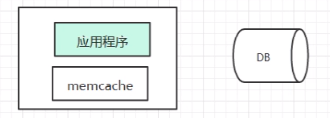
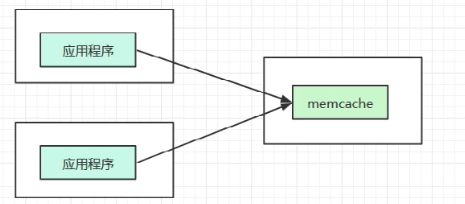
# 背景

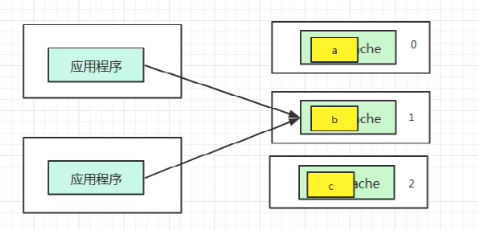
对于数据量不是很大的情况下，采用本地缓存即可解决快速访问数据库的问题：



对于数据量很大的情况下，需要将应用程序和缓存分别置于不同机器，即采用远程分布式缓存（Redis单机并发10W+）：



对于大型高并发系统（高并发，海量数据），针对客户端高并发Redis缓存无法满足需要，需要做缓存集群，针对海量数据，缓存数据会超过单台机器的内存限制，所以需要将数据分割，采用数据分布式存储：



应用程序要想读取不同节点上的缓存数据，可以采用如下方法：

1. 哈希求余：hash(key) % 缓存集群节点数
2. 一致性hash
3. 一致性hash+虚拟节点

# 原理

传统哈希算法有这样的一个缺陷，如果增加服务器数量，则原来哈希计算出来的存储节点会失效，即缓存失效，造成雪崩。

一致性哈希算法将经过哈希计算后的结果映射到一个环上，然后顺时针查找，下一个节点就是需要存放数据的节点。缓存机器数量发生变化的时候，一致性哈希算法可以使得大部分缓存数据可以被访问，即不是所有缓存数据失效，只有少部分缓存失效。

一致性hash算法的步骤：

1. hash值是个整数非负，非负整数的值范围做成一个圆环；
2. 对集群的节点的某个属性（比如节点名）求hash值，放到环上；
3. 对数据key求hash值，也放到环上，按照顺时针方向找到离它最近的节点，放在它上面。

但是一致性哈希算法存在hash偏斜的问题，即缓存不均匀，导致系统崩溃。针对这种问题，可以增加服务器，使得尽可能均匀分布，为此引入虚拟节点（缓存读写🡪虚拟节点🡪真实节点🡪读写）。

# 应用

应用一致性哈希算法：缓存（Redis/memcache），Hadoop，搜索引擎（ES），分布式数据库。即数据分布式存储的场景，需要考虑使用一致性哈希算法。

分布式数据库领域的应用：

1. 水平拓展：

https://blog.csdn.net/linuxheik/article/details/51025879

https://blog.csdn.net/ydyang1126/article/details/70313981

1. 其他

https://yq.aliyun.com/articles/57954

https://www.cnblogs.com/lonecloud/p/8093328.html