Kafka集群搭建

Kafka集群搭建步骤

之前的已经介绍了Kafka相关的基本概念，现在开始搭建一个Kafka集群，集群是把状态保存在Zookeeper中的，所以在安装Kafka之前要搭建一个Zookeeper集群。

1、软件环境

主机名	IP	安装的软件
myj04	192.168.2.84	Jdk、Zookeeper、Kafka
myj05	192.168.2.85	Jdk、Zookeeper、Kafka
myj06	192.168.2.86	Jdk、Zookeeper、Kafka

2、安装Zookeeper

先选定一台机器安装zookeeper，安装后直接拷贝到其他两台机器。

2.1、解压zookeeper

tar -zxvf /home/tar/zookeeper-3.4.8.tar.gz -C /home/cluster/

2.2、创建配置zoo.cfg并修改

切换到zookeeper的conf目录下，复制zoo_sample.cfg并重命名为zoo.cfg

```
cd /home/cluster/zookeeper-3.4.8/conf/
cp zoo_sample.cfg  zoo.cfg
```

编辑zoo.cfg配置中的内容，将dataDir的目录改为（需提前创建data目录）

```
dataDir=/home/cluster/zookeeper-3.4.8/data
```

并在文件末尾添加如下内容

```
server.1=myj04:2888:3888
server.2=myj05:2888:3888
server.3=myj06:2888:3888
```

```
[root@myj06 conf]# more zoo.cfg
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=/home/cluster/zookeeper-3.4.8/data
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1


server.1=myj04:2888:3888
server.2=myj05:2888:3888
server.3=myj06:2888:3888
```

2.3、创建文件myid

切换到/home/cluster/zookeeper-3.4.8/data目录，创建文件myid，在文件中输入1后保存退出（1表示server.1=myj04:2888:3888 中的1）

cd /home/cluster/zookeeper-3.4.8/data
touch myid
vim myid

```
[root@myj04 data]# more myid
1
```

2.4、拷贝zookeeper到另外两台几点

scp -r /home/cluster/zookeeper-3.4.8/ root@myj05:/home/cluster/

scp -r /home/cluster/zookeeper-3.4.8/ root@myj06:/home/cluster/

2.5、修改这两台节点中的myid文件

分别修改myj05、myj06上的/home/cluster/zookeeper-3.4.8/data下myid的内容分别为2和3

vim myid

```
[root@myj05 ~]# more /home/cluster/zookeeper-3.4.8/data/myid
2

[root@myj06 data]#  more /home/cluster/zookeeper-3.4.8/data/myid
3
```

2.5、启动zookeeper

分别在三台机器上执行如下命令

cd /home/cluster/zookeeper-3.4.8/bin
./zkServer.sh start

```
[root@myj04 data]# cd   /home/cluster/zookeeper-3.4.8/bin
[root@myj04 bin]# ./zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /home/cluster/zookeeper-3.4.8/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

2.5、验证zookeeper是否启动成功

验证zookeeper启动是否成功可以使用命令查看zookeeper的状态，如果启动成功，其中两台机器为follower角色，一台为leader角色

/home/cluster/zookeeper-3.4.8/bin/zkServer.sh status

```
[root@myj06 bin]# /home/cluster/zookeeper-3.4.8/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /home/cluster/zookeeper-3.4.8/bin/../conf/zoo.cfg
Mode: leader
```

```
[root@myj05 bin]# /home/cluster/zookeeper-3.4.8/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /home/cluster/zookeeper-3.4.8/bin/../conf/zoo.cfg
Mode: follower
```

```
[root@myj04 bin]# /home/cluster/zookeeper-3.4.8/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /home/cluster/zookeeper-3.4.8/bin/../conf/zoo.cfg
Mode: follower
```

3、安装Kafka

安装zookeeper成功后开始安装Kafka，同样先选定一台机器安装后直接拷贝到其他两台机器。

3.1、解压Kafka

tar -zxf /home/tar/kafka_2.11-1.0.0.tgz -C /home/cluster/

3.2、修改配置文件server.properties

vim /home/cluster/kafka_2.11-1.0.0/config/server.properties

修改后的配置文件如下

```
[root@myj04 config]# more server.properties
broker.id=1
port=9092
host.name=myj04
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
log.dirs=/home/cluster/kafka_2.11-1.0.0/kafka-logs
num.partitions=1
default.replication.factor=2
num.recovery.threads.per.data.dir=1
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
zookeeper.connect=myj04:2181,myj05:2181,myj06:2181
zookeeper.connection.timeout.ms=6000
[root@myj04 config]#
```

其中我们主要修改的参数如下：
broker.id=1
port=9092
host.name=myj04
log.dirs=/home/cluster/kafka_2.11-1.0.0/kafka-logs
zookeeper.connect=myj04:2181,myj05:2181,myj06:2181

配置文件解释如下：

#当前机器在集群中的唯一标识，和zookeeper的myid性质一样
broker.id=1
#当前kafka对外提供服务的端口默认是9092
port=9092
#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。
host.name=myj04
#这个是borker进行网络处理的线程数
num.network.threads=3
#这个是borker进行I/O处理的线程数
num.io.threads=8
#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.send.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.receive.buffer.bytes=102400
#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
socket.request.max.bytes=104857600
#消息存放的目录
log.dirs=/home/cluster/kafka_2.11-1.0.0/kafka-logs
#默认的分区数，一个topic默认1个分区数
num.partitions=1
#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
default.replication.factor=2
num.recovery.threads.per.data.dir=1
#默认消息的最大持久化时间，168小时，7天
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
#设置zookeeper的连接端口

zookeeper.connect=myj04:2181,myj05:2181,myj06:2181
zookeeper.connection.timeout.ms=6000

3.3、拷贝Kafka到另外两台几点

scp -r /home/cluster/kafka_2.11-1.0.0/ root@myj05:/home/cluster/

scp -r /home/cluster/kafka_2.11-1.0.0/ root@myj06:/home/cluster/

需要注意的点是分别修改这两台机器中的配置文件server.properties 中的broker.id的值分别配置为2、3，host.name的值分别配置myj05、myj06，log.dirs必须保证目录存在，修改后的配置文件如下：

myj05的配置文件：

```
[root@myj05 kafka_2.11-1.0.0]# more config/server.properties
broker.id=2
port=9092
host.name=myj05
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
log.dirs=/home/cluster/kafka_2.11-1.0.0/kafka-logs
num.partitions=1
default.replication.factor=2
num.recovery.threads.per.data.dir=1
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
zookeeper.connect=myj04:2181,myj05:2181,myj06:2181
zookeeper.connection.timeout.ms=6000
```

myj06的配置文件：

```
[root@myj06 kafka_2.11-1.0.0]# more config/server.properties
broker.id=3
port=9092
host.name=myj06
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
log.dirs=/home/cluster/kafka_2.11-1.0.0/kafka-logs
num.partitions=1
default.replication.factor=2
num.recovery.threads.per.data.dir=1
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
zookeeper.connect=myj04:2181,myj05:2181,myj06:2181
zookeeper.connection.timeout.ms=6000
```

3.4、启动Kafka

启动kafka集群，进入kafka目录，执行如下命令 ：

cd /home/cluster/kafka_2.11-1.0.0/

./bin/kafka-server-start.sh -daemon config/server.properties &

三个节点均要启动，启动无报错，即搭建成功。

```
[root@myj04 kafka_2.11-1.0.0]# jps
34961 Kafka
35044 Jps
32519 QuorumPeerMain
```

3.5、验证是否启动成功

#创建Topic

/home/cluster/kafka_2.11-1.0.0/bin/kafka-topics.sh --create --zookeeper myj04:2181 --replication-factor 2 --partitions 1 --topic testCreateTopic

#上述命令解释

--replication-factor 2 #复制两份

--partitions 1 #创建1个分区

--topic #主题为testCreateTopic

#查看Topic list

/home/cluster/kafka_2.11-1.0.0/bin/kafka-topics.sh --list --zookeeper myj04:2181,myj05:2181,myj05:2181

#在一台机器上创建一个发布者

/home/cluster/kafka_2.11-1.0.0/bin/kafka-console-producer.sh --broker-list myj04:9092,myj05:9092,myj06:9092 --topic testCreateTopic

发送消息如下：

```
[root@myj04 config]# /home/cluster/kafka_2.11-1.0.0/bin/kafka-console-producer.sh --broker-list myj04:9092,myj05:9092,myj05:9092 --topic testCreateTopic
>1
>2
>3
>4
>5
>6
>7
>
```

#在一台服务器上创建一个订阅者

/home/cluster/kafka_2.11-1.0.0/bin/kafka-console-consumer.sh --zookeeper myj04:2181,myj05:2181,myj06:2181 --topic testCreateTopic --from-beginning

解释消息如下图

```
[root@myj05 config]# /home/cluster/kafka_2.11-1.0.0/bin/kafka-console-consumer.sh --zookeeper myj04:2181,myj05:2181,myj05:2181 --topic testCreateTopic --from-beginning
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].
1
2
3
4
5
6
7
```

4、Java操作Kafka

上面已在控制台中演示了向Kafka集群发送消息和从Kafka中消费消息，下面用Java演示如何向Kafka主题发送消息和如何订阅Kafka主题中的消息。

首先在项目中加入Kafka的Maven依赖

```
<dependency>
 <groupId>org.apache.kafka</groupId>
 <artifactId>kafka-clients</artifactId>
 <version>1.0.0</version>
```

```
</dependency>
```

4.1、发布消息

```
package com.myj.test.producer;

import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.Properties;

public class TestProducer {

    public static void main(String[] args) {

        Properties props = new Properties();
        props.put("bootstrap.servers","myj04:9092,myj05:9092,myj06:9092");//该地址是集群的子集，用来探测集群。
        props.put("acks", "all");// 记录完整提交，最慢的但是最大可能的持久化
        props.put("retries", 3);// 请求失败重试的次数
        props.put("batch.size", 16384);// batch的大小
        props.put("linger.ms", 1);// 默认情况即使缓冲区有剩余的空间，也会立即发送请求，设置一段时间用来等待从而将缓冲区填的更多，单位为毫秒，producer发送数据会延迟1ms，可以减少发送到kafka服务器的请求数据
        props.put("buffer.memory", 33554432);// 提供给生产者缓冲内存总量
        props.put("key.serializer","org.apache.kafka.common.serialization.StringSerializer");// 序列化的方式，ByteArraySerializer或者StringSerializer
        props.put("value.serializer","org.apache.kafka.common.serialization.StringSerializer");
        KafkaProducer<String, String> kafkaProducer = new KafkaProducer<>(props);

        for (int i = 1; i <= 10; i++) {
            // 三个参数分别为topic, key,value，send()是异步的，添加到缓冲区立即返回，更高效。
            kafkaProducer.send(new ProducerRecord<String, String>("testCreateTopic","key-"+Integer.toString(i), "value-"+Integer.toString(i)));
        }
        kafkaProducer.close();

    }

}
```

4.2、订阅消息

```
package com.myj.test.consumer;

import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;

import java.util.Arrays;
import java.util.Properties;

public class TestConsumer {

    public static void main(String[] args) {
        Properties props = new Properties();
        props.put("bootstrap.servers","myj04:9092,myj05:9092,myj06:9092");//该地址是集群的子集，用来探测集群。
```

```
                props.put("group.id", "group1");// cousumer的分组id
                props.put("enable.auto.commit", "true");// 自动提交offsets
                props.put("auto.commit.interval.ms", "1000");// 每隔1s，自动提交offsets
                props.put("session.timeout.ms", "30000");// Consumer向集群发送自己的心跳，超时则认为Consumer已经死了，kafka会把它的分区分配给其他进程
                props.put("key.deserializer","org.apache.kafka.common.serialization.StringDeserializer");// 反序列化器
                props.put("value.deserializer","org.apache.kafka.common.serialization.StringDeserializer");
                KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);
                consumer.subscribe(Arrays.asList("testCreateTopic"));// 订阅的topic,可以多个
                while (true) {
                    ConsumerRecords<String, String> records = consumer.poll(100);
                    for (ConsumerRecord<String, String> record : records) {
                        System.out.printf("offset = %d, key = %s, value = %s",
                                record.offset(), record.key(), record.value());
                        System.out.println();
                    }
                }
            }

        }
```