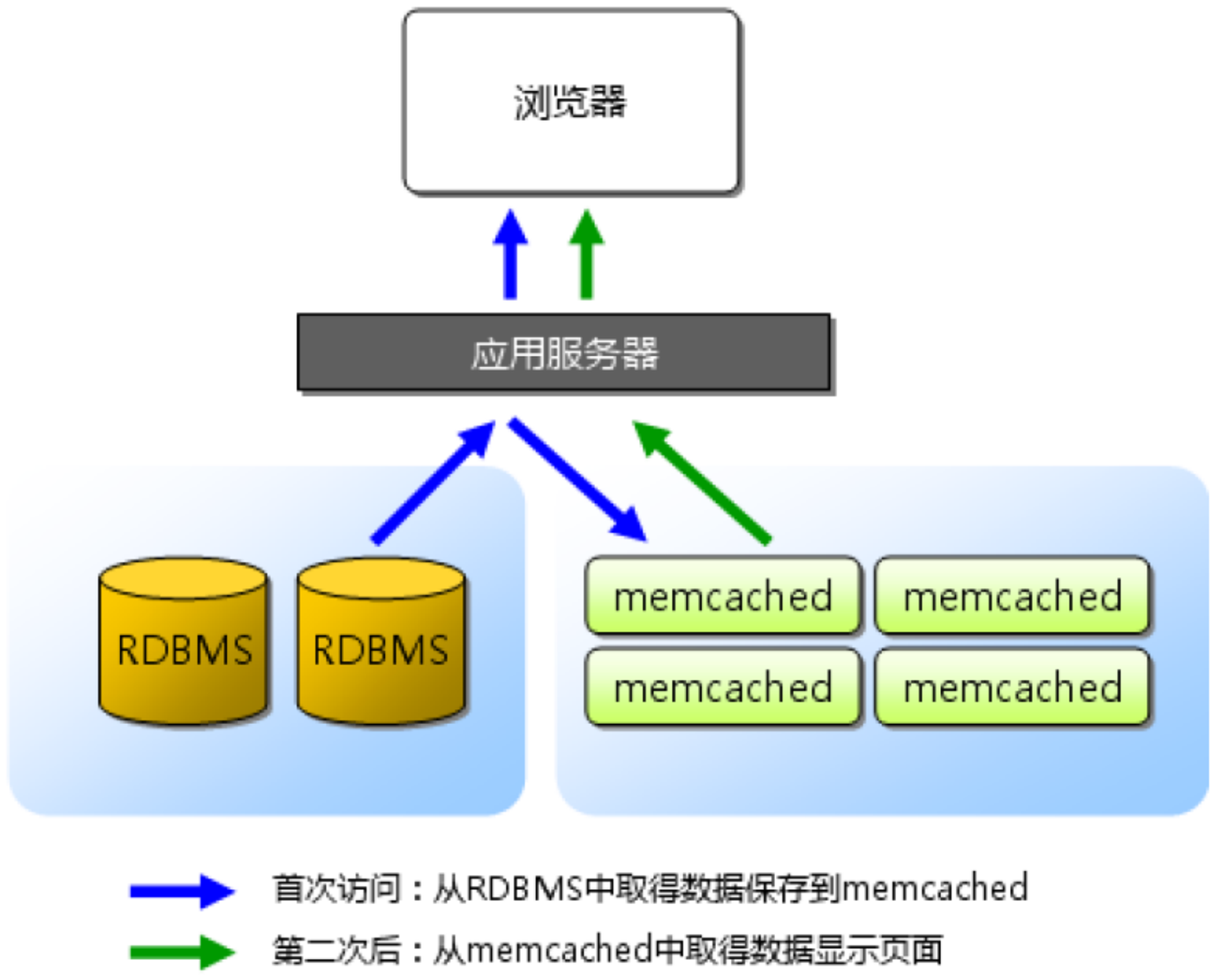
一. 分布式缓存

分布式缓存是网站服务端经常用到的一种技术，在读多写少的业务场景中，通过使用缓存可以有效地支撑高并发的访问量，对后端的数据库等数据源做到很好地保护。现在市面上有很多分布式缓存，比如Redis、Memcached以及阿里的Tair等，不管我们使用的哪种缓存产品，基本上都会遇到缓存击穿、缓存失效以及热点key的问题。

通常我们在使用缓存时候都是先检查缓存中是否存在，如果存在直接返回缓存内容，如果不存在就直接查询数据库然后再缓存查询结果返回，例如下图所示，



二. 缓存击穿

查询一个数据库中不存在的数据，比如商品详情，查询一个不存在的ID，每次都会访问DB，如果有人恶意破坏，很可能直接对DB造成过大地压力。

解决方案：

1. 当通过某一个key去查询数据的时候，如果对应在数据库中的数据都不存在，我们将此key对应的value设置为一个默认的值，比如“NULL”，并设置一个缓存的失效时间，这时在缓存失效之前，所有通过此key的访问都被缓存挡住了。后面如果此key对应的数据在DB中存在时，缓存失效之后，通过此key再去访问数据，就能拿到新的value了。
2. 设置一个布隆过滤器确认一个key是否有缓存

三. 缓存失效

在高并发的环境下，如果此时key对应的缓存失效，此时有多个进程就会去同时去查询DB，然后再去同时设置缓存。这个时候如果这个key是系统中的热点key或者同时失效的数量比较多时，DB访问量会瞬间增大，造成过大的压力。

解决方案

1. 将系统中key的缓存失效时间均匀地错开，防止同一时间点有大量的key对应的缓存失效
2. 重新设计缓存的使用方式，当我们通过key去查询数据时，首先查询缓存，如果此时缓存中查询不到，就通过分布式锁进行加锁，取得锁的进程查DB并设置缓存，然后解锁；其他进程如果发现有锁就等待，然后等解锁后返回缓存数据或者再次查询DB。

四. 热点key

缓存中的某些Key(可能对应用与某个促销商品)对应的value存储在集群中一台机器，使得所有流量涌向同一机器，成为系统的瓶颈，该问题的挑战在于它无法通过增加机器容量来解决。

解决方案：

1. 客户端热点key缓存：将热点key对应value并缓存在客户端本地，并且设置一个失效时间。对于每次读请求，将首先检查key是否存在于本地缓存中，如果存在则直接返回，如果不存在再去访问分布式缓存的机器。
2. 将热点key分散为多个子key，然后存储到缓存集群的不同机器上，这些子key对应的value都和热点key是一样的。当通过热点key去查询数据时，通过某种hash算法随机选择一个子key，然后再去访问缓存机器，将热点分散到了多个子key上。