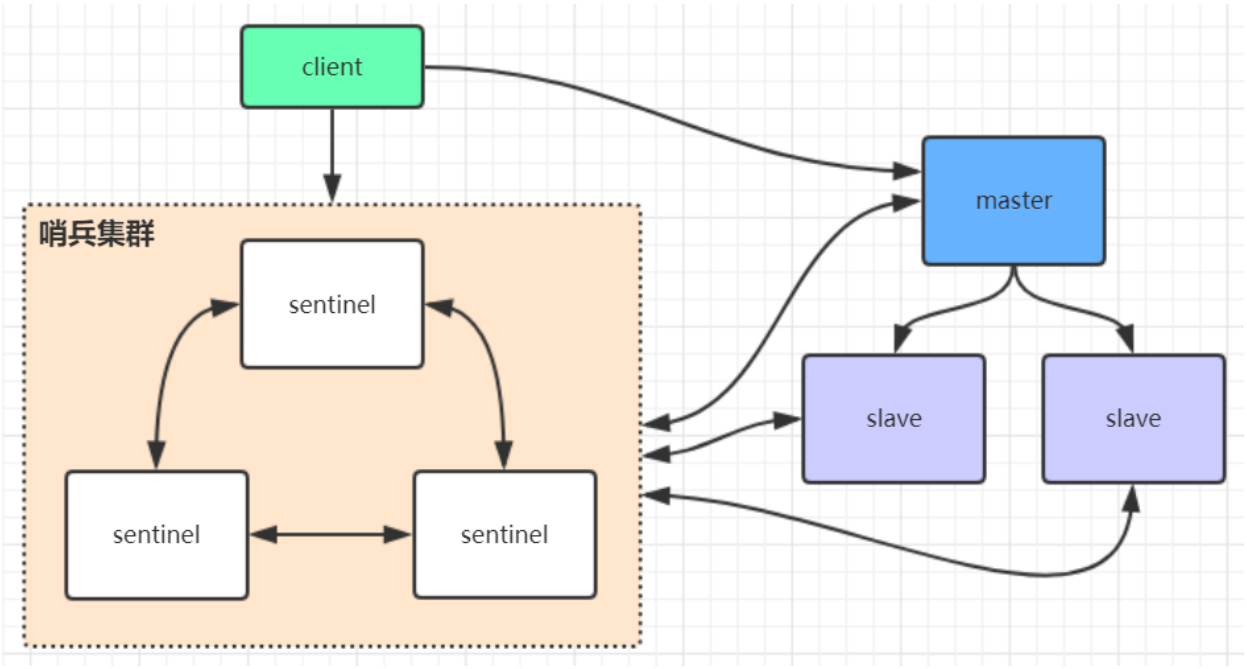
作者：图灵-诸葛老师

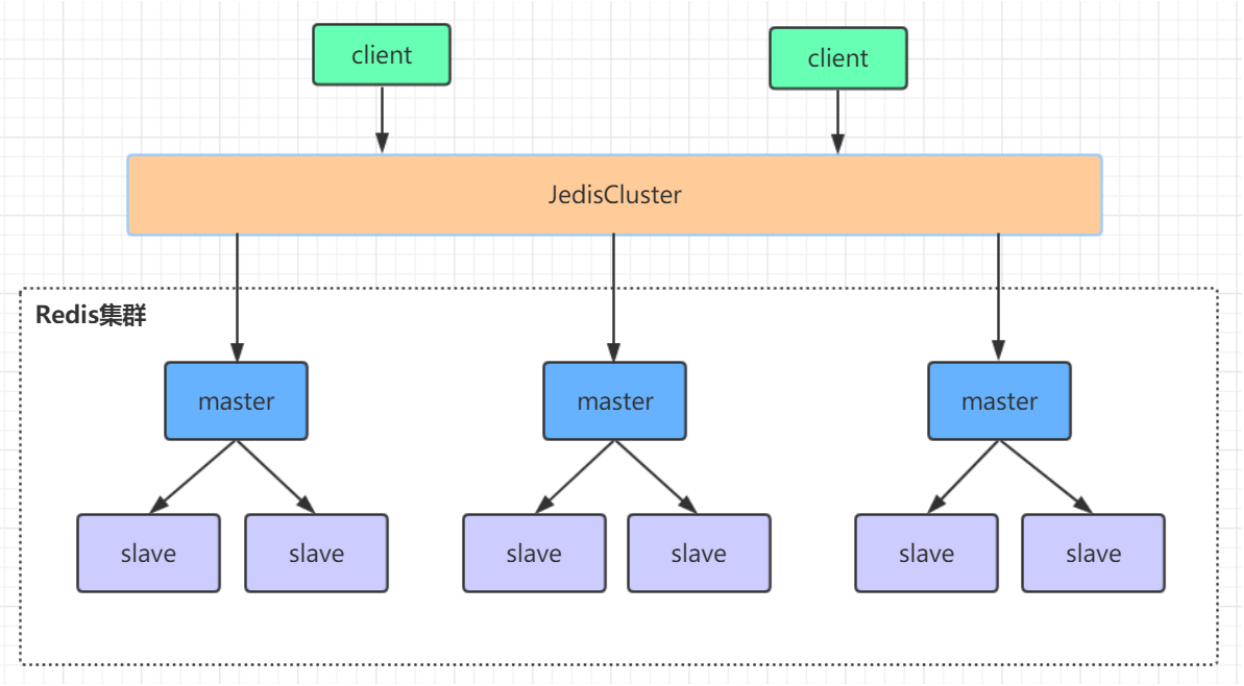
1、Redis集群方案比较

- 哨兵模式



在redis3.0以前的版本要实现集群一般是借助哨兵sentinel工具来监控master节点的状态，如果master节点异常，则会做主从切换，将某一台slave作为master，哨兵的配置略微复杂，并且性能和高可用性等各方面表现一般，特别是在主从切换的瞬间存在访问瞬断的情况，而且哨兵模式只有一个主节点对外提供服务，没法支持很高的并发，且单个主节点内存也不宜设置得过大，否则会导致持久化文件过大，影响数据恢复或主从同步的效率

- 高可用集群模式



redis集群是一个由多个主从节点群组成的分布式服务器群，它具有复制、高可用和分片特性。Redis集群不需要sentinel哨兵也能完成节点移除和故障转移的功能。需要将每个节点设置成集群模式，这种集群模式没有中心节点，可水平扩展，据官方文档称可以线性扩展到上万个节点(官方推荐不超过1000个节点)。redis集群的性能和高可用性均优于之前版本的哨兵模式，且集群配置非常简单

2、Redis高可用集群搭建

- **redis安装**

下载地址：http://redis.io/download

安装步骤：

```
# 安装gcc
yum install gcc

# 把下载好的redis-5.0.2.tar.gz放在/usr/local文件夹下，并解压
wget http://download.redis.io/releases/redis-5.0.2.tar.gz
tar xzf redis-5.0.2.tar.gz
cd redis-5.0.2

# 进入到解压好的redis-5.0.2目录下，进行编译与安装
make & make install

# 启动并指定配置文件
src/redis-server redis.conf（注意要使用后台启动，所以修改redis.conf里的daemonize
```

改为yes)

验证启动是否成功
ps -ef | grep redis

进入redis客户端
/usr/local/redis/bin/redis-cli

退出客户端
quit

退出redis服务：
（1）pkill redis-server
（2）kill 进程号
（3）src/redis-cli shutdown

- **redis集群搭建**

redis集群需要至少要三个master节点，我们这里搭建三个master节点，并且给每个master再搭建一个slave节点，总共6个redis节点，这里用三台机器部署6个redis实例，每台机器一主一从，搭建集群的步骤如下：

第一步：在第一台机器的/usr/local下创建文件夹redis-cluster，然后在其下面分别创建2个文件夹如下
（1）mkdir -p /usr/local/redis-cluster
（2）mkdir 8001、 mkdir 8004

第一步：把之前的redis.conf配置文件copy到8001下，修改如下内容：
（1）daemonize yes
（2）port 8001（分别对每个机器的端口号进行设置）
（3）dir /usr/local/redis-cluster/8001/（指定数据文件存放位置，必须要指定不同的目录位置，不然会丢失数据）
（4）cluster-enabled yes（启动集群模式）
（5）cluster-config-file nodes-8001.conf（集群节点信息文件，这里800x最好和port对应上）
（6）cluster-node-timeout 5000
（7）# bind 127.0.0.1（去掉bind绑定访问ip信息）

（8）protected-mode no （关闭保护模式）
（9）appendonly yes
如果要设置密码需要增加如下配置：
（10）requirepass zhuge （设置redis访问密码）
（11）masterauth zhuge （设置集群节点间访问密码，跟上面一致）

第三步：把修改后的配置文件，copy到8002，修改第2、3、5项里的端口号，可以用批量替换：
:%s/源字符串/目的字符串/g

第四步：另外两台机器也需要做上面几步操作，第二台机器用8002和8005，第三台机器用8003和8006

第五步：分别启动6个redis实例，然后检查是否启动成功
（1）/usr/local/redis-5.0.2/src/redis-server /usr/local/redis-cluster/800*/redis.conf
（2）ps -ef | grep redis 查看是否启动成功

第六步：用redis-cli创建整个redis集群(redis5以前的版本集群是依靠ruby脚本redis-trib.rb实现)
（1）/usr/local/redis-5.0.2/src/redis-cli -a zhuge --cluster create --cluster-replicas 1 192.168.0.61:8001 192.168.0.62:8002 192.168.0.63:8003 192.168.0.61:8004 192.168.0.62:8005 192.168.0.63:8006 代表为每个创建的主服务器节点创建一个从服务器节点

第七步：验证集群：
（1）连接任意一个客户端即可：./redis-cli -c -h -p (-a访问服务端密码，-c表示集群模式，指定ip地址和端口号）如：/usr/local/redis-5.0.2/src/redis-cli -a zhuge -c -h 192.168.0.61 -p 800*
（2）进行验证： cluster info（查看集群信息）、cluster nodes（查看节点列表）
（3）进行数据操作验证
（4）关闭集群则需要逐个进行关闭，使用命令：
/usr/local/redis/bin/redis-cli -a zhuge -c -h 192.168.0.60 -p 800* shutdown

---

3、Java操作redis集群

借助redis的java客户端jedis可以操作以上集群，引用jedis版本的maven坐标如下：

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.9.0</version>
</dependency>
```

Java编写访问redis集群的代码非常简单，如下所示：

```
import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.JedisCluster;
import redis.clients.jedis.JedisPoolConfig;

/**
 * 访问redis集群
 * @author 图灵-诸葛老师
 *
 */
public class RedisCluster
{
    public static void main(String[] args) throws IOException
    {
```

```
        Set<HostAndPort> jedisClusterNode = new HashSet<HostAndPort>();
        jedisClusterNode.add(new HostAndPort("192.168.0.61", 8001));
        jedisClusterNode.add(new HostAndPort("192.168.0.62", 8002));
        jedisClusterNode.add(new HostAndPort("192.168.0.63", 8003));
        jedisClusterNode.add(new HostAndPort("192.168.0.61", 8004));
        jedisClusterNode.add(new HostAndPort("192.168.0.62", 8005));
        jedisClusterNode.add(new HostAndPort("192.168.0.63", 8006));

        JedisPoolConfig config = new JedisPoolConfig();
        config.setMaxTotal(100);
        config.setMaxIdle(10);
        config.setTestOnBorrow(true);
        //connectionTimeout：指的是连接一个url的连接等待时间
        //soTimeout：指的是连接上一个url，获取response的返回等待时间
        JedisCluster jedisCluster = new JedisCluster(jedisClusterNode, 6000, 5000, 10,
"zhuge", config);
        System.out.println(jedisCluster.set("student", "zhuge"));
        System.out.println(jedisCluster.set("age", "19"));

        System.out.println(jedisCluster.get("student"));
        System.out.println(jedisCluster.get("age"));

        jedisCluster.close();
    }
```

```
}
```

运行效果如下：

```
OK
OK
aaron
18
```

3、Redis集群原理分析

Redis Cluster 将所有数据划分为 16384 的 slots(槽位)，每个节点负责其中一部分槽位。槽位的信息存储于每个节点中，。

当 Redis Cluster 的客户端来连接集群时，它也会得到一份集群的槽位配置信息并将其缓存在客户端本地。这样当客户端要查找某个 key 时，可以直接定位到目标节点。同时因为槽位的信息可能会存在客户端与服务器不一致的情况，还需要纠正机制来实现槽位信息的校验调整。

槽位定位算法

Cluster 默认会对 key 值使用 crc16 算法进行 hash 得到一个整数值，然后用这个整数值对 16384 进行取模来得到具体槽位。

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

跳转重定位

当客户端向一个错误的节点发出了指令，该节点会发现指令的 key 所在的槽位并不归自己管理，这时它会向客户端发送一个特殊的跳转指令携带目标操作的节点地址，告诉客户端去连这个节点去获取数据。客户端收到指令后除了跳转到正确的节点上去操作，还会同步更新纠正本地的槽位映射表缓存，后续所有 key 将使用新的槽位映射表。

```
192.168.0.61:8001> set name abc
-> Redirected to slot [5798] located at 192.168.0.61:8004
OK
192.168.0.61:8004>
```

网络抖动

真实世界的机房网络往往并不是风平浪静的，它们经常会发生各种各样的小问题。比如网络抖动就是非常常见的一种现象，突然之间部分连接变得不可访问，然后很快又恢复正常。

为解决这种问题，Redis Cluster 提供了一种选项`cluster-node-timeout`，表示当某个节点持续 timeout 的时间失联时，才可以认定该节点出现故障，需要进行主从切换。如果没有这个选项，网络抖动会导致主从频繁切换 (数据的重新复制)。