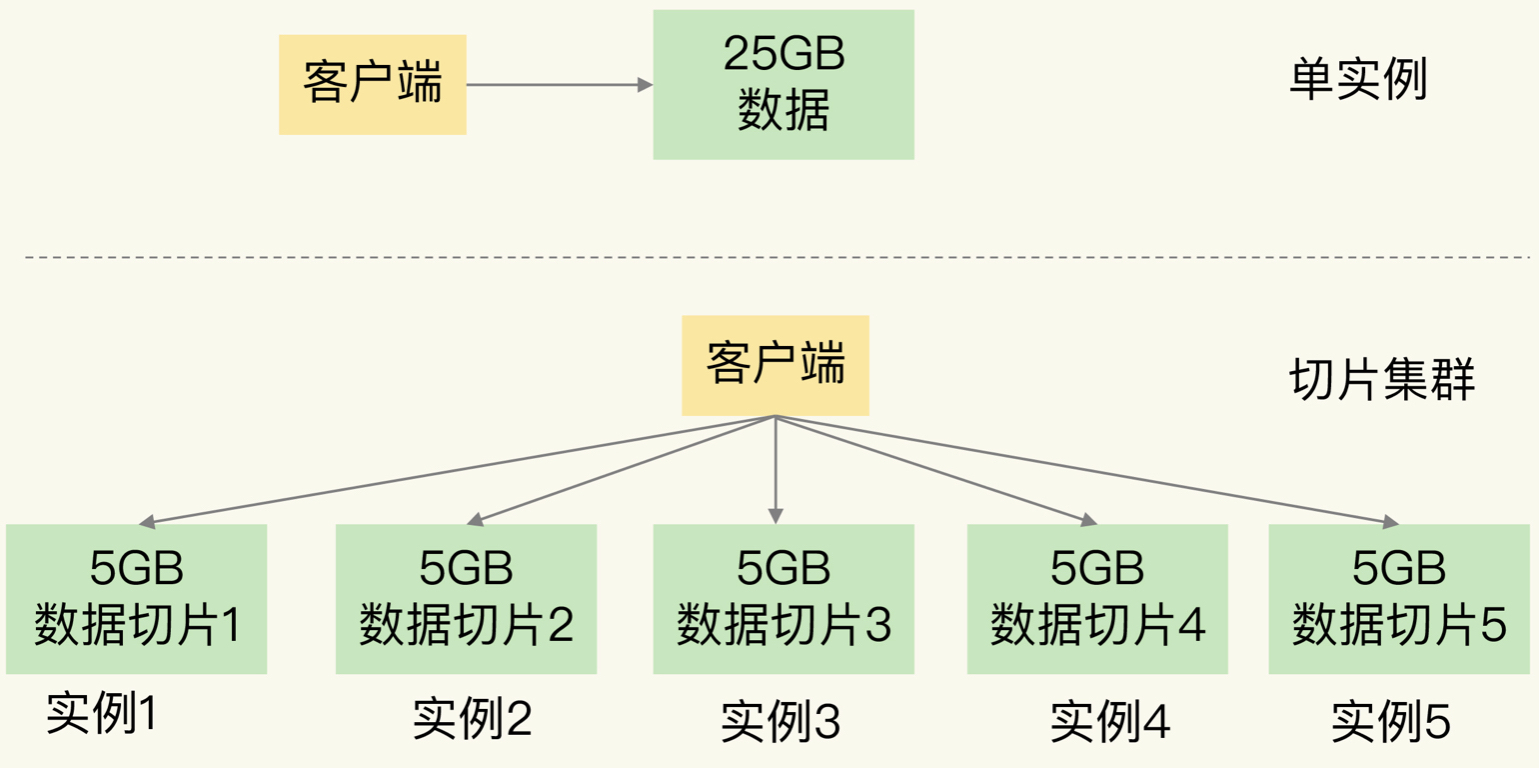
分片集群

切片集群，也叫分片集群，就是指启动多个Redis实例组成一个集群，然后按照一定的规则，把收到的数据划分成多份，每一份用一个实例来保存。



在切片集群中，实例在为5GB数据生成RDB时，数据量就小了很多，fork子进程一般不会给主线程带来较长时间的阻塞。采用多个实例保存数据切片后，我们既能保存25GB数据，又避免了fork子进程阻塞主线程而导致的响应突然变慢。

一、如何保存更多数据？

为了保存大量数据，使用大内存云主机和切片集群两种方法。实际上，这两种方法分别对应着Redis应对数据量增多的两种方案：纵向扩展（scale up）和横向扩展（scale out）。

- [ ] 纵向扩展：升级单个Redis实例的资源配置，包括增加内存容量、增加磁盘容量、使用更高配置的CPU。就像下图中，原来的实例内存是8GB，硬盘是50GB，纵向扩展后，内存增加到24GB，磁盘增加到150GB。

- [ ] 横向扩展：横向增加当前Redis实例的个数，就像下图中，原来使用1个8GB内存、50GB磁盘的实例，现在使用三个相同配置的实例。

1.1 纵向扩展

好处：实施起来简单、直接。

坏处：

当使用RDB对数据进行持久化时，如果数据量增加，需要的内存也会增加，主线程fork子进程时就可能会阻塞。不过，如果你不要求持久化保存Redis数据，那么，纵向扩展会是一个不错的选择。

纵向扩展会受到硬件和成本的限制。

1.2 横向扩展

好处：

只用增加Redis的实例个数就行了，不用担心单个实例的硬件和成本限制。

在面向百万、千万级别的用户规模时，横向扩展的Redis切片集群会是一个非常好的选择。

坏处：不可避免地涉及到多个实例的分布式管理问题。

• 数据切片后，在多个实例之间如何分布？

• 客户端怎么确定想要访问的数据在哪个实例上？

二、数据切片和实例的对应分布关系

在Redis 3.0之前，官方并没有针对切片集群提供具体的方案。从3.0开始，官方提供了一个名为Redis Cluster的方案，用于实现切片集群。Redis Cluster方案中就规定了数据和实例的对应规则。

具体来说，Redis Cluster方案采用哈希槽（Hash Slot，接下来我会直接称之为Slot），来处理数据和实例之间的映射关系。在Redis Cluster方案中，一个切片集群共有16384个哈希槽，这些哈希槽类似于数据分区，每个键值对都会根据它的key，被映射到一个哈希槽中。

- [ ] 首先根据键值对的key，按照CRC16算法计算一个16 bit的值；然后，再用这个16bit值对16384取模，得到0~16383范围内的模数，每个模数代表一个相应编号的哈希槽。

- [ ] 在部署Redis Cluster方案时，可以使用cluster create命令创建集群，此时，Redis会自动把这些槽平均分布在集群实例上。例如，如果集群中有N个实例，那么，每个实例上的槽个数为16384/N个。

注：在手动分配哈希槽时，需要把16384个槽都分配完，否则Redis集群无法正常工作。

三、客户端定位数据

Redis实例会把自己的哈希槽信息发给和它相连接的其它实例，来完成哈希槽分配信息的扩散。当实例之间相互连接后，每个实例就有所有哈希槽的映射关系了。

客户端收到哈希槽信息后，会把哈希槽信息缓存在本地。当客户端请求键值对时，会先计算键所对应的哈希槽，然后就可以给相应的实例发送请求了。但是，在集群中，实例和哈希槽的对应关系并不是一成不变的，最常见的变化有两个：

- [ ] 在集群中，实例有新增或删除，Redis需要重新分配哈希槽；

- [ ] 为了负载均衡，Redis需要把哈希槽在所有实例上重新分布一遍。

3.1 重定向机制

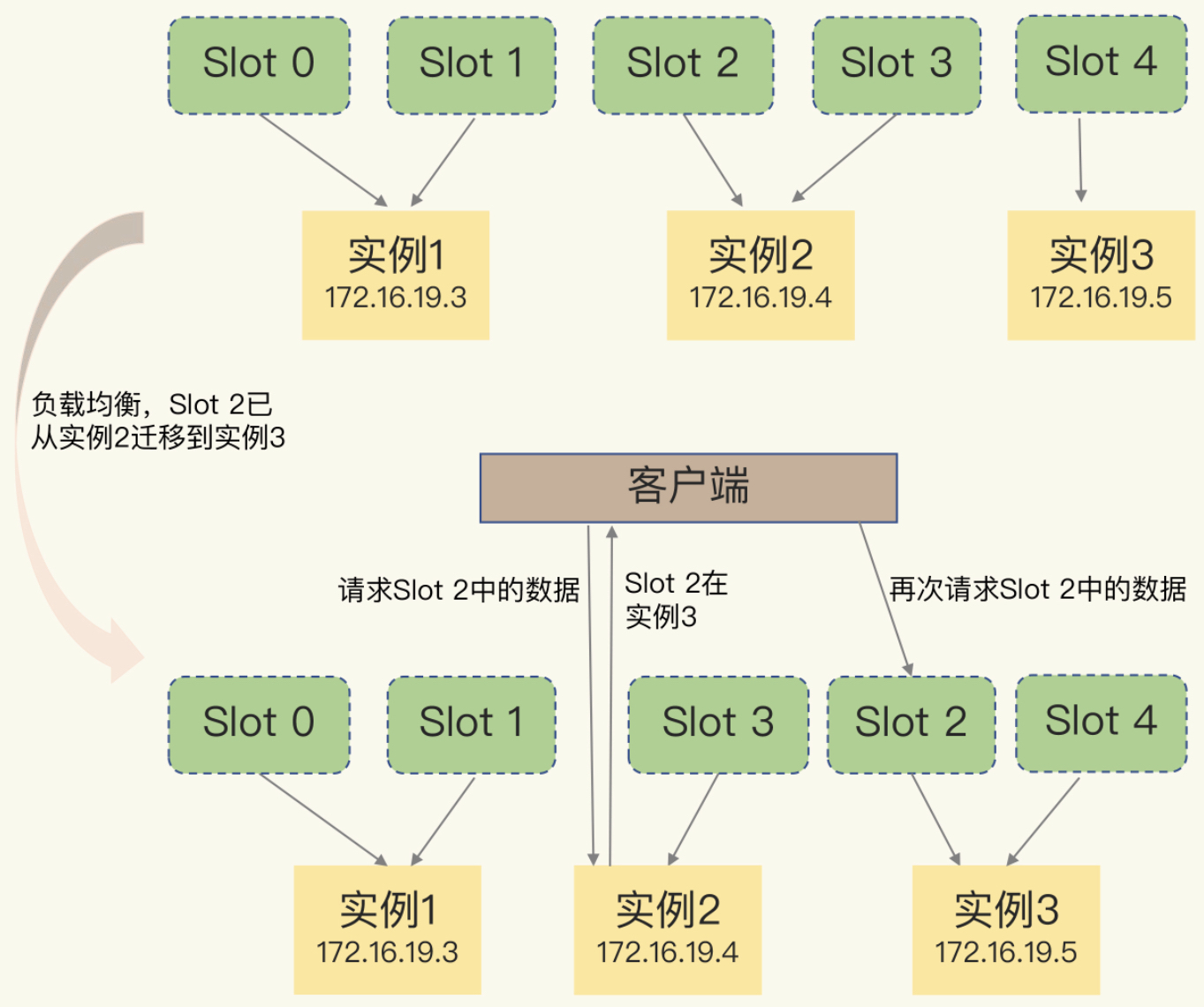
所谓的“重定向”，是指客户端给一个实例发送数据读写操作时，这个实例上并没有相应的数据，客户端要再给一个新实例发送操作命令。

当客户端把一个键值对的操作请求发给一个实例时，如果这个实例上并没有这个键值对映射的哈希槽，那么，这个实例就会给客户端返回下面的MOVED命令响应结果，这个结果中就包含了新实例的访问地址。

GET hello:key

(error) MOVED 13320 172.16.19.5:6379

其中，MOVED命令表示，客户端请求的键值对所在的哈希槽13320，实际是在172.16.19.5这个实例上。通过返回的MOVED命令，就相当于把哈希槽所在的新实例的信息告诉给客户端了。这样一来，客户端就可以直接和172.16.19.5连接，并发送操作请求了。

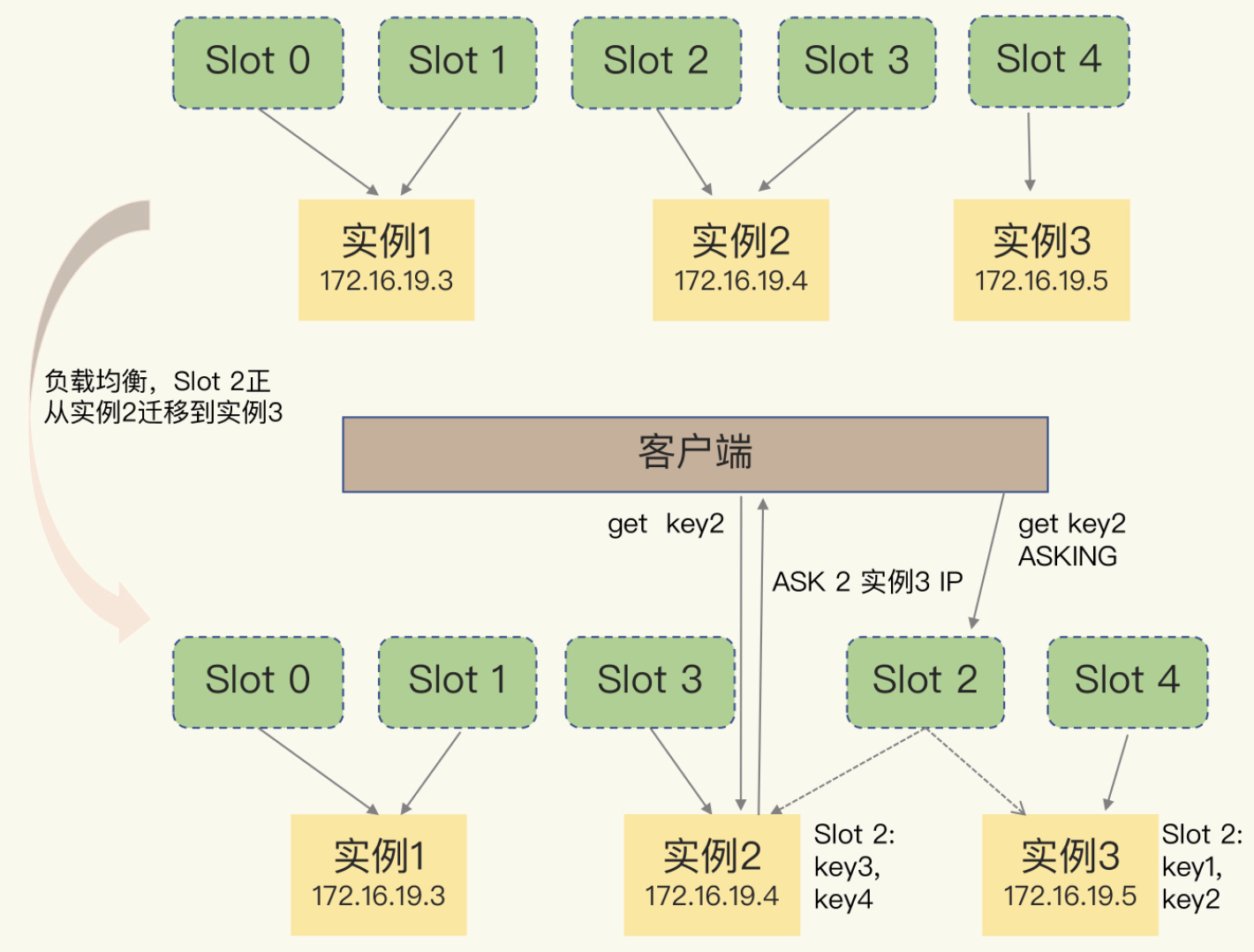


若客户端向实例2发送请求，但此时Slot 2中的数据只有一部分迁移到了实例3，还有部分数据没有迁移。在这种迁移部分完成的情况下，客户端就会收到一条ASK报错信息，如下所示：

GET hello:key

(error) ASK 13320 172.16.19.5:6379

这个结果中的ASK命令就表示，客户端请求的键值对所在的哈希槽13320，在172.16.19.5这个实例上，但是这个哈希槽正在迁移。此时，客户端需要先给172.16.19.5这个实例发送一个ASKING命令。这个命令的意思是，让这个实例允许执行客户端接下来发送的命令。然后，客户端再向这个实例发送GET命令，以读取数据。



和MOVED命令不同，ASK命令并不会更新客户端缓存的哈希槽分配信息。所以，在上图中，如果客户端再次请求Slot 2中的数据，它还是会给实例2发送请求。这也就是说，ASK命令的作用只是让客户端能给新实例发送一次请求，而不像MOVED命令那样，会更改本地缓存，让后续所有命令都发往新实例。