一、安装zookeeper及配置日志

1、将安装包放进服务器，并解压

mkdir usr/local/zookeeper-cluster

tar -zxvf \*\*\*.tar.gz /usr/local/zookeeper-cluster/

2、解压后，在zookeeper目录下创建data目录

mkdir data

3、将conf下的zoo\_sample.cfg文件改名：

mv zoo\_sample.cfg zoo.cfg

4、将解压后的zookeeper在zookeeper-cluster目录中复制N份（具体视情况复制）

cp -r zookeeper-1 zookeeper-2

5、配置每一个zookeeper中zoo.cfg，如：

tickTime=2000

clientPort=2181

dataDir=/usr/local/zookeeper-cluster/zookeeper-1/data

initLimit=5

syncLimit=2

6、创建myid文件，记录每个服务器的id，简单方法，如内容是1：

echo 1 >myid

7、再设置zoo.cfg文件，将客户端访问端口和集群服务器IP列表写进文件中

如：server.1=192.168.136.22:2881:3881

server.2=192.168.136.22:2882:3882

server.3=192.168.136.22:2883:3883

或 server.1=192.168.136.22:2888:3881

server.2=192.168.136.22:2888:3882

server.3=192.168.136.22:2888:3883

注意：clientPort是客户端访问zookeeper的端口号，而2881等是zookeeper每个节点间通讯端口，3881是投票端口

8、修改日志路径

(1)修改bin/zkEnv.sh

命令：

mkdir /usr/local/zookeeper-cluster/zookeeper-1/logs

vi /usr/local/zookeeper-cluster/zookeeper-1/bin/zkEnv.sh

修改内容：

ZOO\_LOG\_DIR=” /usr/local/zookeeper-cluster/zookeeper-1/logs”

ZOO\_LOG4J\_PROP=”INFO,ROLLINGFILE”

(2)修改conf/log4j.properties

命令：

vi /usr/local/zookeeper-cluster/zookeeper-1/conf/log4j.properties

修改内容：

zookeeper.root.logger=INFO,ROLLINGFILE 🡪与zkEnv.sh的一致

注释：

log4j.appender.ROLLINGFILE默认是以日志文件大小轮转的，如果按照天轮转，可修改为DailyRollingFileAppender

9、启动集群，启动每个zookeeper，启动前要关闭防火墙

关闭防火墙命令：systemctl stop firewalld.service

打开防火墙命令：systemctl start firewalld.service

查看防火枪状态：systemctl status firewalld

/usr/local/zookeeper-cluster/zookeeper-1/bin/zkServer.sh start

10、查询集群服务

/usr/local/zookeeper-cluster/zookeeper-1/bin/zkServer.sh status

注：Mode为standalone，表示单机模式

Mode为follower，表示跟随者

Mode为leader，表示领导者

11、关闭集群服务

/usr/local/zookeeper-cluster/zookeeper-1/bin/zkServer.sh stop

12、查看集群服务

./zkServer.sh status

二、solr集群配置

1.1、将已部署完solr的tomcat放到服务器上

1.2、创建solr-cloud文件夹，命令：

mkdir /usr/local/solr-cloud

1.3、复制tomcat到solr-cloud中，命令：

cp -r apache-tomcat-solr /usr/local/solr-cloud/tomcat-1

cp -r apache-tomcat-solr /usr/local/solr-cloud/tomcat-2

cp -r apache-tomcat-solr /usr/local/solr-cloud/tomcat-3

cp -r apache-tomcat-solr /usr/local/solr-cloud/tomcat-4

1.4、将solrhome放到服务器上，并创建solrhomes文件夹，命令：

mkdir /usr/local/solrhomes

1.5、复制solrhome到solrhomes中，命令：

cp -r solr /usr/local/solrhomes/solrhome-1

cp -r solr /usr/local/solrhomes/solrhome-2

cp -r solr /usr/local/solrhomes/solrhome-3

cp -r solr /usr/local/solrhomes/solrhome-4

1.6、修改tomcat每个solr的web.xml，关联solrhome：

|  |
| --- |
| <env-entry>  <env-entry-name>solr/home</env-entry-name>  <env-entry-value>g:\solr</env-entry-value>  <env-entry-type>java.lang.String</env-entry-type>  </env-entry> |

将env-entry-value的值写为/usr/local/solrhomes/solrhome-x，x为当前tomcat的id

1.7、修改每个tomcat的端口号，server.xml：

|  |  |  |  |
| --- | --- | --- | --- |
| 原始端口号 | 关闭 8005 | 运行 8080 | 和其他http通信端口8009 |

修改为：8105 8180 8109

8205 8280 8209

8305 8380 8309

8405 8480 8409

1.8、修改tomcat/bin/Catalina.sh：

|  |
| --- |
| #JAVA\_OPTS="$JAVA\_OPTS -Dorg.apache.catalina.security.SecurityListener.UMASK=`umask`" |

（1）去掉#

（2）修改成：

JAVA\_OPTS="-DzkHost=192.168.136.130:2181,192.168.136.130:2182,192.168.136.130:2183"

让tomcat能找到zookeeper集群

1.9、修改solrhome/solr.xml：

|  |
| --- |
| <solrcloud>  <str name="host">${host:}</str>  <int name="hostPort">${jetty.port:8983}</int>  <str name="hostContext">${hostContext:solr}</str>  <int name="zkClientTimeout">${zkClientTimeout:30000}</int>  <bool name="genericCoreNodeNames">${genericCoreNodeNames:true}</bool>  </solrcloud> |

将host修改成192.168.136.130

hostPost修改成tomcat对应的运行端口，如：8180

1.10、由zookeeper统一管理配置文件，工具名：zkcli.sh,该文件在solr源码包里。

（1）将solr压缩包上传到服务器上，并解压

（2）进入solr-4.10.3/example/scripts/cloud-scripts/，执行一次：

./zkcli.sh -zkhost 192.168.136.130:2181,192.168.136.130:2182,192.168.136.130:2183 -cmd upconfig -confdir /usr/local/solrhomes/solrhome-1/collection1/conf -confname myconf

（3）注：修改配置，需在本地先配置，然后再重新执行（2）的命令

参数解释：

-zkhost：指定zookeeper地址列表

-cmd：指定命令，upconfig为上传配置的命令

-confdir：配置文件所在目录

-confname：配置名称

1.11、启动集群：

（1）启动每个tomcat：先看文件夹权限是否已给，chmod -R 777 solr-cloud

1.12、项目配置solr集群：

|  |
| --- |
| <bean id=”solrServer” class=”org.apache.solr.client.solrj.impl.CloudSolrServer”>  <constructor-arg value=”192.168.136.130:2181,192.168.136.130:2182,192.168.136.130:2183”/>  <property name=”defaultCollection” value=”collection1”></property>  </bean>  <bean id=”solrTemplate” class=”org.springframework.data.solr.core.SolrTemplate”>  <constructor-arg ref=”solrServer”/>  </bean> |

|  |
| --- |
| 单机solr：  <solr:solr-server id=”solrServer” url=”http://127.0.0.1:8080/solr”/> |

2.1分片配置

2.1.1 创建新的Collection进行分片处理

在浏览器输入以下地址，可以按照我们的要求创建新的Collection

|  |
| --- |
| http://192.168.136.130:8180/solr/admin/collections?action=CREATE&name=collection2&numShards=2&replicationFactor=2 |

看到success表示成功

参数：

name:将被创建的集合的名字

numShards：集合创建时需要创建逻辑碎片的个数

replicationFactor：分片的副本数

2.1.2 删除不用的Collection，执行以下命令：

|  |
| --- |
| http://192.168.136.130:8180/solr/admin/collections?action=DELETE&name=collection1 |

三、Redis集群

（此步骤为伪集群，真集群时不需要修改端口号，只要放开集群开关即可）

1、安装gcc，命令：因为Redis是C语言开发的，需要C语言的编译环境

yum install gcc-c++

2、安装ruby，需要ruby脚本来实现集群搭建

yum install ruby

yum install rubygems

3、将redis-3.0.0.tar.gz放到服务器上并解压

tar zxvf redis-3.0.0.tar.gz

4、进入redis-3.0.0进行编译，命令：

make

5、创建目录/usr/local/redis-cluster，在redis源码目录上执行命令安装6个redis实例，命令：

mkdir /usr/local/redis-cluster

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

make install PREFIX=/usr/local/redis-cluster/redis-2

make install PREFIX=/usr/local/redis-cluster/redis-3

make install PREFIX=/usr/local/redis-cluster/redis-4

make install PREFIX=/usr/local/redis-cluster/redis-5

make install PREFIX=/usr/local/redis-cluster/redis-6

6、把源码包中的配置文件复制到每个redis下的bin目录中：

cp redis.conf /usr/local/redis-cluster/redis-1/bin

cp redis.conf /usr/local/redis-cluster/redis-2/bin

cp redis.conf /usr/local/redis-cluster/redis-3/bin

cp redis.conf /usr/local/redis-cluster/redis-4/bin

cp redis.conf /usr/local/redis-cluster/redis-5/bin

cp redis.conf /usr/local/redis-cluster/redis-6/bin

7、修改每个redis节点的配置文件redis.conf中的端口号：

|  |
| --- |
| port 6379 45行 |

改为7001-7006

|  |
| --- |
| # cluster-enabled yes 632行 |

把#去掉，开放集群

8、启动每个redis实例，要指定每个启动的配置文件

./redis-server redis.conf

9、所有redis节点启动后，上传redis-3.0.0.gem,该脚本的作用是把所有的节点连接起来

gem install redis-3.0.0.gem

10、在/redis-3.0.0/src中，执行redis-trib.rb：

./redis-trib.rb create --replicas 1 192.168.136.130:7001 192.168.136.130:7002 192.168.136.130:7003 192.168.136.130:7004 192.168.136.130:7005 192.168.136.130:7006

11、redis客户端连接集群命令：

redis-cli -h 192.168.136.130 -p 7001 -c

参数：

-c是连接集群

12、SpringDataRedis连接Redis集群

修改品优购工程 在pinyougou-common工程添加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> |

添加属性文件redis-cluster-config.properties

|  |
| --- |
| #cluster configuration  redis.host1=192.168.25.140  redis.port1=7001  redis.host2=192.168.25.140  redis.port2=7002  redis.host3=192.168.25.140  redis.port3=7003  redis.host4=192.168.25.140  redis.port4=7004  redis.host5=192.168.25.140  redis.port5=7005  redis.host6=192.168.25.140  redis.port6=7006  redis.maxRedirects=3  redis.maxIdle=100  redis.maxTotal=600 |

13、关闭redis节点：

./redis-cli -h IP地址 -p 端口 shutdown