Kafka集群环境搭建

## 1 JDK安装

系统环境：centos-6.4  
安装方式：rpm安装   
软件：jdk-8u11-linux-x64.rpm  
下载地址：http://blog.csdn.net/doupeihua/article/details/51831947

检验系统原版本

[root@admin ~]# java -version  
java version "1.6.0\_24"  
OpenJDK Runtime Environment (IcedTea6 1.11.1) (rhel-1.45.1.11.1.el6-x86\_64)  
OpenJDK 64-Bit Server VM (build 20.0-b12, mixed mode)

进一步查看JDK信息：

|  |
| --- |
| [root@admin ~]# rpm -qa | grep java tzdata-java-2012c-1.el6.noarch java-1.6.0-openjdk-1.6.0.0-1.45.1.11.1.el6.x86\_64 |

卸载OpenJDK，执行以下操作：

|  |
| --- |
| [root@admin ~]# rpm -e --nodeps tzdata-java-2012c-1.el6.noarch [root@admin ~]# rpm -e --nodeps java-1.6.0-openjdk-1.6.0.0-1.45.1.11.1.el6.x86\_64 |

安装JDK

上传新的jdk-8u11-linux-x64.rpm软件到/usr/local/执行以下操作：

|  |
| --- |
| [root@admin local]# rpm -ivh jdk-8u11-linux-x64.rpm |

JDK默认安装在/usr/java中。

验证安装

执行以下操作，查看信息是否正常：

|  |
| --- |
| [root@admin bin]# java [root@admin bin]# java -version  java version "1.8.0\_11"  Java(TM) SE Runtime Environment (build 1.8.0\_11-b12)  Java HotSpot(TM) 64-Bit Server VM (build 25.11-b03, mixed mode) |

恭喜，安装成功！

配置环境变量

我的机器安装完jdk-7-linux-x64.rpm后不用配置环境变量也可以正常执行javac、java –version操作，因此我没有进行JDK环境变量的配置。但是为了以后的不适之需，这里还是记录一下怎么进行配置，操作如下：   
修改系统环境变量文件

|  |
| --- |
| vi /etc/profile |

向文件里面追加以下内容：

|  |
| --- |
| JAVA\_HOME=/usr/java/jdk1.8.0\_11 JRE\_HOME=/usr/java/jdk1.8.0\_11/jre PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib export JAVA\_HOME JRE\_HOME PATH CLASSPATH |

使修改生效

|  |
| --- |
| [root@admin local]# source etc/profile   //使修改立即生效 |

查看系统环境状态

|  |
| --- |
| [root@admin ~]# echo $JAVA\_HOME  /usr/java/jdk1.8.0\_11  [root@admin ~]# echo $PATH //查看PATH值  .:.:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/root/bin:/usr/local/go/bin:/usr/local/codis/bin:/bin:/root/bin:/usr/local/go/bin:/usr/local/codis/bin:/bin:/usr/java/jdk1.8.0\_11/bin:/usr/java/jdk1.8.0\_11/jre/bin |

## 2 zookeeper安装

准备工作：下载好zookeeper3.4.8文件<http://apache.fayea.com/zookeeper/>  
IMG_256

将zookeeper-3.4.8.tar.gz安装文件移至虚拟机或linux操作系统下

解压文件

tar zxvf zookeeper-3.4.8.tar.gz -C /usr/local/zookeeper

配置一台机器上的zookeeper配置文件

先在一个节点机器上配置   
 切换到zookeeper目录下的conf子目录下创建zoo.cfg文件   
 zoo\_sample.cfg是框架自带的示例文件，我在这里直接将 zoo\_sample.cfg文件名修改成zoo.cfg，然后加以修改。

准备3台机器

假设有三台机器，hostname和ip对应关系是(hosts文件配置)：

192.168.100.123 zk01

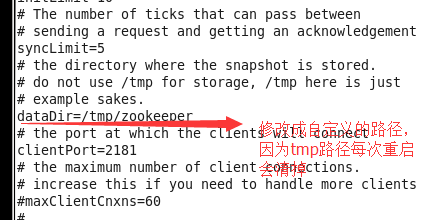
192.168.100.128 zk02

192.168.100.129 zk03

ZooKeeper不存在明显的master/slave关系，各个节点都是服务器，leader挂了，会立马从follower中选举一个出来作为leader.

由于没有主从关系，也不用配置SSH无密码登录了，各个zk服务器是自己启动的，互相之间通过TCP端口来交换数据。

修改配置文件zoo.cfg



改为：

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/usr/local/zookeeper/zookeeper-3.4.8/data

clientPort=2181

server.1=zk01:2888:3888

server.2=zk02:2888:3888

server.3=zk03:2888:3888

c、在zookeeper主目录下新建指定的data目录   
mkdir data   
cd data/   
touch myid   
echo 1 > myid

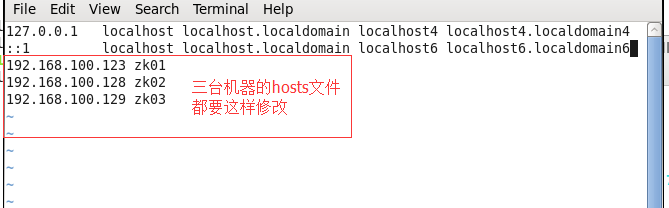
d、将配置好的zookeeper文件夹整个拷贝到其他机器上   
scp -r /usr/local/zookeeper/zookeeper-3.4.8 root@/ 192.168.100.128:/usr/local/zookeeper/  
scp -r /usr/local/zookeeper/zookeeper-3.4.8 root@// 192.168.100.129:/usr/local/zookeeper/

e、修改zk02和zk03上的myid文件

cd data/   
 touch myid   
 echo 2 > myid

cd data/   
 touch myid   
 echo 3 > myid

f、配置主机名和IP地址映射   
vi /etc/hosts



g、到此为止zookeeper集群就搭建完成了，因为修改了hosts文件，要想生效需要重启机器，这一步可以放在最开始做，我是因为开始忘记了，才放到后边实现的。   
重启成功后，我们只需要分别切换到zookeeper/bin目录下执行脚本本件zkServer.sh就可以启动集群   
./zkServer.sh start 分别启动三台机器上的zookeeper.

[root@zk01 bin]# ./zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/zookeeper-3.4.8/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

查看状态 ./zkServer.sh status

[root@zk02 bin]# ./zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/zookeeper-3.4.8/bin/../conf/zoo.cfg

Mode: leader

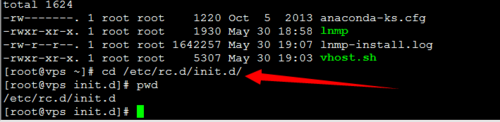
f.使用java客户端连接ZooKeeper集群

找一台机器，解压zookeeper压缩包，不用配置，就可以使用[Java](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "http://blog.csdn.net/pdw2009/article/details/_blank)客户端连接ZooKeeper集群中的任意一台服务器了。对于客户端来说，ZooKeeper是一个整体（ensemble），连接到ZooKeeper集群实际上感觉在独享整个集群的服务，所以，你可以在任何一个结点上建立到服务集群的连接。

[root@zk01 bin]# ./zkCli.sh -server zk02:2181

g.配置Zookeeper开机启

用cd 命令切换到/etc/rc.d/init.d/目录下

[](http://jingyan.baidu.com/album/fec7a1e5136c571190b4e720.html?picindex=2" \t "http://jingyan.baidu.com/article/_self)

接着用touch zookeeper创建一个文件

[IMG_257](http://jingyan.baidu.com/album/fec7a1e5136c571190b4e720.html?picindex=3" \t "http://jingyan.baidu.com/article/_self)

然后为这个文件添加可执行权限

chmod +x zookeeper

[IMG_258](http://jingyan.baidu.com/album/fec7a1e5136c571190b4e720.html?picindex=4" \t "http://jingyan.baidu.com/article/_self)

接着用vi zookeeper来编辑这个文件

[IMG_259](http://jingyan.baidu.com/album/fec7a1e5136c571190b4e720.html?picindex=5" \t "http://jingyan.baidu.com/article/_self)

接着在zookeeper里面输入如下内容

#!/bin/bash

#chkconfig:2345 20 90

#description:zookeeper

#processname:zookeeper

case $1 in

          start) su root /usr/local/zookeeper/zookeeper-3.4.8/bin/zkServer.sh start-foreground &;;

          stop) su root /usr/local/zookeeper/zookeeper-3.4.8/bin/zkServer.sh stop;;

          status) su root /usr/local/zookeeper/zookeeper-3.4.8/bin/zkServer.sh status;;

          restart) su root /usr/local/zookeeper/zookeeper-3.4.8/bin/zkServer.sh restart;;

          \*)  echo "require start|stop|status|restart" ;;

esac

这个时候我们就可以用service zookeeper start/stop来启动停止zookeeper服务了

最后一点我们需要开机自动启动

所以需要添加到启动里面

使用chkconfig --add zookeeper命令吧zookeeper添加到开机启动里面

添加完成之后接这个使用chkconfig --list 来看看我们添加的zookeeper是否在里面

## 3 kafka安装

进入下载页面：[http://kafka.apache.org/downloads.html](http://kafka.apache.org/downloads.html" \t "http://czj4451.iteye.com/blog/_blank)   
  
选择Binary downloads下载 （Source download需要编译才能使用）

软件环境

1、linux一台或多台，大于等于2

2、已经搭建好的zookeeper集群

3、软件版本kafka\_2.11-0.10.1.0.tgz

创建目录并下载安装软件

#创建目录

cd /usr/local

mkdir kafka #创建项目目录cd kafka

mkdir kafkalogs #创建kafka消息目录，主要存放kafka消息

#下载软件

wget http://apache.opencas.org/kafka/0.10.1.0/kafka\_2.11-0.10.1.0.tgz

#解压软件

tar -zxvf kafka\_2.11-0.10.1.0.tgz

修改配置文件

进入到config目录

cd /usr/local/kafka/kafka\_2.11-0.9.0.1/config/

主要关注：server.properties 这个文件即可，我们可以发现在目录下：

有很多文件，这里可以发现有Zookeeper文件，我们可以根据Kafka内带的zk集群来启动，但是建议使用独立的zk集群.

修改配置文件:

broker.id=0 #当前机器在集群中的唯一标识，和zookeeper的myid性质一样

port=19092 #当前kafka对外提供服务的端口默认是9092

host.name=192.168.7.100 #这个参数默认是关闭的，在0.8.1有个bug，DNS解析问题，失败率的问题。

num.network.threads=3 #这个是borker进行网络处理的线程数

num.io.threads=8 #这个是borker进行I/O处理的线程数

log.dirs=/opt/kafka/kafkalogs/ #消息存放的目录，这个目录可以配置为“，”逗号分割的表达式，上面的num.io.threads要大于这个目录的个数这个目录，如果配置多个目录，新创建的topic他把消息持久化的地方是，当前以逗号分割的目录中，那个分区数最少就放那一个

socket.send.buffer.bytes=102400 #发送缓冲区buffer大小，数据不是一下子就发送的，先回存储到缓冲区了到达一定的大小后在发送，能提高性能

socket.receive.buffer.bytes=102400 #kafka接收缓冲区大小，当数据到达一定大小后在序列化到磁盘

socket.request.max.bytes=104857600 #这个参数是向kafka请求消息或者向kafka发送消息的请请求的最大数，这个值不能超过java的堆栈大小

num.partitions=1 #默认的分区数，一个topic默认1个分区数

log.retention.hours=168 #默认消息的最大持久化时间，168小时，7天

message.max.byte=5242880 #消息保存的最大值5M

default.replication.factor=2 #kafka保存消息的副本数，如果一个副本失效了，另一个还可以继续提供服务

replica.fetch.max.bytes=5242880 #取消息的最大直接数

log.segment.bytes=1073741824 #这个参数是：因为kafka的消息是以追加的形式落地到文件，当超过这个值的时候，kafka会新起一个文件

log.retention.check.interval.ms=300000 #每隔300000毫秒去检查上面配置的log失效时间（log.retention.hours=168 ），到目录查看是否有过期的消息如果有，删除

log.cleaner.enable=false #是否启用log压缩，一般不用启用，启用的话可以提高性能

zookeeper.connect=zk01:2181,zk02:2181,1zk03:2181#设置zookeeper的连接端口

[IMG_257](http://www.cnblogs.com/luotianshuai/p/javascript:void(0);)

上面是参数的解释，实际的修改项为：

[IMG_258](http://www.cnblogs.com/luotianshuai/p/javascript:void(0);)

#broker.id=1 每台服务器的broker.id都不能相同

#hostname

host.name=zk01

#在log.retention.hours=168 下面新增下面三项

message.max.byte=5242880

default.replication.factor=2

replica.fetch.max.bytes=5242880

#设置zookeeper的连接端口

zookeeper.connect=zk01:2181,zk02:2181,1zk03:2181

注：这里配置broker的时候，每台机器上的broker保证唯一，从1开始。如：在另外2台机器上分别配置broker.id=2，broker.id=3

[IMG_259](http://www.cnblogs.com/luotianshuai/p/javascript:void(0);)

配置环境变量

vi /etc/profile

export KAFKA\_HOME=/usr/local/kafka/kafka\_2.11-0.10.1.0

export PATH=$PATH:$KAFKA\_HOME/bin

配置Kafka的zookeeper.properties

# the directory where the snapshot is stored.

dataDir=/usr/local/zookeeper/zookeeper-3.4.8/data

# the port at which the clients will connect

clientPort=2181

# disable the per-ip limit on the number of connections since this is a non-production config

maxClientCnxns=0

配置producer.properties

# list of brokers used for bootstrapping knowledge about the rest of the cluster

# format: host1:port1,host2:port2 ...

bootstrap.servers=zk01:9092,zk02:9092

配置consumer.properties

# Zookeeper connection string

# comma separated host:port pairs, each corresponding to a zk

# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002"

zookeeper.connect=zk01:2181,zk02:2181,zk03:2181

启动Kafka集群并测试

1、启动服务

#从后台启动Kafka集群（3台都需要启动）

cd /usr/local/kafka/kafka\_2.11-0.10.0.1/bin #进入到kafka的bin目录   
./kafka-server-start.sh - daemon ../config/server.properties &

2、检查服务是否启动

#执行命令jps

20348 Jps4233 QuorumPeerMain18991

Kafka

3、创建Topic来验证是否创建成功

更多请看官方文档：http://kafka.apache.org/documentation.html

[IMG_256](http://www.cnblogs.com/luotianshuai/p/javascript:void(0);)

#创建Topic

./kafka-topics.sh --create --zookeeper zk01:2181 --replication-factor 2 --partitions 1 --topic shuaige#解释

--replication-factor 2 #复制两份

--partitions 1 #创建1个分区

--topic #主题为shuaige

'''在一台服务器上创建一个发布者'''#创建一个broker，发布者

./kafka-console-producer.sh --broker-list zk01:9092 --topic shuaige

'''在一台服务器上创建一个订阅者'''

./kafka-console-consumer.sh --bootstrap-server zk02:9092--topic shuaige --from-beginning

[IMG_257](http://www.cnblogs.com/luotianshuai/p/javascript:void(0);)

测试（在发布者那里发布消息看看订阅者那里是否能正常收到~）：

查看该Topic的相关信息，命令如下所示：

kafka-topics.sh --zookeeper zk01:2181,zk02:2181 --topic shuaige --describe

topic:shuaige PartitionCount:1 ReplicationFactor:2 Configs:

Topic: shuaige Partition: 0 Leader: 1 Replicas: 2,1 Isr: 1,2

下面解释一下这些输出。第一行是对所有分区的一个描述，然后每个分区都会对应一行，因为我们只有一个分区所以下面就只加了一行。

Leader：负责处理消息的读和写，Leader是从所有节点中随机选择的。

Replicas：列出了所有的副本节点，不管节点是否在服务中。

Isr：是正在服务中的节点

详情请查看官方文档

<http://kafka.apache.org/quickstart>

相关命令：

./kafka-topics.sh --create --zookeeper zk01:2181 --replication-factor 3 --partitions 1 --topic test-replicated-topic

./kafka-topics.sh --describe --zookeeper zk01:2181 --topic test-replicated-topic

./kafka-console-producer.sh --broker-list 192.168.100.123:9092,192.168.100.128:9092,192.168.100.129:9092 --topic test-replicated-topic

./kafka-console-consumer.sh --bootstrap-server zk01:9092,zk02:9092,zk03:9092 --from-beginning --topic test-replicated-topic

题外话：

在真正的应用环境中，Consumer端将消息拉取下来后要做的肯定不止是输出出来这么简单，在消费消息时很有可能需要花掉更多的时间。1个Consumer线程消费消息的速度很有可能是赶不上Producer产生消息的速度，所以我们不得不考虑Consumer端采用多线程模型来消费消息。  
然而KafkaConsumer并不是线程安全的，多个线程操作同一个KafkaConsumer实例会出现各种问题，Kafka官方对于Consumer端的多线程处理给出的指导建议如下：  
1. 每个线程都持有一个KafkaConsumer对象  
好处：

实现简单

不需要线程间的协作，效率最高

最容易实现每个Partition内消息的顺序处理

弊端：

每个KafkaConsumer都要与集群保持一个TCP连接

线程数不能超过Partition数

每一batch拉取的数据量会变小，对吞吐量有一定影响

2. 解耦，1个Consumer线程负责拉取消息，数个Worker线程负责消费消息  
好处：

可自由控制Worker线程的数量，不受Partition数量限制

弊端：

消息消费的顺序无法保证

难以控制手动提交offset的时机

个人认为第二种方式更加可取，consumer数不能超过partition数这个限制是很要命的，不可能为了提高Consumer消费消息的效率而把Topic分成更多的partition，partition越多，集群的高可用性就越低。