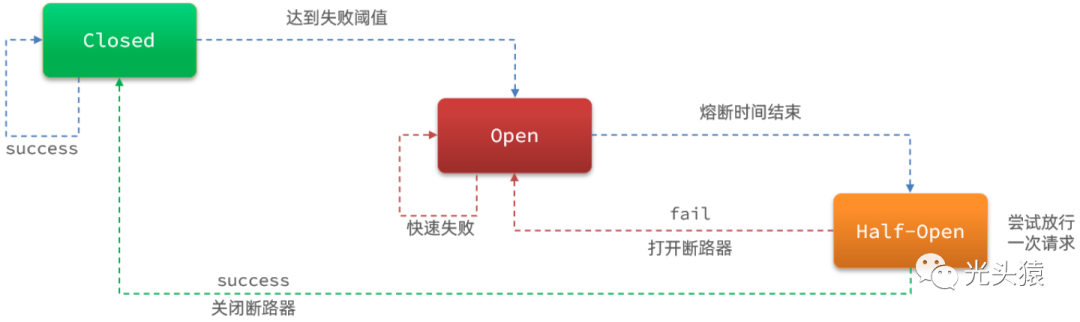
2、信号量隔离：简单，开销小

在Sentinel是通过信号量来实现线程隔离。如下图所示：



# 7、简述Sentinel中熔断器的工作原理？

熔断降级是解决雪崩问题的重要手段。其思路是由断路器统计服务调用的异常比例、慢请求比例，如果超出阈值则会熔断该服务。即拦截访问该服务的一切请求；而当服务恢复时，断路器会放行访问该服务的请求。断路器控制熔断和放行是通过状态机来完成的：



状态机包括三个状态：

1、closed：关闭状态，断路器放行所有请求，并开始统计异常比例、慢请求比例。超过阈值则切换到open状态

2、open：打开状态，服务调用被熔断，访问被熔断服务的请求会被拒绝，快速失败，直接走降级逻辑。Open状态5秒后会进入half-open状态

3、half-open：半开状态，放行一次请求，根据执行结果来判断接下来的操作。

（1）请求成功：则切换到closed状态

（2）请求失败：则切换到open状态