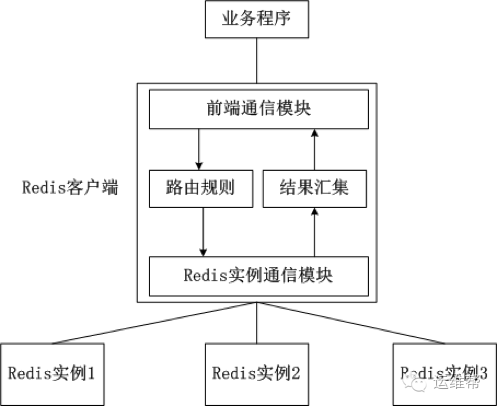
　由于Redis出众的性能，其在众多的移动互联网企业中得到广泛的应用。Redis在3.0版本前只支持单实例模式，虽然现在的服务器内存可以到100GB、200GB的规模，但是单实例模式限制了Redis没法满足业务的需求（例如新浪微博就曾经用Redis存储了超过1TB的数据）。Redis的开发者Antirez早在博客上就提出在Redis 3.0版本中加入集群的功能，但3.0版本等到2015年才发布正式版。各大企业在3.0版本还没发布前为了解决Redis的存储瓶颈，纷纷推出了各自的Redis集群方案。这些方案的核心思想是把数据分片（sharding）存储在多个Redis实例中，每一片就是一个Redis实例。

　　下面介绍Redis的集群方案。

****1.客户端分片****

　　客户端分片是把分片的逻辑放在Redis客户端实现，通过Redis客户端预先定义好的路由规则，把对Key的访问转发到不同的Redis实例中，最后把返回结果汇集。这种方案的模式如图1所示。



*图1 客户端分片的模式*

　　客户端分片的好处是所有的逻辑都是可控的，不依赖于第三方分布式中间件。开发人员清楚怎么实现分片、路由的规则，不用担心踩坑。

　　客户端分片方案有下面这些缺点。

这是一种静态的分片方案，需要增加或者减少Redis实例的数量，需要手工调整分片的程序。

可运维性差，集群的数据出了任何问题都需要运维人员和开发人员一起合作，减缓了解决问题的速度，增加了跨部门沟通的成本。

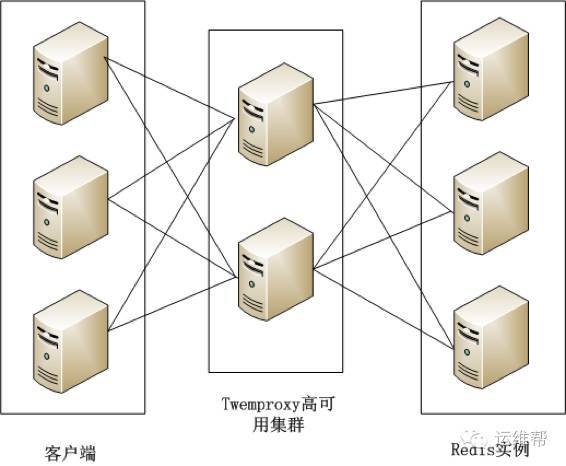
在不同的客户端程序中，维护相同的分片逻辑成本巨大。例如，系统中有两套业务系统共用一套Redis集群，一套业务系统用Java实现，另一套业务系统用PHP实现。为了保证分片逻辑的一致性，在Java客户端中实现的分片逻辑也需要在PHP客户端实现一次。相同的逻辑在不同的系统中分别实现，这种设计本来就非常糟糕，而且需要耗费巨大的开发成本保证两套业务系统分片逻辑的一致性。

****2.Twemproxy****

　　Twemproxy是由Twitter开源的Redis代理，其基本原理是：Redis客户端把请求发送到Twemproxy，Twemproxy根据路由规则发送到正确的Redis实例，最后Twemproxy把结果汇集返回给客户端。

　　Twemproxy通过引入一个代理层，将多个Redis实例进行统一管理，使Redis客户端只需要在Twemproxy上进行操作，而不需要关心后面有多少个Redis实例，从而实现了Redis集群。

　　Twemproxy集群架构如图2所示。



*图2Twemproxy集群架构*

　　Twemproxy的优点如下。

客户端像连接Redis实例一样连接Twemproxy，不需要改任何的代码逻辑。

支持无效Redis实例的自动删除。

Twemproxy与Redis实例保持连接，减少了客户端与Redis实例的连接数。

　　Twemproxy有如下不足。

由于Redis客户端的每个请求都经过Twemproxy代理才能到达Redis服务器，这个过程中会产生性能损失。

没有友好的监控管理后台界面，不利于运维监控。

最大的问题是Twemproxy无法平滑地增加Redis实例。对于运维人员来说，当因为业务需要增加Redis实例时工作量非常大。

　　Twemproxy作为最被广泛使用、最久经考验、稳定性最高的Redis代理，在业界被广泛使用。

****3.Codis****

　　Twemproxy不能平滑增加Redis实例的问题带来了很大的不便，于是豌豆荚自主研发了Codis，一个支持平滑增加Redis实例的Redis代理软件，其基于Go和C语言开发，并于2014年11月在GitHub上开源。

　　Codis包含下面4个部分。

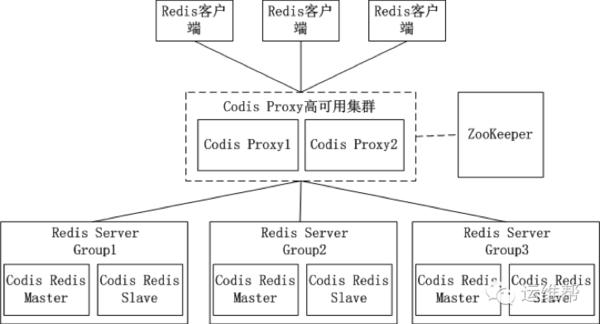
Codis Proxy：Redis客户端连接到Redis实例的代理，实现了Redis的协议，Redis客户端连接到Codis Proxy进行各种操作。Codis Proxy是无状态的，可以用Keepalived等负载均衡软件部署多个Codis Proxy实现高可用。

CodisRedis：Codis项目维护的Redis分支，添加了slot和原子的数据迁移命令。Codis上层的 Codis Proxy和Codisconfig只有与这个版本的Redis通信才能正常运行。

Codisconfig：Codis管理工具。可以执行添加删除CodisRedis节点、添加删除Codis Proxy、数据迁移等操作。另外，Codisconfig自带了HTTP server，里面集成了一个管理界面，方便运维人员观察Codis集群的状态和进行相关的操作，极大提高了运维的方便性，弥补了Twemproxy的缺点。

ZooKeeper：分布式的、开源的应用程序协调服务，是Hadoop和Hbase的重要组件，其为分布式应用提供一致性服务，提供的功能包括：配置维护、名字服务、分布式同步、组服务等。Codis依赖于ZooKeeper存储数据路由表的信息和Codis Proxy节点的元信息。另外，Codisconfig发起的命令都会通过ZooKeeper同步到CodisProxy的节点。

　　Codis的架构如图3所示。



*图3Codis的架构图*

　　在图3的Codis的架构图中，Codis引入了Redis Server Group，其通过指定一个主CodisRedis和一个或多个从CodisRedis，实现了Redis集群的高可用。当一个主CodisRedis挂掉时，Codis不会自动把一个从CodisRedis提升为主CodisRedis，这涉及数据的一致性问题（Redis本身的数据同步是采用主从异步复制，当数据在主CodisRedis写入成功时，从CodisRedis是否已读入这个数据是没法保证的），需要管理员在管理界面上手动把从CodisRedis提升为主CodisRedis。

　　如果觉得麻烦，豌豆荚也提供了一个工具Codis-ha，这个工具会在检测到主CodisRedis挂掉的时候将其下线并提升一个从CodisRedis为主CodisRedis。

　　Codis中采用预分片的形式，启动的时候就创建了1024个slot，1个slot相当于1个箱子，每个箱子有固定的编号，范围是1~1024。slot这个箱子用作存放Key，至于Key存放到哪个箱子，可以通过算法“crc32(key)%1024”获得一个数字，这个数字的范围一定是1~1024之间，Key就放到这个数字对应的slot。例如，如果某个Key通过算法“crc32(key)%1024”得到的数字是5，就放到编码为5的slot（箱子）。1个slot只能放1个Redis Server Group，不能把1个slot放到多个Redis Server Group中。1个Redis Server Group最少可以存放1个slot，最大可以存放1024个slot。因此，Codis中最多可以指定1024个Redis Server Group。

　　Codis最大的优势在于支持平滑增加（减少）Redis Server Group（Redis实例），能安全、透明地迁移数据，这也是Codis 有别于Twemproxy等静态分布式 Redis 解决方案的地方。Codis增加了Redis Server Group后，就牵涉到slot的迁移问题。例如，系统有两个Redis Server Group，Redis Server Group和slot的对应关系如下。

|  |  |
| --- | --- |
| Redis Server Group | slot |
| 1 | 1~500 |
| 2 | 501~1024 |

　　当增加了一个Redis Server Group，slot就要重新分配了。Codis分配slot有两种方法。

　　第一种：通过Codis管理工具Codisconfig手动重新分配，指定每个Redis Server Group所对应的slot的范围，例如可以指定Redis Server Group和slot的新的对应关系如下。

|  |  |
| --- | --- |
| Redis Server Group | slot |
| 1 | 1~500 |
| 2 | 501~700 |
| 3 | 701~1024 |

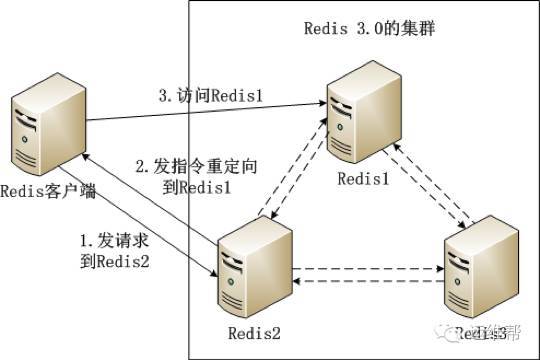
　　第二种：通过Codis管理工具Codisconfig的rebalance功能，会自动根据每个Redis Server Group的内存对slot进行迁移，以实现数据的均衡。

****4.Redis 3.0集群****

　　Redis 3.0集群采用了P2P的模式，完全去中心化。Redis把所有的Key分成了16384个slot，每个Redis实例负责其中一部分slot。集群中的所有信息（节点、端口、slot等），都通过节点之间定期的数据交换而更新。

　　Redis客户端在任意一个Redis实例发出请求，如果所需数据不在该实例中，通过重定向命令引导客户端访问所需的实例。

　　Redis 3.0集群的工作流程如图4所示。



*图4Redis 3.0集群的工作流程图*

　　如图4所示Redis集群内的机器定期交换数据，工作流程如下。

　　（1） Redis客户端在Redis2实例上访问某个数据。

　　（2） 在Redis2内发现这个数据是在Redis3这个实例中，给Redis客户端发送一个重定向的命令。

　　（3） Redis客户端收到重定向命令后，访问Redis3实例获取所需的数据。

　　Redis 3.0的集群方案有以下两个问题。

一个Redis实例具备了“数据存储”和“路由重定向”，完全去中心化的设计。这带来的好处是部署非常简单，直接部署Redis就行，不像Codis有那么多的组件和依赖。但带来的问题是很难对业务进行无痛的升级，如果哪天Redis集群出了什么严重的Bug，就只能回滚整个Redis集群。

对协议进行了较大的修改，对应的Redis客户端也需要升级。升级Redis客户端后谁能确保没有Bug？而且对于线上已经大规模运行的业务，升级代码中的Redis客户端也是一个很麻烦的事情。

　　综合上面所述的两个问题，Redis 3.0集群在业界并没有被大规模使用。

****5.云服务器上的集群服务****

　　国内的云服务器提供商阿里云、UCloud等均推出了基于Redis的云存储服务。

　　这个服务的特性如下。

　　（1）动态扩容

　　用户可以通过控制面板升级所需的Redis存储空间，扩容的过程中服务部不需要中断或停止，整个扩容过程对用户透明、无感知，这点是非常实用的，在前面介绍的方案中，解决Redis平滑扩容是个很烦琐的任务，现在按几下鼠标就能搞定，大大减少了运维的负担。

　　（2）数据多备

　　数据保存在一主一备两台机器中，其中一台机器宕机了，数据还在另外一台机器上有备份。

　　（3）自动容灾

　　主机宕机后系统能自动检测并切换到备机上，实现服务的高可用。

　　（4）实惠

　　很多情况下为了使Redis的性能更高，需要购买一台专门的服务器用于Redis的存储服务，但这样子CPU、内存等资源就浪费了，购买Redis云存储服务就很好地解决了这个问题。

　　有了Redis云存储服务，能使App后台开发人员从烦琐运维中解放出来。App后台要搭建一个高可用、高性能的Redis服务，需要投入相当的运维成本和精力。如果使用云存储服务，就没必要投入这些成本和精力，可以让App后台开发人员更专注于业务。