[Kafka集群搭建](http://www.cnblogs.com/luotianshuai/p/5206662.html)

**Kafka初识**

**1、Kafka使用背景**

在我们大量使用分布式数据库、分布式计算集群的时候，是否会遇到这样的一些问题：

1. 我们想分析下用户行为（pageviews），以便我们设计出更好的广告位
2. 我想对用户的搜索关键词进行统计，分析出当前的流行趋势
3. 有些数据，存储数据库浪费，直接存储硬盘效率又低

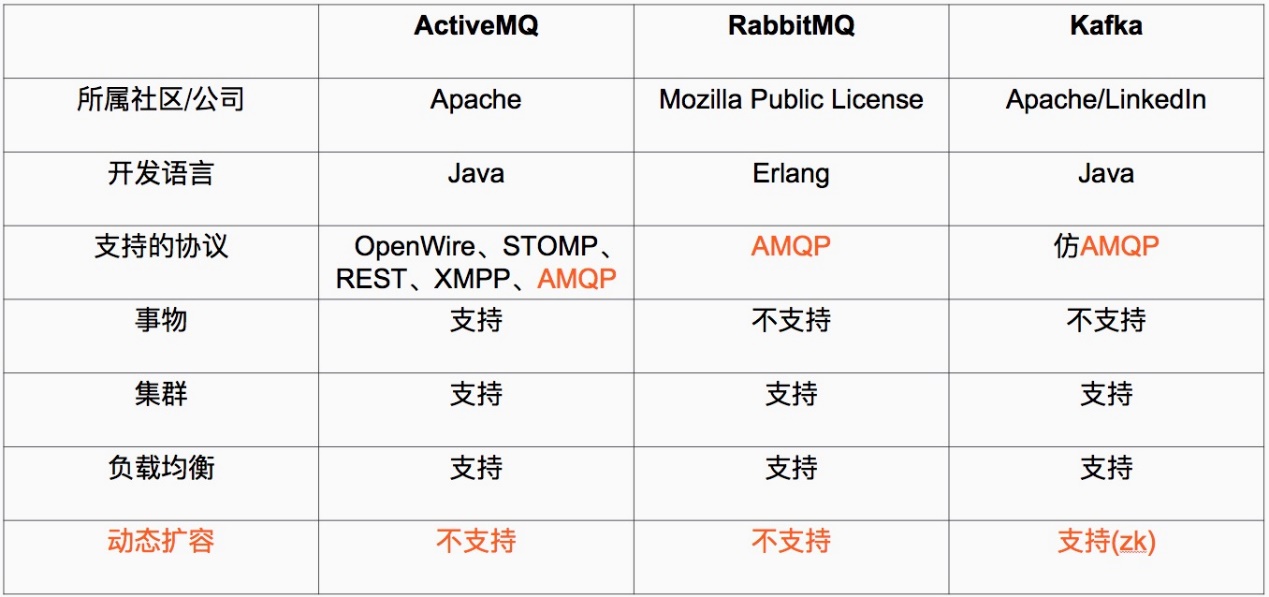
这些场景都有一个共同点：

数据是由上游模块产生，使用上游模块的数据计算、统计、分析，这个时候就可以使用消息系统，尤其是分布式消息系统！

**2、Kafka的定义**

What is Kafka：它是一个分布式消息系统，由linkedin使用scala编写，用作LinkedIn的活动流（Activity Stream）和运营数据处理管道（Pipeline）的基础。具有高水平扩展和高吞吐量。

**3、Kafka和其他主流分布式消息系统的对比**



定义解释：

1、Java 和 scala都是运行在JVM上的语言。

2、erlang和最近比较火的go语言一样是从代码级别就支持高并发的一种语言，所以RabbitMQ天生就有很高的并发性能，但是 有RabbitMQ严格按照AMQP进行实现，受到了很多限制。kafka的设计目标是高吞吐量，所以kafka自己设计了一套高性能但是不通用的协议，他也是仿照AMQP（ Advanced Message Queuing Protocol   高级消息队列协议）设计的。

3、事物的概念：在数据库中，多个操作一起提交，要么操作全部成功，要么全部失败。举个例子， 在转账的时候付款和收款，就是一个事物的例子，你给一个人转账，你转成功，并且对方正常行收到款项后，这个操作才算成功，有一方失败，那么这个操作就是失败的。

对应在消息队列中，就是多条消息一起发送，要么全部成功，要么全部失败。3个中只有ActiveMQ支持，这个是因为，RabbitMQ和Kafka为了更高的性能，而放弃了对事物的支持 。

4、集群：多台服务器组成的整体叫做集群，这个整体对生产者和消费者来说，是透明的。其实对消费系统组成的集群添加一台服务器减少一台服务器对生产者和消费者都是无感之的。

5、负载均衡，对消息系统来说负载均衡是大量的生产者和消费者向消息系统发出请求消息，系统必须均衡这些请求使得每一台服务器的请求达到平衡，而不是大量的请求，落到某一台或几台，使得这几台服务器高负荷或超负荷工作，严重情况下会停止服务或宕机。

6、动态扩容是很多公司要求的技术之一，不支持动态扩容就意味着停止服务，这对很多公司来说是不可以接受的。

注：

阿里巴巴的Metal,RocketMQ都有Kafka的影子，他们要么改造了Kafka或者借鉴了Kafka，最后Kafka的动态扩容是通过Zookeeper来实现的。

Zookeeper是一种在分布式系统中被广泛用来作为：分布式状态管理、分布式协调管理、分布式配置管理、和分布式锁服务的集群。kafka增加和减少服务器都会在Zookeeper节点上触发相应的事件kafka系统会捕获这些事件，进行新一轮的负载均衡，客户端也会捕获这些事件来进行新一轮的处理。

**Kafka相关概念**

**1、 AMQP协议**

Advanced Message Queuing Protocol （高级消息队列协议）

**The Advanced Message Queuing Protocol (AMQP)：**是一个标准开放的应用层的消息中间件（Message Oriented Middleware）协议。AMQP定义了通过网络发送的字节流的数据格式。因此兼容性非常好，任何实现AMQP协议的程序都可以和与AMQP协议兼容的其他程序交互，可以很容易做到跨语言，跨平台。

上面说的3种比较流行的消息队列协议，要么支持AMQP协议，要么借鉴了AMQP协议的思想进行了开发、实现、设计。

**2、 一些基本的概念**

1、消费者：（Consumer）：从消息队列中请求消息的客户端应用程序

2、生产者：（Producer）  ：向broker发布消息的应用程序

3、AMQP服务端（broker）：用来接收生产者发送的消息并将这些消息路由给服务器中的队列，便于fafka将生产者发送的消息，动态的添加到磁盘并给每一条消息一个偏移量，所以对于kafka一个broker就是一个应用程序的实例

kafka支持的客户端语言：Kafka客户端支持当前大部分主流语言，包括：C、C++、Erlang、Java、.net、perl、PHP、Python、Ruby、Go、Javascript

可以使用以上任何一种语言和kafka服务器进行通信（即便是自己的consumer从kafka集群订阅消息也可以自己写producer程序）

**3、Kafka架构**

生产者生产消息、kafka集群、消费者获取消息这样一种架构，如下图：



kafka集群中的消息，是通过Topic（主题）来进行组织的，如下图：



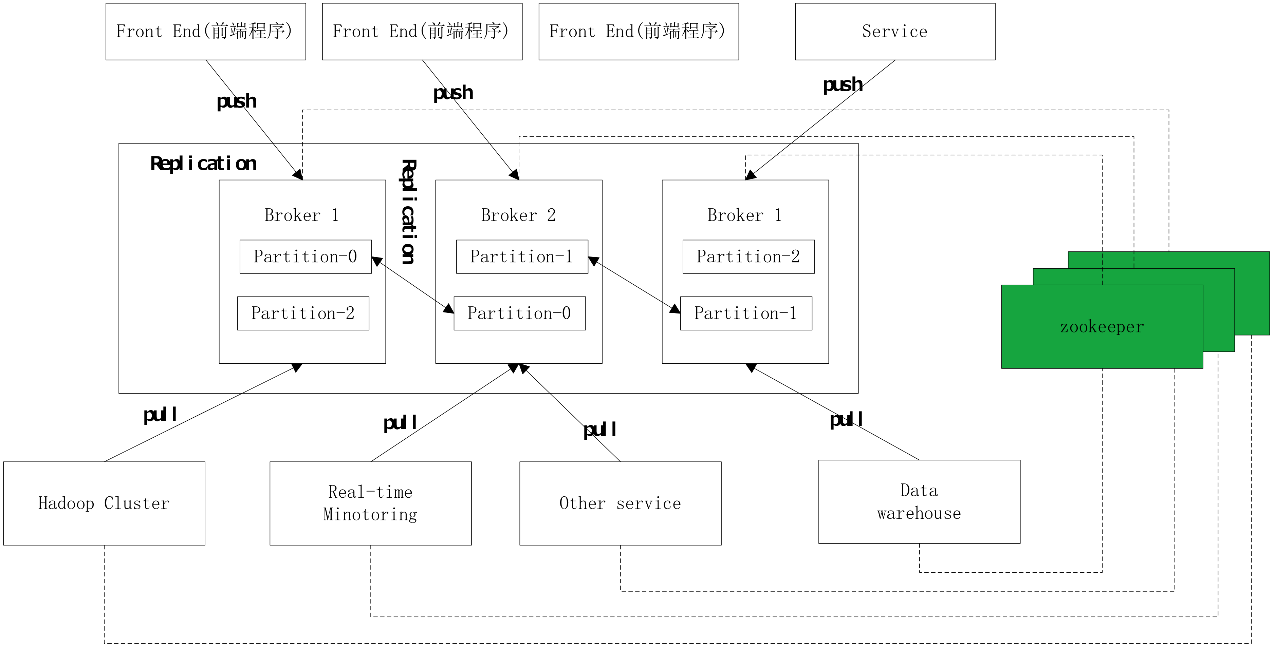
一些基本的概念：

1、主题（Topic）：一个主题类似新闻中的体育、娱乐、教育等分类概念，在实际工程中通常一个业务一个主题。

2、分区（Partition）：一个Topic中的消息数据按照多个分区组织，分区是kafka消息队列组织的最小单位，一个分区可以看作是一个FIFO（ First Input First Output的缩写，先入先出队列）的队列。

kafka分区是提高kafka性能的关键所在，当你发现你的集群性能不高时，常用手段就是增加Topic的分区，分区里面的消息是按照从新到老的顺序进行组织，消费者从队列头订阅消息，生产者从队列尾添加消息。

工作图：



备份（Replication）：为了保证分布式可靠性，kafka0.8开始对每个分区的数据进行备份（不同的Broker上），防止其中一个Broker宕机造成分区上的数据不可用。

kafka0.7是一个很大的改变：1、增加了备份2、增加了控制节点概念，增加了集群领导者选举 。

**Zookeeper集群搭建**

Kafka集群是把状态保存在Zookeeper中的，首先要搭建Zookeeper集群。

**1、软件环境**

（3台服务器-我的测试）

192.168.88.88 master

192.168.88.89 slave1

192.168.88.90 slave2

1、Linux服务器一台、三台、五台、（2\*n+1），Zookeeper集群的工作是超过半数才能对外提供服务，3台中超过两台超过半数，允许1台挂掉 ，是否可以用偶数，其实没必要。

如果有四台那么挂掉一台还剩下三台服务器，如果在挂掉一个就不行了，这里记住是超过半数。

2、Java jdk1.7 zookeeper是用java写的所以他的需要JAVA环境，java是运行在java虚拟机上的

3、Zookeeper的稳定版本Zookeeper 3.4.5版本

**2、配置&安装Zookeeper**

下面的操作是：3台服务器统一操作

1、安装Java

yum list java\*

yum -y install java-1.7.0-openjdk\*

2、下载Zookeeper

首先要注意在生产环境中目录结构要定义好，防止在项目过多的时候找不到所需的项目

#我的目录统一放在/opt/bigdata下面

#首先创建Zookeeper项目目录

mkdir zookeeper #项目目录

mkdir zkdata #存放快照日志

mkdir zkdatalog#存放事物日志

下载Zookeeper

#下载软件

cd /opt/bigdata/zookeeper/

wget http://mirrors.cnnic.cn/apache/zookeeper/zookeeper-3.4.5/zookeeper-3.4.5.tar.gz

#解压软件

tar -zxvf zookeeper-3.4.5.tar.gz

**3、修改配置文件**

进入到解压好的目录里面的conf目录中，查看

#进入conf目录

/opt/bigdata/zookeeper/zookeeper-3.4.5/conf

#查看

[root@192.168.88.90]$ ll

-rw-rw-r--. 1 1000 1000 535 Feb 20 2014 configuration.xsl

-rw-rw-r--. 1 1000 1000 2161 Feb 20 2014 log4j.properties

-rw-rw-r--. 1 1000 1000 922 Feb 20 2014 zoo\_sample.cfg

#zoo\_sample.cfg  这个文件是官方给我们的zookeeper的样板文件，给他复制一份命名为zoo.cfg，zoo.cfg是官方指定的文件命名规则。

**3台服务器的配置文件**

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/opt/bigdata/zookeeper/zkdata

dataLogDir=/opt/bigdata/zookeeper/zkdatalog

clientPort=2181

server.1=192.168.88.88:12888:13888

server.2=192.168.88.89:12888:13888

server.3=192.168.88.90:12888:13888

#server.1 这个1是服务器的标识也可以是其他的数字， 表示这个是第几号服务器，用来标识服务器，这个标识要写到快照目录下面myid文件里

#192.168.88.90为集群里的IP地址，第一个端口是master和slave之间的通信端口，默认是2888，第二个端口是leader选举的端口，集群刚启动的时候选举或者leader挂掉之后进行新的选举的端口默认是3888

**配置文件解释：**

#tickTime：

这个时间是作为 Zookeeper 服务器之间或客户端与服务器之间维持心跳的时间间隔，也就是每个 tickTime 时间就会发送一个心跳。

#initLimit：

这个配置项是用来配置 Zookeeper 接受客户端（这里所说的客户端不是用户连接 Zookeeper 服务器的客户端，而是 Zookeeper 服务器集群中连接到 Leader 的 Follower 服务器）初始化连接时最长能忍受多少个心跳时间间隔数。当已经超过 5个心跳的时间（也就是 tickTime）长度后 Zookeeper 服务器还没有收到客户端的返回信息，那么表明这个客户端连接失败。总的时间长度就是 5\*2000=10 秒

#syncLimit：

这个配置项标识 Leader 与Follower 之间发送消息，请求和应答时间长度，最长不能超过多少个 tickTime 的时间长度，总的时间长度就是5\*2000=10秒

#dataDir：

快照日志的存储路径

#dataLogDir：

事物日志的存储路径，如果不配置这个那么事物日志会默认存储到dataDir制定的目录，这样会严重影响zk的性能，当zk吞吐量较大的时候，产生的事物日志、快照日志太多

#clientPort：

这个端口就是客户端连接 Zookeeper 服务器的端口，Zookeeper 会监听这个端口，接受客户端的访问请求。修改他的端口改大点

创建myid文件

#master

echo "1" > /opt/bigdata/zookeeper/zkdata/myid

#slave1

echo "2" > /opt/bigdata/zookeeper/zkdata/myid

#slave2

echo "3" > /opt/bigdata/zookeeper/zkdata/myid

**4、重要配置说明**

1、myid文件和server.myid  在快照目录下存放的标识本台服务器的文件，他是整个zk集群用来发现彼此的一个重要标识。

2、zoo.cfg 文件是zookeeper配置文件 在conf目录里。

3、log4j.properties文件是zk的日志输出文件 在conf目录里用java写的程序基本上有个共同点日志都用log4j，来进行管理。

configuration for log4j

4、zkEnv.sh和zkServer.sh文件

zkServer.sh 主要的管理程序文件

zkEnv.sh 是主要配置，zookeeper集群启动时配置环境变量的文件

5、还有一个需要注意

ZooKeeper server **will not remove old snapshots and log files** when using the default configuration (see autopurge below), this is the responsibility of the operator

zookeeper不会主动的清除旧的快照和日志文件，这个是操作者的责任。

但是可以通过命令去定期的清理。

#!/bin/bash

#snapshot file dir

dataDir=/opt/bigdata/zookeeper/zkdata/version-2

#tran log dir

dataLogDir=/opt/bigdata/zookeeper/zkdatalog/version-2

#Leave 66 files

count=66

count=$[$count+1]

ls -t $dataLogDir/log.\* | tail -n +$count | xargs rm -f

ls -t $dataDir/snapshot.\* | tail -n +$count | xargs rm -f

#以上这个脚本定义了删除对应两个目录中的文件，保留最新的66个文件，可以将他写到crontab中，设置为每天凌晨2点执行一次就可以了。

#zk log dir del the zookeeper log

#logDir=

#ls -t $logDir/zookeeper.log.\* | tail -n +$count | xargs rm -f

其他方法：

第二种：使用ZK的工具类PurgeTxnLog，它的实现了一种简单的历史文件清理策略，可以在这里看一下他的使用方法 http://zookeeper.apache.org/doc/r3.4.5/zookeeperAdmin.html

第三种：对于上面这个执行，ZK自己已经写好了脚本，在bin/zkCleanup.sh中，所以直接使用这个脚本也是可以执行清理工作的。

第四种：从3.4.0开始，zookeeper提供了自动清理snapshot和事务日志的功能，通过配置 autopurge.snapRetainCount 和 autopurge.purgeInterval 这两个参数能够实现定时清理了。这两个参数都是在zoo.cfg中配置的：

**autopurge.purgeInterval**  这个参数指定了清理频率，单位是小时，需要填写一个1或更大的整数，默认是0，表示不开启自己清理功能。

**autopurge.snapRetainCount** 这个参数和上面的参数搭配使用，这个参数指定了需要保留的文件数目。默认是保留3个。

推荐使用第一种方法，对于运维人员来说，将日志清理工作独立出来，便于统一管理也更可控。毕竟zk自带的一些工具并不怎么给力。

**5、启动服务并查看**

1、启动服务

#进入到Zookeeper的bin目录下

cd /opt/bigdata/zookeeper/zookeeper-3.4.5/bin

#启动服务（3台都需要操作）

./zkServer.sh start

2、检查服务状态

#检查服务器状态

./zkServer.sh status

通过status就能看到状态：

./zkServer.sh status

JMX enabled by default

Using config: /opt/zookeeper/zookeeper-3.4.5/bin/../conf/zoo.cfg #配置文件

Mode: follower #他是否为领导

zk集群一般只有一个leader，多个follower，主一般是相应客户端的读写请求，而从主同步数据，当主挂掉之后就会从follower里投票选举一个leader出来。

可以用“jps”查看zk的进程，这个是zk的整个工程的main

#执行命令jps

20348 Jps

4233 QuorumPeerMain

**Kafka集群搭建**

**1、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\_2.11-1.0.0.tgz

**2、创建目录并下载安装软件**

#下载软件

cd /tmp/pkg

wget <http://mirrors.tuna.tsinghua.edu.cn/apache/kafka/1.0.0/kafka_2.11-1.0.0.tgz>

#解压软件

cd /opt/bigdata/

tar -zxvf /tmp/pkg/kafka\_2.11-1.0.0.tgz -C ./

mv kafka\_2.11-1.0.0/ kafka

#创建kafka消息目录，主要存放kafka消息

cd /data

sudo mkdir kafka

sudo chown hadoop:hadoop kafka

**3、修改配置文件**

进入到config目录

cd /opt/bigdata/kafka/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

-rw-r--r--. 1 root root 5699 Feb 22 09:41 192.168.88.89

-rw-r--r--. 1 root root 906 Feb 12 08:37 connect-console-sink.properties

-rw-r--r--. 1 root root 909 Feb 12 08:37 connect-console-source.properties

-rw-r--r--. 1 root root 2110 Feb 12 08:37 connect-distributed.properties

-rw-r--r--. 1 root root 922 Feb 12 08:38 connect-file-sink.properties

-rw-r--r--. 1 root root 920 Feb 12 08:38 connect-file-source.properties

-rw-r--r--. 1 root root 1074 Feb 12 08:37 connect-log4j.properties

-rw-r--r--. 1 root root 2055 Feb 12 08:37 connect-standalone.properties

-rw-r--r--. 1 root root 1199 Feb 12 08:37 consumer.properties

-rw-r--r--. 1 root root 4369 Feb 12 08:37 log4j.properties

-rw-r--r--. 1 root root 2228 Feb 12 08:38 producer.properties

-rw-r--r--. 1 root root 5699 Feb 15 18:10 server.properties

-rw-r--r--. 1 root root 3325 Feb 12 08:37 test-log4j.properties

-rw-r--r--. 1 root root 1032 Feb 12 08:37 tools-log4j.properties

-rw-r--r--. 1 root root 1023 Feb 12 08:37 zookeeper.properties

修改配置文件：

#每台服务器的broker.id都不能相同

broker.id=1

#listeners

listeners=PLAINTEST://master:9092

#日志

log.dirs=/data/kafka

#设置zookeeper的连接端口，3台，必须全部加进去

zookeeper.connect=192.168.88.88:2181,192.168.88.89:2181,192.168.88.90:2181

参数的解释：

broker.id=1 #当前机器在集群中的唯一标识，和zookeeper的myid性质一样

listeners=PLAINTEST://master:9092 #当前kafka对外提供服务的端口默认是9092

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/bigdata/kafka/logs/ #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=192.168.88.88:2181,192.168.88.89:2181,192.168.88.90:2181 #设置zookeeper的连接端口

**4、分发到集群其他机器**

master：

cd /opt/bigdata

scp -r kafka/ hadoop@slave1:/opt/bigdata/

scp -r kafka/ hadoop@slave2:/opt/bigdata/

slave1：

sudo mkdir /data/kafka

sudo chown hadoop:hadoop /data/kafka

cd /opt/bigdata/kafka/config/

vim server.properties

broker.id=2

listeners=PLAINTEST://slave1:9092

slave2：

sudo mkdir /data/kafka

sudo chown hadoop:hadoop /data/kafka

cd /opt/bigdata/kafka/config/

vim server.properties

broker.id=3

listeners=PLAINTEST://slave2:9092

**5、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 /opt/bigdata/kafka/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 Jps

4233 QuorumPeerMain

18991 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

#创建Topic

./kafka-topics.sh --create --zookeeper master:2181,slave1:2181,slave2:2181 --replication-factor 2 --partitions 1 --topic test

#解释

--replication-factor 2 #复制两份

--partitions 1 #创建1个分区

--topic #主题为test

'''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 master:9092 --topic test

'''在一台服务器上创建一个订阅者'''

./kafka-console-consumer.sh --zookeeper master:2181,slave1:2181,slave2:2181 --topic test --from-beginning

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~）：

4、其他命令

大部分命令可以去官方文档查看

4.1、查看topic

./kafka-topics.sh --list --zookeeper master:2181,slave1:2181,slave2:2181

#就会显示我们创建的所有topic

4.2、查看topic状态

./kafka-topics.sh --describe --zookeeper master:2181,slave1:2181,slave2:2181 --topic test

#下面是显示信息

Topic:test PartitionCount:2 ReplicationFactor:2 Configs:

Topic: test Partition: 0 Leader: 3 Replicas: 3,2 Isr: 3,2

Topic: test Partition: 1 Leader: 1 Replicas: 1,3 Isr: 1,3

OK kafka集群搭建完毕

5、其他说明标注

5.1、日志说明

默认kafka的日志是保存在/opt/bigdata/kafka/logs目录下的，这里说几个需要注意的日志

server.log #kafka的运行日志

state-change.log #kafka他是用zookeeper来保存状态，所以他可能会进行切换，切换的日志就保存在这里

controller.log #kafka选择一个节点作为“controller”,当发现有节点down掉的时候它负责在游泳分区的所有节点中选择新的leader,这使得Kafka可以批量的高效的管理所有分区节点的主从关系。如果controller down掉了，活着的节点中的一个会备切换为新的controller.

5.2、上面的大家你完成之后可以登录zk来查看zk的目录情况

#使用客户端进入zk

zkCli.sh #默认是不用加’-server‘参数的，除非我们修改了他的端口

#查看目录情况 执行“ls /”

[zk: localhost:2181(CONNECTED) 0] ls /

#显示结果：[cluster, controller, storm, brokers, zookeeper, hadoop-ha, admin, isr\_change\_notification, log\_dir\_event\_notification, controller\_epoch, tmp, consumers, latest\_producer\_id\_block, config, hbase]

'''

上面的显示结果中：只有zookeeper是zookeeper原生的，其他有的是Kafka创建的

'''

#标注一个重要的

[zk: localhost:2181(CONNECTED) 1] get /brokers/ids/1

{"listener\_security\_protocol\_map":{"PLAINTEXT":"PLAINTEXT"},"endpoints":["PLAINTEXT://master:9092"],"jmx\_port":-1,"host":"master","timestamp":"1518643440088","port":9092,"version":4}

cZxid = 0x100000004f

ctime = Wed Jan 10 05:32:32 CST 2018

mZxid = 0x100000004f

mtime = Wed Jan 10 05:32:32 CST 2018

pZxid = 0x100000004f

cversion = 0

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x360dc90ae5d0001

dataLength = 182

numChildren = 0

#还有一个是查看partion

[zk: localhost:2181(CONNECTED) 2] get /brokers/topics/test/partitions/0

null

cZxid = 0x100000006f

ctime = Wed Jan 10 05:38:50 CST 2018

mZxid = 0x100000006f

mtime = Wed Jan 10 05:38:50 CST 2018

pZxid = 0x1000000070

cversion = 1

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 0

numChildren = 1