16-用Redis构建缓存集群的最佳实践有哪些？

你好，我是李玥。

之前连续几节课，我们都在以MySQL为例子，讲如何应对海量数据，如何应对高并发，如何实现高可用，我先带你简单复习一下。

- 数据量太大查询慢怎么办？存档历史数据或者分库分表，这是数据分片。
- 并发太高扛不住怎么办？读写分离，这是增加实例数。
- 数据库宕机怎么办？增加从节点，主节点宕机的时候用从节点顶上，这是主从复制。但是这里面要特别注意数据一致性的问题。

我在之前课程中，也多次提到过，**这些方法不仅仅是MySQL特有的，对于几乎所有的存储系统，都是适用的。**

今天这节课，我们来聊一聊，如何构建一个生产系统可用的Redis缓存集群。你将会看到，几种集群解决方案，它们用到的思想，基本和我们上面讲的都是一样的。

Redis Cluster如何解决数据量大、高可用和高并发问题？

Redis从3.0版本开始，提供了官方的集群支持，也就是Redis Cluser。**Redis Cluster相比于单个节点的Redis，能保存更多的数据，支持更多的并发，并且可以做到高可用，在单个节点故障的情况下，继续提供服务。**

为了能够保存更多的数据，和MySQL分库分表的方式类似，Redis Cluster也是通过分片的方式，把数据分布到集群的多个节点上。

Redis Cluster是如何来分片的呢？它引入了一个“槽（Slot）”的概念，这个槽就是哈希表中的哈希槽，槽是Redis分片的基本单位，每个槽里面包含一些Key。每个集群的槽数是固定的16384（16 * 1024）个，每个Key落在哪个槽中也是固定的，计算方法是：

```
HASH_SLOT = CRC16(key) mod 16384
```

这个算法很简单，先计算Key的CRC值，然后把这个CRC之后的Key值直接除以16384，余数就是Key所在的槽。这个算法就是我们上节课讲过的哈希分片算法。

这些槽又是如何存放到具体的Redis节点上的呢？这个映射关系保存在集群的每个Redis节点上，集群初始化的时候，Redis会自动平均分配这16384个槽，也可以通过命令来调整。这个分槽的方法，也是我们上节课讲到过的分片算法：查表法。

客户端可以连接集群的任意一个节点来访问集群的数据，当客户端请求一个Key的时候，被请求的那个Redis实例先通过上面的公式，计算出这个Key在哪个槽中，然后再查询槽和节点的映射关系，找到数据所在的真正节点，如果这个节点正好是自己，那就直接执行命令返回结果。如果数据不在当前这个节点上，那就给客户端返回一个重定向的命令，告诉客户端，应该去连哪个节点上请求这个Key的数据。然后客户端会

再连接正确的节点来访问。

解决分片问题之后，Redis Cluster就可以通过水平扩容来增加集群的存储容量，但是，每次往集群增加节点的时候，需要从集群的那些老节点中，搬运一些槽到新节点，你可以手动指定哪些槽迁移到新节点上，也可以利用官方提供的redis-trib.rb脚本来自动重新分配槽，自动迁移。

分片可以解决Redis保存海量数据的问题，并且客观上提升了Redis的并发能力和查询性能。但是并不能解决高可用的问题，每个节点都保存了整个集群数据的一个子集，任何一个节点宕机，都会导致这个宕机节点上的那部分数据无法访问。

那Redis Cluster是怎么解决高可用问题的?

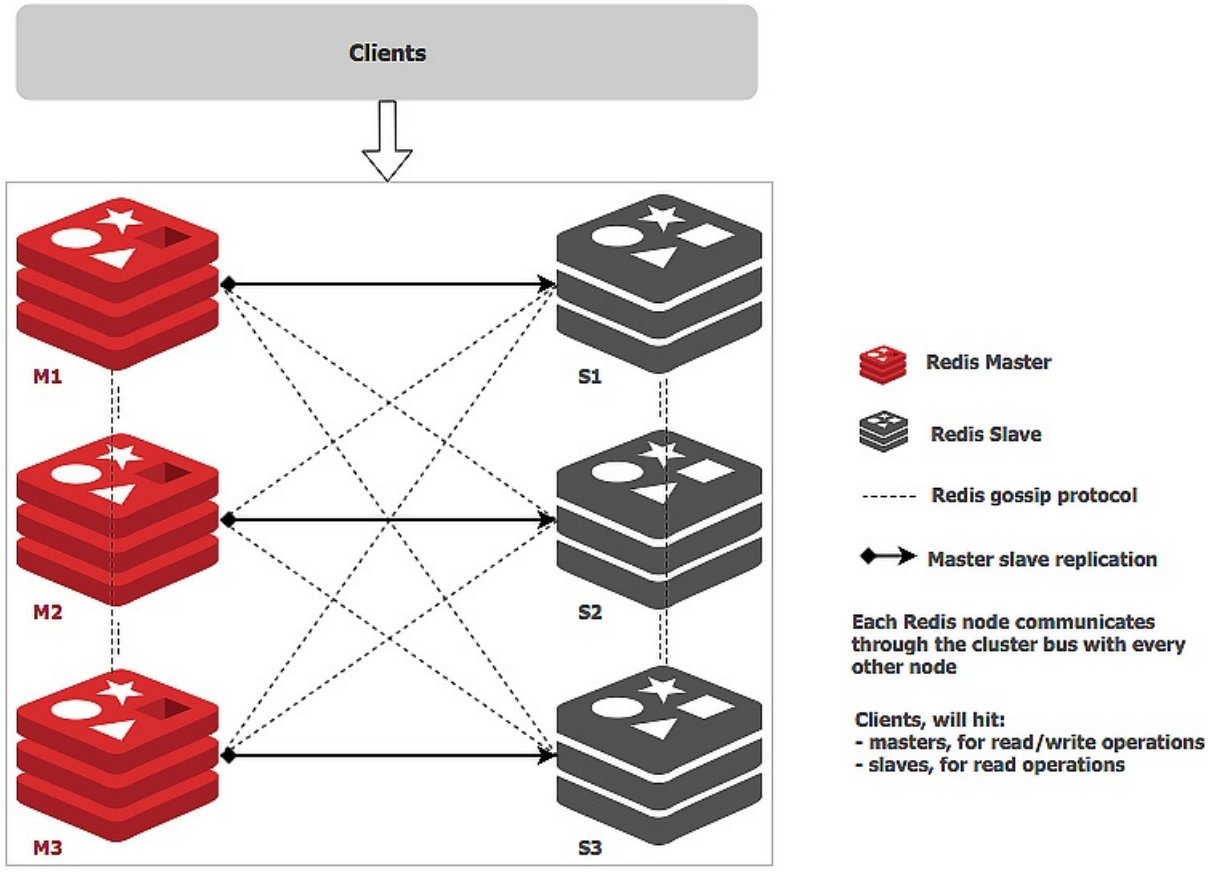
参见上面我们讲到的方法：**增加从节点，做主从复制**。Redis Cluster支持给每个分片增加一个或多个从节点，每个从节点在连接到主节点上之后，会先给主节点发送一个SYNC命令，请求一次全量复制，也就是把主节点上全部的数据都复制到从节点上。全量复制完成之后，进入同步阶段，主节点会把刚刚全量复制期间收到的命令，以及后续收到的命令持续地转发给从节点。

因为Redis不支持事务，所以它的复制相比MySQL更简单，连Binlog都省了，直接就是转发客户端发来的更新数据命令来实现主从同步。如果某个分片的主节点宕机了，集群中的其他节点会在这个分片的从节点中选出一个新的节点作为主节点继续提供服务。新的主节点选举出来后，集群中的所有节点都会感知到，这样，如果客户端的请求Key落在故障分片上，就会被重定向到新的主节点上。

最后我们看一下，Redis Cluster是如何应对高并发的。

一般来说，Redis Cluster进行了分片之后，每个分片都会承接一部分并发的请求，加上Redis本身单节点的性能就非常高，所以大部分情况下不需要再像MySQL那样做读写分离来解决高并发的问题。默认情况下，集群的读写请求都是由主节点负责的，从节点只是起一个热备的作用。当然了，Redis Cluster也支持读写分离，在从节点上读取数据。

以上就是Redis Cluster的基本原理，你可以对照下图来加深理解。



（图片来源：网络）

你可以看到，Redis Cluster整体的架构完全就是照抄MySQL构建集群的那一套东西（当然，这些设计和方法也不是MySQL发明的），抄作业抄的就差把名字一起也抄上了。

具体如何搭建Redis Cluster以及相关的操作命令你可以看一下Redis官方的这篇教程。

为什么Redis Cluster不适合超大规模集群？

Redis Cluster的优点是易于使用。分片、主从复制、弹性扩容这些功能都可以做到自动化，通过简单的部署就可以获得一个大容量、高可靠、高可用的Redis集群，并且对于应用来说，近乎于是透明的。

所以，**Redis Cluster是非常适合构建中小规模Redis集群**，这里的中小规模指的是，大概几个到几十个节点这样规模的Redis集群。

但是Redis Cluster不太适合构建超大规模集群，主要原因是，它采用了去中心化的设计。刚刚我们讲了，Redis的每个节点上，都保存了所有槽和节点的映射关系表，客户端可以访问任意一个节点，再通过重定向命令，找到数据所在的那个节点。那你有没有想过一个问题，这个映射关系表，它是如何更新的呢？比如说，集群加入了新节点，或者某个主节点宕机了，新的主节点被选举出来，这些情况下，都需要更新集群每一个节点上的映射关系表。

Redis Cluster采用了一种去中心化的流言(Gossip)协议来传播集群配置的变化。一般涉及到协议都比较复杂，这里我们不去深究具体协议和实现算法，我大概给你讲一下这个协议原理。

所谓流言，就是八卦，比如说，我们上学的时候，班上谁和谁偷偷好上了，搞对象，那用不了一天，全班同学都知道了。咋知道的？张三看见了，告诉李四，李四和王小二特别好，又告诉了王小二，这样人传人，不久就传遍全班了。这个就是八卦协议的传播原理。

这个八卦协议它的好处是去中心化，传八卦不需要组织，吃瓜群众自发就传开了。这样部署和维护就更简单，也能避免中心节点的单点故障。八卦协议的缺点就是传播速度慢，并且是集群规模越大，传播的越慢。这个也很好理解，比如说，换成某两个特别出名的明星搞对象，即使是全国人民都很八卦，但要想让全国每一个人都知道这个消息，还是需要很长的时间。在集群规模太大的情况下，数据不同步的问题会被明显放大，还有一定的不确定性，如果出现问题很难排查。

如何用Redis构建超大规模集群？

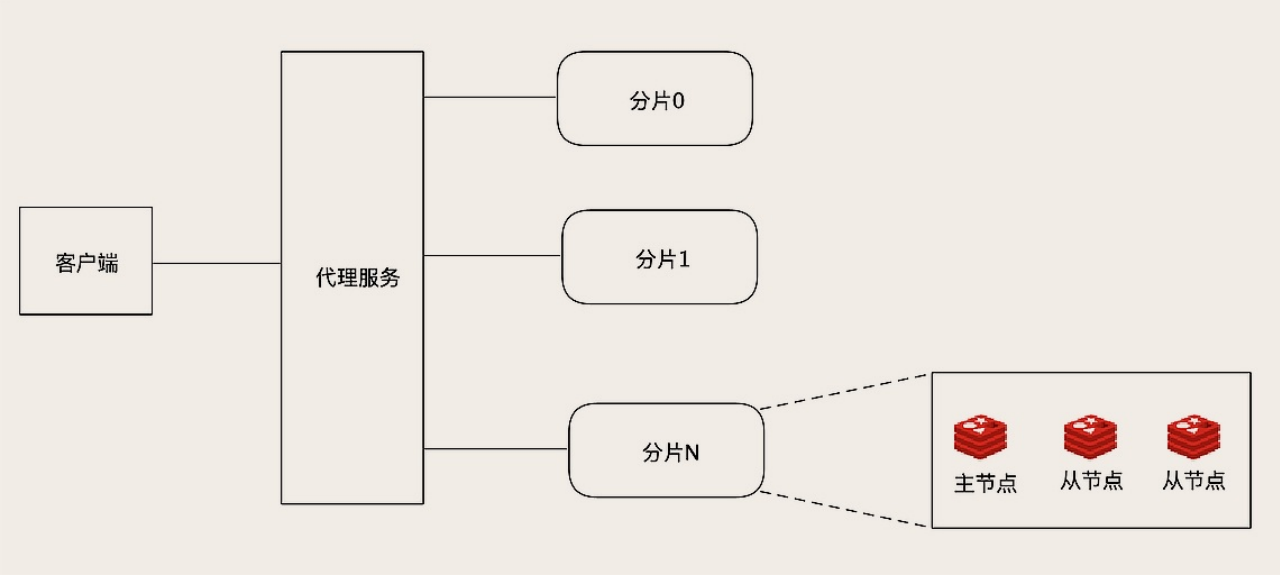
Redis Cluster不太适合用于大规模集群，所以很多大厂，都选择自己去搭建Redis集群。这里面，每一家的解决方案都有自己的特色，但其实总体的架构都是大同小异的。

一种是基于代理的方式，在客户端和Redis节点之间，还需要增加一层代理服务。这个代理服务有三个作用。

第一个作用是，负责在客户端和Redis节点之间转发请求和响应。客户端只和代理服务打交道，代理收到客户端的请求之后，再转发到对应的Redis节点上，节点返回的响应再经由代理转发返回给客户端。

第二个作用是，负责监控集群中所有Redis节点状态，如果发现有问题节点，及时进行主从切换。

第三个作用就是维护集群的元数据，这个元数据主要就是集群所有节点的主从信息，以及槽和节点关系映射表。这个架构和我在《12 | MySQL如何应对高并发（二）：读写分离》这节课中给你讲过的，用HAProxy+Keepalived来代理MySQL请求的架构是类似的，只是多了一个自动分片路由的功能而已。



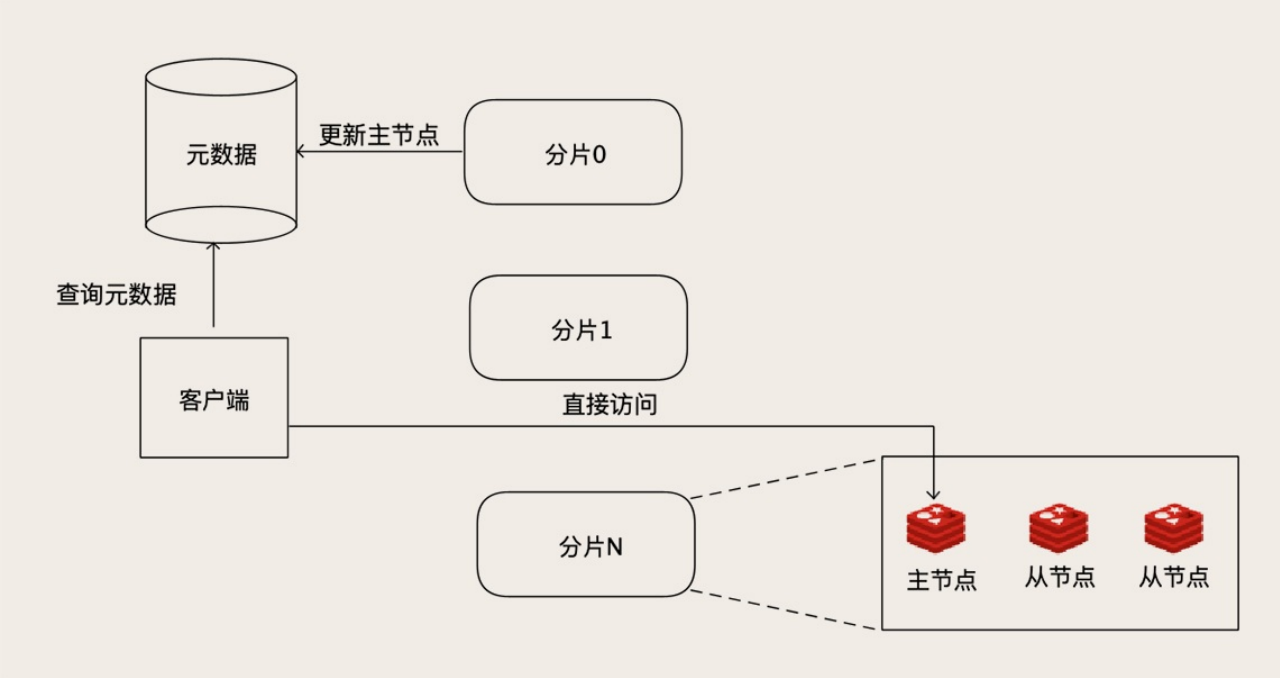
像开源的Redis集群方案twemproxy和Codis，都是这种架构的。

这个架构最大的优点是对客户端透明，在客户端视角来看，整个集群和一个超大容量的单节点Redis是一样的。并且，由于分片算法是代理服务控制的，扩容也比较方便，新节点加入集群后，直接修改代理服务中的元数据就可以完成扩容。

不过，这个架构的缺点也很突出，增加了一层代理转发，每次数据访问的链路更长了，必然会带来一定的性能损失。而且，代理服务本身又是集群的一个单点，当然，我们可以把代理服务也做成一个集群来解决单点问题，那样集群就更复杂了。

另外一种方式是，不用这个代理服务，把代理服务的寻址功能前移到客户端中去。客户端在发起请求之前，先去查询元数据，就可以知道要访问的是哪个分片和哪个节点，然后直连对应的Redis节点访问数据。

当然，客户端不用每次都去查询元数据，因为这个元数据是不怎么变化的，客户端可以自己缓存元数据，这样访问性能基本上和单机版的Redis是一样的。如果某个分片的主节点宕机了，新的主节点被选举出来之后，更新元数据里面的信息。对集群的扩容操作也比较简单，除了迁移数据的工作必须要做以外，更新一下元数据就可以了。



虽然说，这个元数据服务仍然是一个单点，但是它的数据量不大，访问量也不大，相对就比较容易实现。我们可以用ZooKeeper、etcd甚至MySQL都能满足要求。这个方案应该是最适合超大规模Redis集群的方案了，在性能、弹性、高可用几方面表现都非常好，缺点是整个架构比较复杂，客户端不能通用，需要开发定制化的Redis客户端，只有规模足够大的企业才负担得起。

小结

今天这节课我们讲了从小到大三种构建Redis集群的方式。

- 小规模的集群建议使用官方的Redis Cluster，在节点数量不多的情况下，各方面表现都不错。
- 再大一些规模的集群，可以考虑使用twemproxy或者Codis这类的基于代理的集群架构，虽然是开源方案，但是已经被很多公司在生产环境中验证过。
- 相比于代理方案，使用定制客户端的方案性能更好，很多大厂采用的都是类似的架构。

还有一个小问题需要注意的是，这几种集群方案对一些类似于“KEYS”这类的多KEY命令，都没法做到百分百支持。原因很简单，数据被分片了之后，这种多KEY的命令很可能需要跨多个分片查询。当你的系统从单个Redis库升级到集群时，可能需要考虑一下这方面的兼容性问题。

思考题

很多存储系统之间都存在“互相抄作业”的嫌疑，其实这对于我们这些存储系统的使用者来说是好事儿，比如我们把MySQL都学透了，你再去看Redis，知道它抄了哪些作业，这部分我们就可以迅速掌握了，只要再研究一下不一样的那一小部分内容，我们就可以精通Redis了是不？

课后请你再去看一下HDFS，它在解决分片、复制和高可用这几方面，哪些是“抄作业”，哪些又是自己独创的。欢迎你在留言区与我讨论。

感谢你的阅读，如果你觉得今天的内容对你有帮助，也欢迎把它分享给你的朋友。

精选留言：

- 李玥 2020-04-03 09:58:31
  Hi，我是李玥。

  这里回顾一下上节课的思考题：

  课后请你想一下，把订单表拆分之后，那些和订单有外键关联的表，该怎么处理？

  对于这些表，我的建议是，和订单表一起拆分，让相同订单ID的订单和关联表的数据分布到相同的分片上，这样便于查询。

- 发条橙子 。 2020-04-03 09:21:34
  老师 有两个疑问点

  1. 大厂自建集群 是不是就是常说的使用一致性哈希来做槽的映射

  2. 既然自带集群包含了哨兵， 代理也包含 ，是不是哨兵的功能也是在自建中使用 ，还是大厂自建的也不用哨兵 是自己做的高可用

- 正在减肥的胖籽。 2020-04-02 11:24:18
  请教老师一个问题：
  1.redis集群新增分片后，线上怎么实现好的平滑迁移？这个一直没有想到好的解决方法

  作者回复2020-04-02 12:55:55
  如果用的是官方的Redis Cluster，可以用它提供的redis-trib.rb自动平滑迁移。

  像Codis也提供了平滑迁移的管理工具。如果是自建的集群，相对就比较麻烦了。

  我会在后面的第20节课讲，怎么来平滑的迁移数据库，方法同样适用于迁移Redis。

- sami 2020-04-02 10:45:05
  cluster命令是受限的，跟mysql的分库分表一样，有一些场景也是无法支持

- haijian.yang 2020-04-02 09:54:26
  阿里云有个 redis 方案，冷数据存磁盘，热数据放在内存。

- 特种流氓 2020-04-02 08:55:07
  redis cluster集群 是不是就没有哨兵的概念了

  作者回复2020-04-02 09:57:37
  是的，redis cluster已经包含哨兵的功能了。

- leslie 2020-04-02 01:59:59
  其实这也反向提出了另一个问题mysql cluster的问题：其自身同样有cluster,可是真实环境下很少企业会

去用mysql cluster；都是各自运用自己的方案去做集群，方案之前的课程中都有提及就不提了。
自身的cluster在实际生产中极少被某些数据库使用这大概算是一大特点，尤其像我们所常见的mysql、redis、以及后面新出来的、、、