https://blog.csdn.net/men\_wen/article/details/72853078

**Redis Cluster 介绍与搭建**

**1. Redis Cluster介绍**

Redis Cluster是Redis的分布式解决方案，在Redis 3.0版本正式推出的，有效解决了Redis分布式方面的需求。当遇到单机内存、并发、流量等瓶颈时，可以采用Cluster架构达到负载均衡的目的。

**1.1 数据分布理论**

分布式数据库首要解决**把整个数据集按照分区规则映射到多个节点**的问题，即把数据集划分到多个节点上，每个节点负责整个数据的一个子集。常见的分区规则有**哈希分区**和**顺序分区**。Redis Cluster采用哈希分区规则，因此接下来会讨论哈希分区规则。常见的哈希分区有以下几种：

* 节点取余分区
* 一致性哈希分区
* 虚拟槽分区

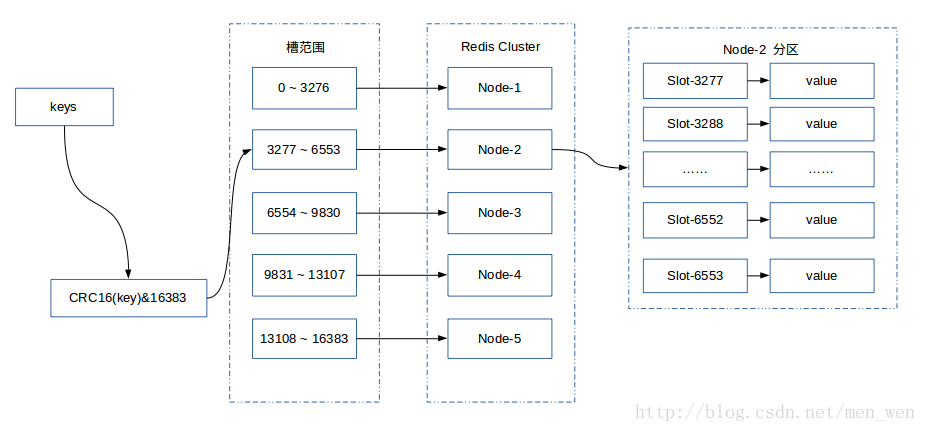
Redis Cluster采用虚拟槽分区，因此先介绍一下虚拟槽分区。

虚拟槽分区巧妙地使用了哈希空间，使用分散度良好的哈希函数把所有的数据映射到一个固定范围内的整数集合，整数定义为槽（slot）。比如Redis Cluster槽的范围是0 ～ 16383。槽是集群内数据管理和迁移的基本单位。采用大范围的槽的主要目的是为了方便数据的拆分和集群的扩展，每个节点负责一定数量的槽。

**1.2 Redis 数据分区**

Redis Cluster采用虚拟槽分区，所有的键根据哈希函数映射到0 ～ 16383，计算公式：slot = CRC16(key)&16383。每一个节点负责维护一部分槽以及槽所映射的键值数据。

下图展现一个五个节点构成的集群，每个节点平均大约负责3276个槽，以及通过计算公式映射到对应节点的对应槽的过程。



Redis虚拟槽分区的特定：

* 解耦数据和节点之间的关系，简化了节点扩容和收缩难度。
* 节点自身维护槽的映射关系，不需要客户端或者代理服务维护槽分区元数据。
* 支持节点、槽、键之间的映射查询，用于数据路由、在线伸缩等场景。

**1.3 Redis 集群功能限制**

Redis集群相对单机在功能上有一定限制。

* key批量操作支持有限。如：MSET``MGET，目前只支持具有相同slot值的key执行批量操作。
* key事务操作支持有限。支持多key在同一节点上的事务操作，不支持分布在多个节点的事务功能。
* key作为数据分区的最小粒度，因此不能将一个大的键值对象映射到不同的节点。如：hash、list。
* 不支持多数据库空间。单机下Redis支持16个数据库，集群模式下只能使用一个数据库空间，即db 0。
* 复制结构只支持一层，不支持嵌套树状复制结构。

**2. 搭建 Redis Cluster**

搭建集群工作分为三步：

* 准备节点
* 节点握手
* 分配槽

**2.1 准备节点**

Redis 集群一般由多个节点组成，节点数量为6个才能保证组成完整高可用的集群。下面给出一个节点的配置，其他的节点和该节点只是端口不同。

port 6379 //端口

cluster-enabled yes //开启集群模式

cluster-config-file nodes-6379.conf //集群内部的配置文件

cluster-node-timeout 15000 //节点超时时间，单位毫秒

// 其他配置和单机模式相同

* 1
* 2
* 3
* 4
* 5

启动所有的节点

sudo redis-server conf/redis-6384.conf

sudo redis-server conf/redis-6383.conf

sudo redis-server conf/redis-6382.conf

sudo redis-server conf/redis-6381.conf

sudo redis-server conf/redis-6380.conf

sudo redis-server conf/redis-6379.conf

* 1
* 2
* 3
* 4
* 5
* 6

可以查看日志文件

cat log/redis-6379.log

13103:M 30 May 15:02:09.577 \* DB loaded from disk: 0.000 seconds

13103:M 30 May 15:02:09.578 \* The server is now ready to accept connections on port 6379

* 1
* 2
* 3

有日志文件可得，节点已经启动成功。这个日志文件是Redis服务器普通的日志文件，在集群模式下，第一次也会自动创建一个日志文件，由配置文件cluster-config-file指定文件。

**集群配置文件的作用：当集群内节点发生信息变化时，如添加节点、节点下线、故障转移等。节点会自动保存集群的状态到配置文件中。**该配置文件由Redis自行维护，不要手动修改，防止节点重启时产生集群信息错乱。

我们来查看一下，集群模式的日志文件：

cat nodes-6379.conf

29978c0169ecc0a9054de7f4142155c1ab70258b :0 myself,master - 0 0 0 connected

vars currentEpoch 0 lastVoteEpoch 0

* 1
* 2
* 3

也可以通过客户端连接该节点，通过命令CLUSTER NODES来查看：

127.0.0.1:6379> CLUSTER NODES

29978c0169ecc0a9054de7f4142155c1ab70258b :6379 myself,master - 0 0 0 connected

* 1
* 2

**2.2 节点握手**

节点握手是指一批运行在集群模式的节点通过Gossip协议彼此通信，达到感知对方的过程。节点握手是集群彼此通信的第一步，由客户端发起命令：cluster meet <ip> <port>

127.0.0.1:6379> CLUSTER MEET 127.0.0.1 6380

OK

// 发送CLUSTER NODES可以查看到已经感知到 6380 端口的节点了。

127.0.0.1:6379> CLUSTER NODES

29978c0169ecc0a9054de7f4142155c1ab70258b 127.0.0.1:6379 myself,master - 0 0 1 connected

8f285670923d4f1c599ecc93367c95a30fb8bf34 127.0.0.1:6380 master - 0 1496129041442 0 connected

* 1
* 2
* 3
* 4
* 5
* 6

让所有的节点都互相感知：

127.0.0.1:6379> CLUSTER MEET 127.0.0.1 6381

OK

127.0.0.1:6379> CLUSTER MEET 127.0.0.1 6382

OK

127.0.0.1:6379> CLUSTER MEET 127.0.0.1 6383

OK

127.0.0.1:6379> CLUSTER MEET 127.0.0.1 6384

OK

// 已经全部感知到所有的节点

127.0.0.1:6379> CLUSTER NODES

e0c7961a1b07ab655bc31d8dfd583da565ec167d 127.0.0.1:6384 master - 0 1496129143703 0 connected

961097d6be64ebd2fd739ff719e97565a8cee7b5 127.0.0.1:6382 master - 0 1496129141678 0 connected

29978c0169ecc0a9054de7f4142155c1ab70258b 127.0.0.1:6379 myself,master - 0 0 1 connected

8f285670923d4f1c599ecc93367c95a30fb8bf34 127.0.0.1:6380 master - 0 1496129142682 3 connected

6fb7dfdb6188a9fe53c48ea32d541724f36434e9 127.0.0.1:6383 master - 0 1496129145699 4 connected

66478bda726ae6ba4e8fb55034d8e5e5804223ff 127.0.0.1:6381 master - 0 1496129147704 2 connected

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16

当前已经使这六个节点组成集群，但是现在还无法工作，因为集群节点还没有分配槽（slot）。

**2.3 分配槽**

可以看一下6379端口的槽个数

127.0.0.1:6379> CLUSTER INFO

cluster\_state:fail

cluster\_slots\_assigned:0 // 被分配槽的个数为0

cluster\_slots\_ok:0

cluster\_slots\_pfail:0

cluster\_slots\_fail:0

cluster\_known\_nodes:6

cluster\_size:0

cluster\_current\_epoch:5

cluster\_my\_epoch:1

cluster\_stats\_messages\_sent:479

cluster\_stats\_messages\_received:479

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12

接下来为节点分配槽空间。通过cluster addslots命令。

redis-cli -h 127.0.0.1 -p 6379 cluster addslots {0..5461}

OK

redis-cli -h 127.0.0.1 -p 6380 cluster addslots {5462..10922}

OK

redis-cli -h 127.0.0.1 -p 6381 cluster addslots {10923..16383}

OK

* 1
* 2
* 3
* 4
* 5
* 6

我们将16383个槽平均分配给6379、6380、6381端口的节点。再次执行CLUSTER INFO查看一下集群的状态：

127.0.0.1:6379> CLUSTER INFO

cluster\_state:ok // 集群状态OK

cluster\_slots\_assigned:16384 // 已经分配了所有的槽

cluster\_slots\_ok:16384

cluster\_slots\_pfail:0

cluster\_slots\_fail:0

cluster\_known\_nodes:6

cluster\_size:3

cluster\_current\_epoch:5

cluster\_my\_epoch:1

cluster\_stats\_messages\_sent:1212

cluster\_stats\_messages\_received:1212

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12

可以通过CLUSTER NODES来查看分配情况：

127.0.0.1:6379> CLUSTER NODES

e0c7961a1b07ab655bc31d8dfd583da565ec167d 127.0.0.1:6384 master - 0 1496129666347 0 connected

961097d6be64ebd2fd739ff719e97565a8cee7b5 127.0.0.1:6382 master - 0 1496129664844 5 connected

29978c0169ecc0a9054de7f4142155c1ab70258b 127.0.0.1:6379 myself,master - 0 0 1 connected 0-5461

8f285670923d4f1c599ecc93367c95a30fb8bf34 127.0.0.1:6380 master - 0 1496129665846 3 connected 5462-10922

6fb7dfdb6188a9fe53c48ea32d541724f36434e9 127.0.0.1:6383 master - 0 1496129661838 4 connected

66478bda726ae6ba4e8fb55034d8e5e5804223ff 127.0.0.1:6381 master - 0 1496129666848 2 connected 10923-16383

* 1
* 2
* 3
* 4
* 5
* 6
* 7

目前还有三个节点没有使用，作为一个完整的集群，每个负责处理槽的节点应该具有从节点，保证当主节点出现故障时，可以自动进行故障转移。集群模式下，首次启动的节点和被分配槽的节点都是主节点，从节点负责复制主节点槽的信息和相关数据。

使用cluster replicate <nodeid>在从节点上执行。

redis-cli -h 127.0.0.1 -p 6382 cluster replicate 29978c0169ecc0a9054de7f4142155c1ab70258b

OK

redis-cli -h 127.0.0.1 -p 6383 cluster replicate 8f285670923d4f1c599ecc93367c95a30fb8bf34

OK

redis-cli -h 127.0.0.1 -p 6384 cluster replicate 66478bda726ae6ba4e8fb55034d8e5e5804223ff

OK

* 1
* 2
* 3
* 4
* 5
* 6

通过CLUSTER NODES可以查看集群节点的状态

127.0.0.1:6379> CLUSTER NODES

e0c7961a1b07ab655bc31d8dfd583da565ec167d 127.0.0.1:6384 slave 66478bda726ae6ba4e8fb55034d8e5e5804223ff 0 1496130082754 2 connected

961097d6be64ebd2fd739ff719e97565a8cee7b5 127.0.0.1:6382 slave 29978c0169ecc0a9054de7f4142155c1ab70258b 0 1496130080749 5 connected

29978c0169ecc0a9054de7f4142155c1ab70258b 127.0.0.1:6379 myself,master - 0 0 1 connected 0-5461

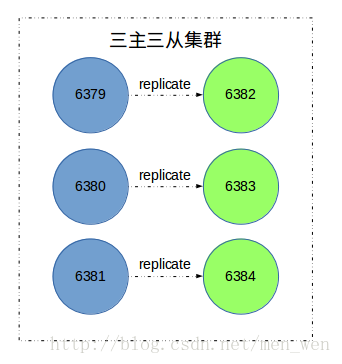
8f285670923d4f1c599ecc93367c95a30fb8bf34 127.0.0.1:6380 master - 0 1496130078744 3 connected 5462-10922

6fb7dfdb6188a9fe53c48ea32d541724f36434e9 127.0.0.1:6383 slave 8f285670923d4f1c599ecc93367c95a30fb8bf34 0 1496130079747 4 connected

66478bda726ae6ba4e8fb55034d8e5e5804223ff 127.0.0.1:6381 master - 0 1496130081751 2 connected 10923-16383

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

这样就完成了一个**3主3从的Redis集群**搭建。如下图所示：



本文参考：《Redis开发与运维》