### 1、缓存雪崩

雪崩：

对于系统A，假设每天高峰每秒5000个请求，本来缓存在高峰期可以抗住每秒4000个请求，但是缓存机器以外发生了全盘宕机。缓存挂了，此时一秒5000个请求全部落数据库，数据库必然扛不住，它会报一下警，然后就挂了。此时，如果没有采用什么特别的方案来处理这个故障，DBA很着急，重启数据库，但是数据库立马又被新的流量给打死了。



缓存雪崩的事前事中事后的解决方案如下：

* 事前：redis高可用，主从 + 哨兵，redis cluster，避免全盘崩溃。
* 事中：本地ehcache缓存 + hystrix限流&降级，避免MySQL被打死。
* 事后：redis持久化，一旦重启，自动从磁盘上加载数据，快速恢复缓存数据。



用户发送一个请求，系统A收到请求后，先查本地ehcache缓存，如果没查到再查redis。如果ehcache和redis都没有，再查数据库，将数据库中的结果，写入ehcache和redis中。

限流组件，可以设置每秒的请求，有多少能通过组件，剩余的未通过的请求，怎么办？走降级！可以返回一些默认的值，或者友情提示，或者空白的值。

好处：

* 数据库绝对不会死，限流组件确保了每秒只有多少个请求能通过。
* 只要数据库不死，就是说，对用户来说，2/5 的请求都是可以被处理的。
* 只要有 2/5 的请求可以被处理，就意味着你的系统没死，对用户来说，可能就是点击几次刷不出来页面，但是多点几次，就可以刷出来一次。

### 2、缓存穿透

穿透：

对于系统A，假设一秒 5000 个请求，结果其中 4000 个请求是黑客发出的恶意攻击。黑客发出的那 4000 个攻击，缓存中查不到，每次你去数据库里查，也查不到。举个栗子。数据库id是从1开始的，结果黑客发过来的请求id全部都是负数。这样的话，缓存中不会有，请求每次都“视缓存于无物”，直接查询数据库。这种恶意攻击场景的缓存穿透就会直接把数据库给打死。



解决方案：

每次系统A从数据库中只要没查到，就写一个空值到缓存里去，比如 set -999 UNKNOWN。然后设置一个过期时间，这样的话，下次有相同的 key 来访问的时候，在缓存失效之前，都可以直接从缓存中取数据。

### 3、缓存击穿

击穿：

某个key非常热点，访问非常频繁，处于集中式高并发访问的情况，当这个key在失效的瞬间，大量的请求就击穿了缓存，直接请求数据库，就像是在一道屏障上凿开了一个洞。

不同场景下的解决方案：

* 若缓存的数据是基本不会发生更新的，则可尝试将该热点数据设置为永不过期。
* 若缓存的数据更新不频繁，且缓存刷新的整个流程耗时较少的情况下，则可以采用基于 redis、zookeeper 等分布式中间件的分布式互斥锁，或者本地互斥锁以保证仅少量的请求能请求数据库并重新构建缓存，其余线程则在锁释放后能访问到新缓存。
* 若缓存的数据更新频繁或者缓存刷新的流程耗时较长的情况下，可以利用定时线程在缓存过期前主动的重新构建缓存或者延后缓存的过期时间，以保证所有的请求能一直访问到对应的缓存。