**品优购电商系统开发**

**第18章**

**集群解决方案**

传智播客.黑马程序员

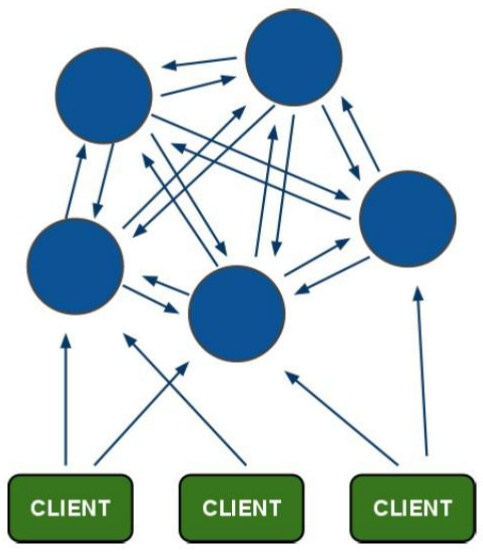
Redis Cluster

# Redis-Cluster简介

## 什么是Redis-Cluster

为何要搭建Redis集群。Redis是在内存中保存数据的，而我们的电脑一般内存都不大，这也就意味着Redis不适合存储大数据，适合存储大数据的是Hadoop生态系统的Hbase或者是MogoDB。Redis更适合处理高并发，一台设备的存储能力是很有限的，但是多台设备协同合作，就可以让内存增大很多倍，这就需要用到集群。

Redis集群搭建的方式有多种，例如使用客户端分片、Twemproxy、Codis等，但从redis 3.0之后版本支持redis-cluster集群，它是Redis官方提出的解决方案，Redis-Cluster采用无中心结构，每个节点保存数据和整个集群状态,每个节点都和其他所有节点连接。其redis-cluster架构图如下：



**架构细节:**

（1）所有的 redis 节点彼此互联(PING-PONG 机制),内部使用二进制协议优化传输速度和带宽.

（2）节点的 fail 是通过集群中超过半数的节点检测失效时才生效.

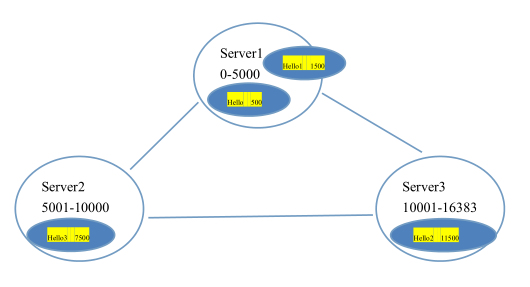
（3）客户端与 redis 节点直连,不需要中间 proxy 层.客户端不需要连接集群所有节点,连接集群中任何一个可用节点即可

## 分布存储机制-槽

（1）redis-cluster 把所有的物理节点映射到[0-16383]slot 上,cluster 负责维护

node<->slot<->value

（2）Redis 集群中内置了 16384 个哈希槽，当需要在 Redis 集群中放置一个 key-value 时，redis 先对 key 使用 crc16 算法算出一个结果，然后把结果对 16384 求余数，这样每个key 都会对应一个编号在 0-16383 之间的哈希槽，redis 会根据节点数量大致均等的将哈希槽映射到不同的节点。



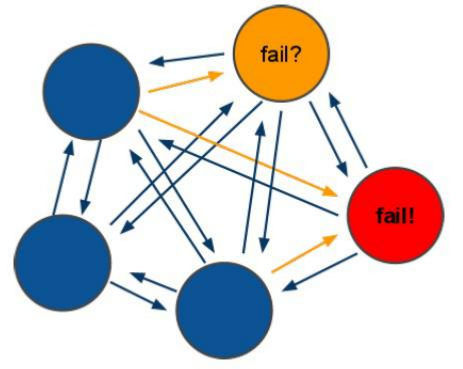
## 容错机制-投票

（1）选举过程是集群中所有master参与,如果半数以上master节点与故障节点通信超过(cluster-node-timeout),认为该节点故障，自动触发故障转移操作.

（2）什么时候整个集群不可用(cluster\_state:fail)?

    a: 如果集群任意master挂掉,且当前master没有slave.集群进入fail状态,也可以理解成集群的slot映射[0-16383]不完成时进入fail状态.

b: 如果集群超过半数以上master挂掉，无论是否有slave集群进入fail状态.



# 搭建Redis-Cluster

## 搭建要求

需要 6 台 redis 服务器。搭建伪集群。

需要 6 个 redis 实例。

需要运行在不同的端口 7001-7006

## 准备工作

1. 安装gcc 【此步省略】

Redis 是 c 语言开发的。安装 redis 需要 c 语言的编译环境。如果没有 gcc 需要在线安装。

|  |
| --- |
| yum install gcc-c++ |

1. 使用yum命令安装 ruby （我们需要使用ruby脚本来实现集群搭建）【此步省略】

|  |
| --- |
| yum install ruby  yum install rubygems |

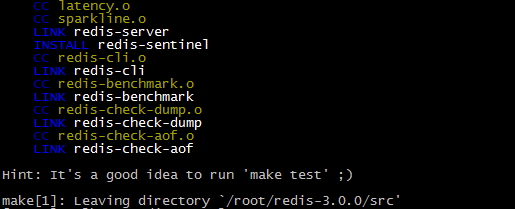
|  |
| --- |
| **----- 知识点小贴士 -----**  Ruby，一种简单快捷的[面向对象](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1)（[面向对象程序设计](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1)）[脚本语言](https://baike.baidu.com/item/%E8%84%9A%E6%9C%AC%E8%AF%AD%E8%A8%80)，在20世纪90年代由日本人松本行弘([Yukihiro Matsumoto](https://baike.baidu.com/item/Yukihiro%20Matsumoto))开发，遵守[GPL](https://baike.baidu.com/item/GPL)协议和Ruby License。它的灵感与特性来自于 [Perl](https://baike.baidu.com/item/Perl)、[Smalltalk](https://baike.baidu.com/item/Smalltalk)、[Eiffel](https://baike.baidu.com/item/Eiffel)、[Ada](https://baike.baidu.com/item/Ada/5606819)以及 [Lisp](https://baike.baidu.com/item/Lisp/22083) 语言。由 Ruby 语言本身还发展出了JRuby（[Java平台](https://baike.baidu.com/item/Java%E5%B9%B3%E5%8F%B0)）、IronRuby（[.NET](https://baike.baidu.com/item/.NET)平台）等其他平台的 Ruby 语言[替代品](https://baike.baidu.com/item/%E6%9B%BF%E4%BB%A3%E5%93%81)。Ruby的作者于1993年2月24日开始编写Ruby，直至1995年12月才正式[公开发布](https://baike.baidu.com/item/%E5%85%AC%E5%BC%80%E5%8F%91%E5%B8%83)于fj（[新闻组](https://baike.baidu.com/item/%E6%96%B0%E9%97%BB%E7%BB%84)）。因为Perl发音与6月诞生石pearl（[珍珠](https://baike.baidu.com/item/%E7%8F%8D%E7%8F%A0/33579)）相同，因此Ruby以7月诞生石ruby（[红宝石](https://baike.baidu.com/item/%E7%BA%A2%E5%AE%9D%E7%9F%B3/24242)）命名  RubyGems简称gems，是一个用于对 Ruby组件进行打包的 Ruby 打包系统 |

（3）将redis源码包上传到 linux 系统 ，解压redis源码包

（4）编译redis源码 ，进入redis源码文件夹

|  |
| --- |
| make |

看到以下输出结果，表示编译成功



（5）创建目录/usr/local/redis-cluster目录， 安装6个redis实例，分别安装在以下目录

/usr/local/redis-cluster/redis-1

/usr/local/redis-cluster/redis-2

/usr/local/redis-cluster/redis-3

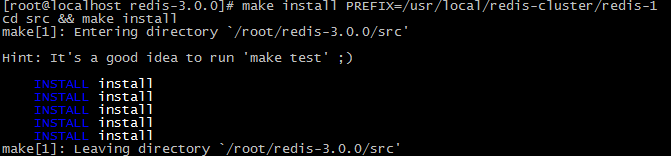
/usr/local/redis-cluster/redis-4

/usr/local/redis-cluster/redis-5

/usr/local/redis-cluster/redis-6

以第一个redis实例为例，命令如下

|  |
| --- |
| make install PREFIX=/usr/local/redis-cluster/redis-1 |



出现此提示表示成功，按此方法安装其余5个redis实例

（6）复制配置文件 将 /redis-3.0.0/redis.conf 复制到redis下的bin目录下

|  |
| --- |
| [root@localhost redis-3.0.0]# cp redis.conf /usr/local/redis-cluster/redis-1/bin  [root@localhost redis-3.0.0]# cp redis.conf /usr/local/redis-cluster/redis-2/bin  [root@localhost redis-3.0.0]# cp redis.conf /usr/local/redis-cluster/redis-3/bin  [root@localhost redis-3.0.0]# cp redis.conf /usr/local/redis-cluster/redis-4/bin  [root@localhost redis-3.0.0]# cp redis.conf /usr/local/redis-cluster/redis-5/bin  [root@localhost redis-3.0.0]# cp redis.conf /usr/local/redis-cluster/redis-6/bin |

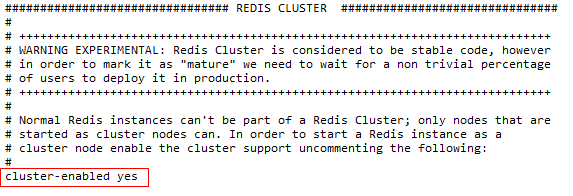
## 配置集群

1. 修改配置文件

修改运行端口为7001 （7002 7003 .....）



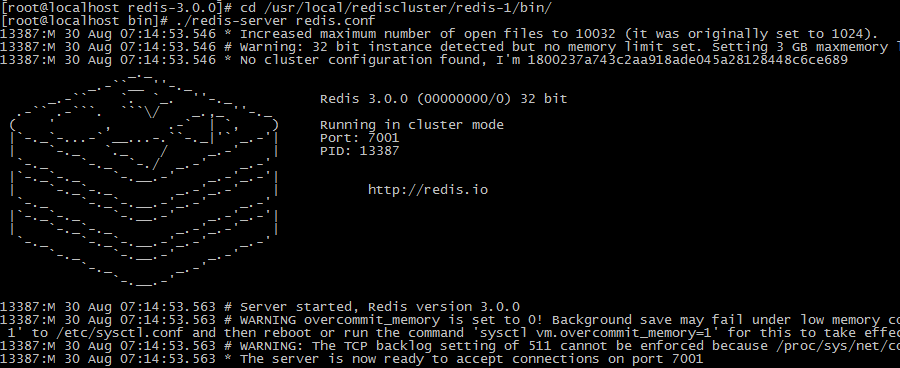
将cluster-enabled yes 前的注释去掉



（2）启动每个redis实例

以第一个实例为例，命令如下

|  |
| --- |
| cd /usr/local/redis-cluster/redis-1/bin/  ./redis-server redis.conf |



把其余的5个也启动起来，然后查看一下是不是都启动起来了

|  |
| --- |
| [root@localhost ~]# ps -ef | grep redis  root 15776 15775 0 08:19 pts/1 00:00:00 ./redis-server \*:7001 [cluster]  root 15810 15784 0 08:22 pts/2 00:00:00 ./redis-server \*:7002 [cluster]  root 15831 15813 0 08:23 pts/3 00:00:00 ./redis-server \*:7003 [cluster]  root 15852 15834 0 08:23 pts/4 00:00:00 ./redis-server \*:7004 [cluster]  root 15872 15856 0 08:24 pts/5 00:00:00 ./redis-server \*:7005 [cluster]  root 15891 15875 0 08:24 pts/6 00:00:00 ./redis-server \*:7006 [cluster]  root 15926 15895 0 08:24 pts/7 00:00:00 grep redis |

（3）上传redis-3.0.0.gem ，安装 ruby用于搭建redis集群的脚本。

|  |
| --- |
| [root@localhost ~]# gem install redis-3.0.0.gem  Successfully installed redis-3.0.0  1 gem installed  Installing ri documentation for redis-3.0.0...  Installing RDoc documentation for redis-3.0.0... |

1. 使用 ruby 脚本搭建集群。

进入

|  |
| --- |
| ./redis-trib.rb create --replicas 1 192.168.25.135:7001 192.168.25.135:7002 192.168.25.135:7003  192.168.25.135:7004 192.168.25.135:7005 192.168.25.135:7006 |

出现下列提示信息

|  |
| --- |
| >>> Creating cluster  Connecting to node 192.168.25.135:7001: OK  Connecting to node 192.168.25.135:7002: OK  Connecting to node 192.168.25.135:7003: OK  Connecting to node 192.168.25.135:7004: OK  Connecting to node 192.168.25.135:7005: OK  Connecting to node 192.168.25.135:7006: OK  >>> Performing hash slots allocation on 6 nodes...  Using 3 masters:  192.168.25.135:7001  192.168.25.135:7002  192.168.25.135:7003  Adding replica 192.168.25.135:7004 to 192.168.25.135:7001  Adding replica 192.168.25.135:7005 to 192.168.25.135:7002  Adding replica 192.168.25.135:7006 to 192.168.25.135:7003  M: 1800237a743c2aa918ade045a28128448c6ce689 192.168.25.135:7001  slots:0-5460 (5461 slots) master  M: 7cb3f7d5c60bfbd3ab28800f8fd3bf6de005bf0d 192.168.25.135:7002  slots:5461-10922 (5462 slots) master  M: 436e88ec323a2f8bb08bf09f7df07cc7909fcf81 192.168.25.135:7003  slots:10923-16383 (5461 slots) master  S: c2a39a94b5f41532cd83bf6643e98fc277c2f441 192.168.25.135:7004  replicates 1800237a743c2aa918ade045a28128448c6ce689  S: b0e38d80273515c84b1a01820d8ecee04547d776 192.168.25.135:7005  replicates 7cb3f7d5c60bfbd3ab28800f8fd3bf6de005bf0d  S: 03bf6bd7e3e6eece5a02043224497c2c8e185132 192.168.25.135:7006  replicates 436e88ec323a2f8bb08bf09f7df07cc7909fcf81  Can I set the above configuration? (type 'yes' to accept): yes  >>> Nodes configuration updated  >>> Assign a different config epoch to each node  >>> Sending CLUSTER MEET messages to join the cluster  Waiting for the cluster to join....  >>> Performing Cluster Check (using node 192.168.25.135:7001)  M: 1800237a743c2aa918ade045a28128448c6ce689 192.168.25.135:7001  slots:0-5460 (5461 slots) master  M: 7cb3f7d5c60bfbd3ab28800f8fd3bf6de005bf0d 192.168.25.135:7002  slots:5461-10922 (5462 slots) master  M: 436e88ec323a2f8bb08bf09f7df07cc7909fcf81 192.168.25.135:7003  slots:10923-16383 (5461 slots) master  M: c2a39a94b5f41532cd83bf6643e98fc277c2f441 192.168.25.135:7004  slots: (0 slots) master  replicates 1800237a743c2aa918ade045a28128448c6ce689  M: b0e38d80273515c84b1a01820d8ecee04547d776 192.168.25.135:7005  slots: (0 slots) master  replicates 7cb3f7d5c60bfbd3ab28800f8fd3bf6de005bf0d  M: 03bf6bd7e3e6eece5a02043224497c2c8e185132 192.168.25.135:7006  slots: (0 slots) master  replicates 436e88ec323a2f8bb08bf09f7df07cc7909fcf81  [OK] All nodes agree about slots configuration.  >>> Check for open slots...  >>> Check slots coverage...  [OK] All 16384 slots covered. |

# 连接Redis-Cluster

## 客户端工具连接

Redis-cli 连接集群。

|  |
| --- |
| redis-cli -p 7002 -c |

-c：代表连接的是 redis 集群

测试值的存取

|  |
| --- |
| [root@localhost bin]# ./redis-cli -p 7004 -c  127.0.0.1:7004> set name abc  -> Redirected to slot [5798] located at 192.168.25.135:7002  OK  192.168.25.135:7002> quit  [root@localhost bin]# ./redis-cli -p 7004  127.0.0.1:7004> get abc  (error) MOVED 7638 192.168.25.135:7002  127.0.0.1:7004> quit  [root@localhost bin]# cd ..  [root@localhost redis-4]# cd ..  [root@localhost redis-cluster]# cd redis-3  [root@localhost redis-3]# cd bin  [root@localhost bin]# ./redis-cli -p 7003 -c  127.0.0.1:7003> get name  -> Redirected to slot [5798] located at 192.168.25.135:7002  "abc"  192.168.25.135:7002> quit  [root@localhost bin]#  [root@localhost bin]#  [root@localhost bin]# cd ../..  [root@localhost redis-cluster]# cd redis-2  [root@localhost redis-2]# cd bin  [root@localhost bin]# ./redis-cli -p 7002  127.0.0.1:7002> get name  "abc"  127.0.0.1:7002> quit |

## SpringDataRedis连接

（1）创建SpringDataRedis-ClusterDemo工程 引入依赖

|  |
| --- |
| <!-- 集中定义依赖版本号 -->  <properties>  <spring.version>4.2.4.RELEASE</spring.version>  </properties>  <dependencies>  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jms</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.9</version>  </dependency>  <!-- 缓存 -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.8.1</version>  </dependency>  <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-redis</artifactId>  <version>1.7.2.RELEASE</version>  </dependency>  </dependencies>  <build>  <plugins>  <!-- java编译插件 -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.2</version>  <configuration>  <source>1.7</source>  <target>1.7</target>  <encoding>UTF-8</encoding>  </configuration>  </plugin>  </plugins>  </build> |

（2）添加spring 配置文件

applicationContext-redis-cluster.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd"*>  <!-- 加载配置属性文件 -->  <context:property-placeholder ignore-unresolvable=*"true"* location=*"classpath:properties/redis-cluster-config.properties"* />  <bean id=*"redis-clusterConfiguration"* class=*"org.springframework.data.redis.connection.redis-clusterConfiguration"*>  <property name=*"maxRedirects"* value=*"${redis.maxRedirects}"*></property>  <property name=*"clusterNodes"*>  <set>  <bean class=*"org.springframework.data.redis.connection.redis-clusterNode"*>  <constructor-arg name=*"host"* value=*"${redis.host1}"*></constructor-arg>  <constructor-arg name=*"port"* value=*"${redis.port1}"*></constructor-arg>  </bean>  <bean class=*"org.springframework.data.redis.connection.redis-clusterNode"*>  <constructor-arg name=*"host"* value=*"${redis.host2}"*></constructor-arg>  <constructor-arg name=*"port"* value=*"${redis.port2}"*></constructor-arg>  </bean>  <bean class=*"org.springframework.data.redis.connection.redis-clusterNode"*>  <constructor-arg name=*"host"* value=*"${redis.host3}"*></constructor-arg>  <constructor-arg name=*"port"* value=*"${redis.port3}"*></constructor-arg>  </bean>  <bean class=*"org.springframework.data.redis.connection.redis-clusterNode"*>  <constructor-arg name=*"host"* value=*"${redis.host4}"*></constructor-arg>  <constructor-arg name=*"port"* value=*"${redis.port4}"*></constructor-arg>  </bean>  <bean class=*"org.springframework.data.redis.connection.redis-clusterNode"*>  <constructor-arg name=*"host"* value=*"${redis.host5}"*></constructor-arg>  <constructor-arg name=*"port"* value=*"${redis.port5}"*></constructor-arg>  </bean>  <bean class=*"org.springframework.data.redis.connection.redis-clusterNode"*>  <constructor-arg name=*"host"* value=*"${redis.host6}"*></constructor-arg>  <constructor-arg name=*"port"* value=*"${redis.port6}"*></constructor-arg>  </bean>  </set>  </property>  </bean>  <bean id=*"jedisPoolConfig"* class=*"redis.clients.jedis.JedisPoolConfig"*>  <property name=*"maxIdle"* value=*"${redis.maxIdle}"* />  <property name=*"maxTotal"* value=*"${redis.maxTotal}"* />  </bean>  <bean id=*"jeidsConnectionFactory"* class=*"org.springframework.data.redis.connection.jedis.JedisConnectionFactory"* >  <constructor-arg ref=*"redis-clusterConfiguration"* />  <constructor-arg ref=*"jedisPoolConfig"* />  </bean>  <bean id=*"redisTemplate"* class=*"org.springframework.data.redis.core.RedisTemplate"*>  <property name=*"connectionFactory"* ref=*"jeidsConnectionFactory"* />  </bean>  </beans> |

（3）添加属性文件redis-cluster-config.properties

|  |
| --- |
| #cluster configuration  redis.host1=192.168.25.135  redis.port1=7001  redis.host2=192.168.25.135  redis.port2=7002  redis.host3=192.168.25.135  redis.port3=7003  redis.host4=192.168.25.135  redis.port4=7004  redis.host5=192.168.25.135  redis.port5=7005  redis.host6=192.168.25.135  redis.port6=7006  redis.maxRedirects=3  redis.maxIdle=100  redis.maxTotal=600 |

（4）编写测试类，测试值的存取

|  |
| --- |
| @RunWith(SpringJUnit4ClassRunner.**class**)  @ContextConfiguration(locations="classpath:spring/applicationContext-redis-cluster.xml")  **public** **class** TestSet {  @Autowired  **private** RedisTemplate<String,String> redisTemplate;  @Test  **public** **void** testSetValue(){  redisTemplate.boundSetOps("nameset").add("liming");  redisTemplate.boundSetOps("nameset").add("liuhu");  redisTemplate.boundSetOps("nameset").add("zhoujian");  }  @Test  **public** **void** testGetValue(){    Set<String> members = redisTemplate.boundSetOps("nameset").members();  System.***out***.println(members);  }  } |