**大**

**数**

**据**

**资**

**料**

**大数据资料**

一、hadoop

二、hive

三、hbase

四、spark

五、flume

六、kafka

七、zookeeper

八、logstash

九、storm

十、maven

十一、flink

十二、zeppelin

**基础准备**

**1.查看本机IP地址**

ifconfig –a

**2.修改hosts文件**

在根目录的etc文件夹中

把IP-机器名的映射加进去

192.168.xxx.xxx hadoop001

**3. 关闭防火墙**

一般来说，ubuntu默认都是安装防火墙软件ufw的，使用命令 sudo ufw version,如果出现ufw的版本信息，则说明已有ufw。

使用命令 sudo ufw status查看防火墙开启状态：

如果是active则说明开启，如果是inactive则说明关闭。

开启/关闭防火墙 sudo ufw enable|disable

使用sudo ufw disble来关闭防火墙，并再次用sudo ufw status 查看防火墙是否关闭。

**4. 安装jdk**

下载JDK，解压，然后在根目录下的~/.bash\_profile文件中添加

export JAVA\_HOME=/home/hadoop/app/jdk1.8.0\_161

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

使之生效source ~/.bash\_profile

检验是否生效命令：echo $JAVA\_HOME

**5. 安装SSH服务**

(1)首先确保虚拟机能连上网

(2)更新源列表：sudo apt-get update

(3)安装ssh:输入"sudo apt-get install openssh-server"-->回车-->输入"y"-->回车-->安装完成。

(4)查看ssh服务是否启动

打开"终端窗口"，输入"sudo ps -e |grep ssh"-->回车-->有sshd,说明ssh服务已经启动，如果没有启动，输入"sudo service ssh start"-->回车-->ssh服务就会启动。

同理，停止ssh服务sudo service ssh stop

重启ssh服务sudo service ssh restart

**6. 建立ssh无密码登录本机**

ssh生成密钥有rsa和dsa两种生成方式，默认情况下采用rsa方式。

创建ssh-key，这里我们采用rsa方式。

ssh –keygen -t rsa

（注：回车后会在~/.ssh/下生成两个文件：id\_rsa和id\_rsa.pub这两个文件是成对出现的）

进入~/.ssh/目录下，将id\_rsa.pub追加到authorized\_keys授权文件中，开始是没有authorized\_keys文件的.

cd ~/.ssh

cp id\_rsa.pub authorized\_keys

cp：复制成新文件

登录localhost ：ssh localhost

登录（连接）到其它机器 ssh 用户名@主机名

若不行，试试ssh 用户名@IP地址

执行退出命令exit﻿​

**7.检测端口**

netstat -antp

netstat -tlp

netstat -ntupl

netstat –ano|more

**8.查看进程与杀死进程**

jps查看进程

jps –m 输出启动时，传给main method的参数

jps –l 输出完全的包名，主类的全类名，jar的完全路径名

jps –v 输出jvm参数

kill -9 进程编号 杀死进程

**9.权限赋予**

sudo chmod 文件名

|  |  |  |
| --- | --- | --- |
| 读 | 写 | 执行 |
| r | w | x |
| 4 | 2 | 1 |
| 读、写、执行权限都没有，则为0 | | |
|  | | |
| 拥有者 | 群组 | 其他组 |
| u | g | o |
| 所有人都有权限为a (all) | | |

**+**添加某个权限

**-**取消某个权限

**=**赋予给定的权限

例如：chmod a+r, ug+w, o-w test.txt

Test.txt文件所有用户可读，拥有者及其所属群组可读写，其他组可读但不可写

|  |  |
| --- | --- |
| rwx=4+2+1=7 | 可读、可写、可执行 |
| rw=4+2=6 | 可读、可写 |
| rx=4+1=5 | 可读、可执行 |
|  | |
| 777 | -rwx-rwx-rwx |
| 764 | -rwx-rw-r |
| 644 | -rw-r-r |
| 600 | -rw-- |

**10.修改时间**

修改系统时间：sudo date MMDDHHmmYYYY.ss

(月、日、时、分、年、秒)

修改硬件时间：sudo hwclock –systohc

**11.发送文件到其它主机**

scp –r ~/app hadoop@hadoop002: ~/

把本机根目录下的app文件夹发送到hadoop002这台机器的hadoop用户的根目录下

**12. 安装并启动 FTP 服务**

一般来说著名的linux系统基本上分两大类：

1.RedHat系列：Redhat、Centos、Fedora等

2.Debian系列：Debian、Ubuntu等

RedHat 系列

1. 常见的安装包格式 rpm包,安装rpm包的命令是“rpm -参数”

2. 包管理工具 yum

3. 支持tar包

Debian系列

1. 常见的安装包格式 deb包,安装deb包的命令是“dpkg -参数”

2. 包管理工具 apt-get

3. 支持tar包

tar 只是一种压缩文件格式，所以，它只是把文件压缩打包而已。

rpm 相当于windows中的安装文件，它会自动处理软件包之间的依赖关系。

优缺点来说，rpm一般都是预先编译好的文件，它可能已经绑定到某种CPU或者发行版上面了。

tar一般包括编译脚本，你可以在你的环境下编译，所以具有通用性。

安装 vsftpd：

sudo apt-get install vsftpd

或者使用 yum 安装 vsftpd：

yum install -y vsftpd

安装完成后，启动 FTP 服务：

sudo service vsftpd start

查看系统是否已经监听了21 端口：

sudo netstat -nltp | grep 21

查看当前FTP服务运行状态：

sudo service vsftpd status

重新启动 FTP 服务：

sudo service vsftpd restart

停止 FTP 服务：

sudo service vsftpd stop

13.

**一、hadoop**

**1.下载**

下载hadoop-2.6.0-cdh5.7.0版本

http://archive-primary.cloudera.com/cdh5/cdh/5/hadoop-2.6.0-cdh5.7.0.tar.gz

**2.解压**

tar -zxvf hadoop-2.6.0-cdh5.7.0.tar.gz -C ~/app

**3.配置环境变量**

在根目录下的~/.bash\_profile文件中添加

export HADOOP\_HOME=/home/hadoop/app/ hadoop-2.6.0-cdh5.7.0

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

使之生效source ~/.bash\_profile

检验是否生效命令：echo $HADOOP\_HOME

**4.配置文件**

这里要涉及到的配置文件有7个：

在hadoop-2.6.0-cdh5.7.0/etc/hadoop目录下

**hadoop-env.sh**

**yarn-env.sh**

**slaves**

**core-site.xml**

**hdfs-site.xml**

**mapred-site.xml**

**yarn-site.xml**

**(1) 首先配置core-site.xml文件**

在<configuration> </configuration>中加入以下代码：

1. <property>
2. <name>hadoop.tmp.dir</name>
3. <value>file:/home/hadoop/app/tmp/hdfs</value>
4. <description>A base for other temporary directories.</description>
5. </property>
6. <property>
7. <name>io.file.buffer.size</name>
8. <value>131072</value>
9. </property>
10. <property>
11. <name>fs.defaultFS</name>
12. <value>hdfs://hadoop001:8020</value>
13. </property>

**(2) 配置 hadoop-env.sh文件**

将JAVA\_HOME文件配置为本机JAVA\_HOME路径

#The java implementation to use

export JAVA\_HOME=/home/hadoop/app/ jdk1.8.0\_161

**(3) 配置 yarn-env.sh文件**

将JAVA\_HOME文件配置为本机JAVA\_HOME路径

#some Java parameters

export JAVA\_HOME=/home/hadoop/app/ jdk1.8.0\_161

**(4) 配置hdfs-site.xml文件**

在<configuration> </configuration>中加入以下代码

1. <property>
2. <name>dfs.replication</name>
3. <value>1</value>
4. </property>
5. <property>
6. <name>dfs.namenode.name.dir</name>
7. <value>file:/home/hadoop/app/tmp/hdfs/name</value>
8. <final>true</final>
9. </property>
10. <property>
11. <name>dfs.datanode.data.dir</name>
12. <value>file:/home/hadoop/app/tmp/hdfs/data</value>
13. <final>true</final>
14. </property>

**(5) 复制mapred-site.xml.template文件，并命名为mapred-site.xml**

配置 mapred-site.xml,在标签<configuration>中添加以下代码

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<property>

<name>mapreduce.jobhistory.address</name>

<value>hadoop001:10020</value>

</property>

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>hadoop001:19888</value>

</property>

</configuration>

**(6)** **配置yarn-site.xml文件**

1. <property>
2. <name>yarn.nodemanager.aux-services</name>
3. <value>mapreduce\_shuffle</value>
4. </property>
5. <property>
6. <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
7. <value>org.apache.hadoop.mapred.ShuffleHandler</value>
8. </property>

**(7) 配置slaves 文件**

把原本的localhost删掉，改为hadoop001

**5.启动hadoop(hdfs)**

先删除/home/hadoop/app/tmp目录下的hdfs文件夹

格式化文件系统：

hadoop namenode -format 或者hdfs namenode -format

进入/home/hadoop/app/hadoop2.6.0-cdh5.7.0/sbin目录

启动hdfs: ./start-dfs.sh

关闭hdfs: ./stop-dfs.sh

启动完成后，可以通过命令 jps 来判断是否成功启动，若成功启动则会列出如下进程: “NameNode”、”DataNode” 和 “SecondaryNameNode”。

Web 界面 http://hadoop001:50070

**6. hdfs dfs基本操作**

Hadoop fs与hdfs dfs一样

hdfs dfs -ls / 或者hadoop fs –ls / 列出hdfs/根目录下的文件和目录

hdfs dfs -ls -R / 以递归形式列出/目录下的文件，有带-R参数

hdfs dfs - lsr / 递归查看当前目录

hdfs dfs -mkdir /test 创建文件夹

hdfs dfs -mkdir -p /test/a/b/c 创建递归文件夹

hdfs dfs -touchz /test/123.txt 创建文件

hdfs dfs -cat /test/123.txt从头查看文件中的内容

hdfs dfs -tail /test/123.txt 从尾部查看最后1K的内容

hdfs dfs -text /test/123.txt查看文件中的内容

hdfs dfs rm /test/123.txt 删除文件(注：rm不可以删除文件夹)

hdfs dfs rm -f /test/123.txt 删除文件

hdfs dfs -rm -r /test/a/b/c 删除文件夹(递归删除)

hdfs dfs -cp 源目录 目标目录 复制文件和目录

hdfs dfs -mv 源目录 目标目录 移动文件和目录

hdfs dfs -chmod [权限参数] path 赋予权限

hdfs dfs -put 源文件夹 目标文件夹 上传文件

hdfs dfs -copyFromLocal 源文件夹 目标文件夹 作用同put

hdfs dfs -moveFromLocal 源文件夹 目标文件夹 上传后删除本地

hdfs dfs -get源文件夹 目标文件夹 下载文件

hdfs dfs -copyToLocal 源文件夹 目标文件夹 作用同get

hdfs dfs -moveToLocal 源文件夹 目标文件夹 下载后删除源文件

**7. 启动YARN**

要先启动hadoop ./sbin/start-dfs.sh

./sbin/start-yarn.sh # 启动YARN

./sbin/mr-jobhistory-daemon.sh start historyserver

# 开启历史服务器，才能在Web中查看任务运行情况

开启后通过 jps 查看，可以看到多了 NodeManager 和 ResourceManager 两个后台进程。

./sbin/stop-yarn.sh #关闭YARN

./sbin/mr-jobhistory-daemon.sh stop historyserver

# 关闭历史服务器

Web 界面 http://hadoop001:8088

**二、hive**

**1.下载**

下载hive-1.1.0-cdh5.7.0版本

http://archive-primary.cloudera.com/cdh5/cdh/5/hive-1.1.0-cdh5.7.0.tar.gz

**2.解压**

tar -zxvf hive-1.1.0-cdh5.7.0.tar.gz -C ~/app

**3.配置环境变量**

在根目录下的~/.bash\_profile文件中添加

export HIVE\_HOME=/home/hadoop/app/hive-1.1.0-cdh5.7.0

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

使之生效source ~/.bash\_profile

检验是否生效命令：echo $HIVE\_HOME

**4.配置文件**

conf目录下

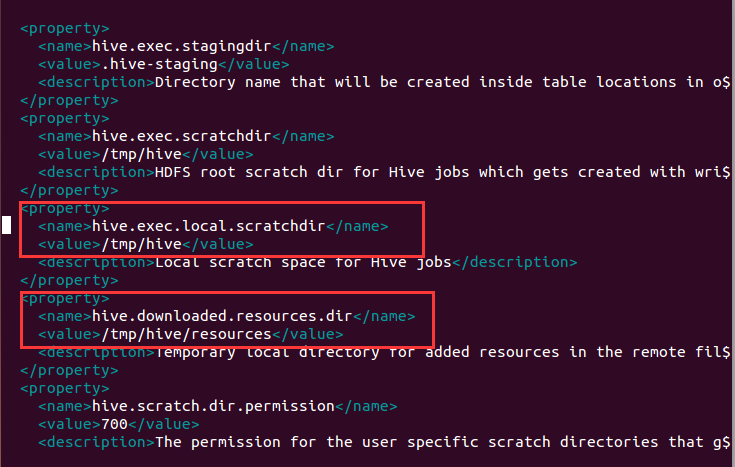
该路径下的 hive-site.xml 是 hive 工程的配置文件。默认情况下，该文件并不存在，我们需要拷贝它的模版来实现：

$ cp hive-default.xml.template hive-site.xml

hive-site.xml 的主要配置有：

* hive.metastore.warehouse.dir  
  该参数指定了 Hive 的数据存储目录，默认位置在 HDFS 上面的 /user/hive/warehouse 路径下。
* hive.exec.scratchdir  
  该参数指定了 Hive 的数据临时文件目录，默认位置为 HDFS 上面的 /tmp/hive 路径下。

同时我们还要修改 Hive 目录下 /conf/hive-env.sh 文件（请根据自己的实际路径修改），该文件默认也不存在，同样是拷贝它的模版来修改：



然后修改hive-site.sh文件权限，使用如下命令

sudo chmod u+x hive-env.sh

**配置hive-log4j.properties**

 在/opt/hive-1.2.1/conf目录下只有hive-log4j.properties.template文件，复制hive-log4j.properties.template并命名为hive-log4j.properties，命令如下

cp hive-log4j.properties.template hive-log4j.properties

将log4j.appender.EventCounter=org.apache.hadoop.hive.shims.HiveEventCounter

修改为：log4j.appender.EventCounter=org.apache.hadoop.log.metrics.EventCounter

否则会出现警告：

WARN conf.HiveConf: HiveConf of name hive.metastore.local does not exist  
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated. Please use org.apache.hadoop.log.metrics.EventCounter in all the log4j.properties files.

**5.启动hive**

先 ./start-all.sh 启动 Hadoop, 然后直接运行 hive：

**6.其它内容**

配置spark为默认引擎，hive使用spark有严格的版本限制， Hive根pom.xml的<spark.version>定义了使用它构建/测试的Spark的版本

版本不对会报如下错误：

FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.spark.SparkTask

*第一种方法*

1.要将Spark依赖项添加到Hive

在Hive 2.2.0之前，将spark-assembly jar链接到HIVE\_HOME / lib

2.配置Hive执行引擎使用Spark：

Hive-site.xml配置

<property>

<name>hive.execution.engine</name>

<value>spark</value>

</property>

*第二种方法*

配置hive-site.xml

<property>

<name>spark.home</name>

<value>/root/spark-without-hive</value>

</property>

**三、hbase**

**1.下载**

下载hbase-1.2.0-cdh5.7.0版本

http://archive-primary.cloudera.com/cdh5/cdh/5/hbase-1.2.0-cdh5.7.0.tar.gz

**2.解压**

tar -zxvf hbase-1.2.0-cdh5.7.0.tar -C ~/app

**3.配置文件**

**（1）单机模式配置**

**修改hbase-env.sh文件**

配置JAVA环境变量，并添加配置HBASE\_MANAGES\_ZK为false

export JAVA\_HOME=/home/hadoop/app/jdk1.8.0\_161

export HBASE\_MANAGES\_ZK=false

**修改hbase-site.xml文件**

添加配置如下：

<configuration>  
        <property>  
                <name>hbase.rootdir</name>  
                <value>file:///home/hadoop/app/tmp/hbase-tmp</value>  
        </property>  
</configuration>

**先启动zookeeper**

**./zkServer.sh start**

**启动hbase**

bin/start-hbase.sh

bin/hbase shell

上述命令中，sudo bin/start-hbase.sh用于启动HBase，bin/hbase shell用于打开shell命令行模式，用户可以通过输入shell命令操作HBase数据库。

**（2）伪分布式模式配置**

**修改hbase-env.sh文件**

配置JAVA环境变量，并添加配置HBASE\_MANAGES\_ZK为false，HBASE\_CLASSPATH设置为本机Hadoop安装目录下的conf目录。

export JAVA\_HOME=/home/hadoop/app/jdk1.8.0\_161

export HBASE\_CLASSPATH=/home/hadoop/app/hadoop-2.6.0-cdh5.7.0/conf

export HBASE\_MANAGES\_ZK=false

**修改hbase-site.xml文件**

修改hbase.rootdir，指定HBase数据在HDFS上的存储路径；将属性hbase.cluter.distributed设置为true。假设当前Hadoop集群运行在伪分布式模式下，在本机上运行，且NameNode运行在8020端口。

<configuration>  
        <property>  
                <name>hbase.rootdir</name>  
                <value>hdfs://hadoop001:8020/hbase</value>  
        </property>  
        <property>  
                <name>hbase.cluster.distributed</name>  
                <value>true</value>  
        </property>

<property>  
                <name>hbase.zookeeper.quorum</name>  
                <value>hadoop001:2181</value>  
     </property>  
</configuration>

**先启动zookeeper**

**./zkServer.sh start**

**接着启动hdfs,然后启动hbase**

bin/start-hbase.sh

进入shell界面：bin/hbase shell

停止HBase运行,命令如下：bin/stop-hbase.sh

**这里启动关闭Hadoop和HBase的顺序一定是：**

**启动Hadoop—>启动HBase—>关闭HBase—>关闭Hadoop**

**四、spark**

**1.下载**

参考博客:

https://blog.csdn.net/panguoyuan/article/details/74330519?utm\_source=blogxgwz8

**2.解压**

tar -zxvf spark-2.1.0-bin-2.6.0-cdh5.7.0.tar -C ~/app

**3.配置环境变量**

在根目录下的~/.bash\_profile文件中添加

export SPARK\_HOME=/home/hadoop/app/spark-2.1.0-bin-2.6.0-cdh5.7.0

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

使之生效source ~/.bash\_profile

检验是否生效命令：echo $SPARK\_HOME

**4.** **修改配置文件**

# 复制模版配置文件

cp conf/spark-env.sh.template conf/spark-env.sh

**（1）修改 spark-env.sh文件**

export JAVA\_HOME=/home/hadoop/app/jdk1.8.0\_161

export SCALA\_HOME=/home/hadoop/app/scala

export HADOOP\_HOME=/home/hadoop/app/hadoop-2.6.0-cdh5.7.0

export HADOOP\_CONF\_DIR=/home/hadoop/app/hadoop-2.6.0-cdh5.7.0/etc/hadoop

#定义管理端口

export SPARK\_MASTER\_WEBUI\_PORT=8088

#定义master域名和端口

export SPARK\_MASTER\_HOST=spark-master

export SPARK\_MASTER\_PORT=7077

#定义master的地址slave节点使用

export SPARK\_MASTER\_IP=spark-master

#定义work节点的管理端口.work节点使用

export SPARK\_WORKER\_WEBUI\_PORT=8088

#每个worker节点能够最大分配给exectors的内存大小

export SPARK\_WORKER\_MEMORY=4g

**（2）修改 slaves文件**

#复制模版配置文件

cp conf/slaves.template conf/slaves

#修改配置文件.

vim conf/slaves

添加内容

spark-slave1

spark-slave2

**（3）修改spark-defaults.conf文件**

spark.eventLog.enabled=true

spark.eventLog.compress=true

#保存在本地

#spark.eventLog.dir=file:///home/hadoop/hadoop-2.6.0-cdh5.7.0/logs/userlogs

#spark.history.fs.logDirectory=file:///home/hadoop/hadoop-2.6.0-cdh5.7.0/logs/userlogs

#保存在hdfs上

spark.eventLog.dir=hdfs://spark-master:8020/tmp/logs/root/logs

spark.history.fs.logDirectory=hdfs://spark-master:8020/tmp/logs/root/logs

spark.yarn.historyServer.address=spark-master:18080

**5. 启动spark**

bin目录下

./spark-shell –master local[2] 以local模式运行，2个线程

以Spark-submit的方式提交

./spark-submit –class 主类名

--master local[2]

--jars 连接包

/home/hadoop/要运行的jar包的地址

查看web管理界面 http://hadoop001:4040

**五、flume**

**1.下载flume**

下载flume-ng-1.6.0-cdh5.7.0 版本

http://archive-primary.cloudera.com/cdh5/cdh/5/flume-ng-1.6.0-cdh5.7.0.tar.gz

**2.解压到~/app目录**

tar -zxvf flume-ng-1.6.0-cdh5.7.0.tar -C ~/app

**3.配置环境变量**

在根目录下的~/.bash\_profile文件中添加

export FLUME\_HOME=/home/hadoop/flume-ng-1.6.0-cdh5.7.0

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

使之生效source ~/.bash\_profile

检验是否生效命令：echo $FLUME\_HOME

**4.启动flume**

Flume启动命令：

flume-ng agent

--name a1

--conf /home/hadoop/app/flume-1.6.0-cdh5.7.0/conf

--conf-file /home/hadoop/app/flume-1.6.0-cdh5.7.0/conf/test.conf

-Dflume.root.logger=INFO, console

日志级别，输出日志到控制台，也可以到log4j.properties日志配置文件中修改。

参数解释：

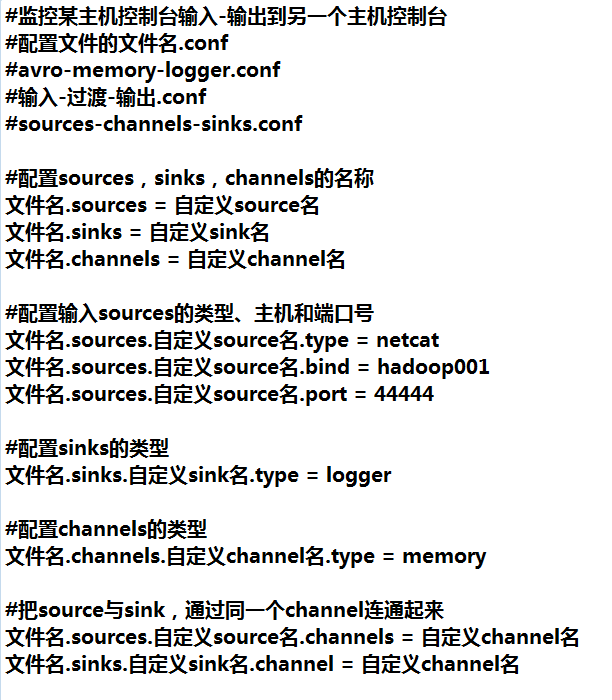
-c (--conf) ： flume的conf文件路径

-f (--conf-file) ： 自定义的flume配置文件

-n (--name)： 自定义的flume配置文件中agent的name

**5.配置文件**

（**1）控制台-控制台**



**（2）文件-控制台**



（3）内存-kafka



**六、kafka**

**1.下载**

下载kafka\_2.11-0.9.0.0版本

https://archive.apache.org/dist/kafka/0.9.0.0/kafka\_2.11-0.9.0.0.tgz

**2.解压**

tar -zxvf kafka\_2.11-0.9.0.0.tgz -C ~/app

**3.配置环境变量**

在根目录下的~/.bash\_profile文件中添加

export KAFKA\_HOME=/home/hadoop/app/kafka\_2.11-0.9.0.0

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

使之生效source ~/.bash\_profile

检验是否生效命令：echo $KAFKA\_HOME

**4.配置文件**

Config目录下的server.properties文件

broker.id=0

port=9092

host.name=hadoop001

log.dirs=/home/hadoop/app/tmp/kafka-logs

num.partitions=1

zookeeper.connect=hadoop001:2181

**5.启动kafka**

先启动zookeeper，可以用kafka的bin目录里面的命令快速启动zookeeper

bin/zookeeper-server-start.sh config/zookeeper.properties

现在启动Kafka服务器：（单节点单broker的方式）

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

-daemon表示以后台进程的方式启动

**6. 创建一个主题topic**

用一个分区和一个副本创建一个名为“test”的主题：

bin/kafka-topics.sh

--create

--zookeeper hadoop001:2181

--replication-factor 1

--partitions 1

--topic test

运行列表主题命令，查看所有主题：

> bin/kafka-topics.sh

--list

--zookeeper hadoop001:2181

test

查看所有主题的详细信息:

bin/kafka-topics.sh

--describe

--zookeeper hadoop001:2181

查看指定主题的详细信息:

> bin/kafka-topics.sh

--describe

--zookeeper hadoop001:2181

--topic test

Topic:test PartitionCount:1 ReplicationFactor:1 Configs:

Topic:test Partition: 0 Leader: 0 Replicas: 0 Isr: 0

“leader”是负责对给定分区进行所有读写的节点。每个节点将是随机选择的分区部分的领导者。

“replicas”是复制此分区日志的节点列表，而不管它们是领导者还是当前活动的节点。

“Isr”是一组“同步”副本。这是副本列表的子集，表示当前还活着的，并且能被leader捕获到的副本。

**7.发送消息**

启动生产者，生产消息

> bin/kafka-console-producer.sh

--broker-list hadoop001:9092

--topic test

This is a message

This is another message

**8.接收消息**

启动消费者，消费消息，将消息转储到标准输出（控制台）

> bin/kafka-console-consumer.sh

--zookeeper hadoop001:2181

--topic test

--from-beginning

This is a message

This is another message

参数解释：

--from-beginning 从头开始消费消息，包含以前的消息；不带这个参数就是只接收消费者启动之后的消息，消费者启动之前的消息不接收。

**9. 启动Kafka服务器以单节点多broker的方式**

一个server.properties就是一个broker

首先，我们为每个broker创建一个配置文件(使用copy命令)：

> cp config/server.properties config/server-1.properties

> cp config/server.properties config/server-2.properties

现在编辑这些新文件并设置以下属性：

config/server-1.properties:

broker.id=1

listeners=PLAINTEXT://:9093

log.dirs=/home/hadoop/app/tmp/kafka-logs-1

config/server-2.properties:

broker.id=2

listeners=PLAINTEXT://:9094

log.dirs=/home/hadoop/app/tmp/kafka-logs-2

启动zookeeper后，启动单节点，我们的单个节点已经启动，所以我们只需要启动两个新节点：

bin/kafka-server-start.sh -daemon config/server-1.properties &

bin/kafka-server-start.sh -daemon config/server-2.properties &

创建一个“1个分区3个副本”的新主题：

bin/kafka-topics.sh

--create

--zookeeper hadoop001:2181

--replication-factor 3

--partitions 1

--topic my-replicated-topic

生产消息到多个篮子：

bin/kafka-console-producer.sh

--broker-list hadoop001:9092 hadoop001:9093 hadoop001:9094

--topic my-replicated-topic

**七、zookeeper**

**1.下载zookeeper**

Zookeeper-3.4.5-cdh5.7.0版本下载地址：

http://archive-primary.cloudera.com/cdh5/cdh/5/zookeeper-3.4.5-cdh5.7.0.tar.gz

**2.解压到~/app目录**

tar -zxvf zookeeper-3.4.5-cdh5.7.0.tar.gz -C ~/app

**3. 配置环境变量**

在根目录下的~/.bash\_profile文件中添加

export ZK\_HOME=/home/hadoop/app/zookeeper-3.4.5-cdh5.7.0

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

使之生效source ~/.bash\_profile

检验是否生效命令：echo $ZK\_HOME

**4.修改配置文件**

在conf目录下，创建一个zoo.cfg文件

cp zoo\_sample.cfg zoo.cfg

修改zoo.cfg文件中的dataDir

dataDir=/home/hadoop/app/tmp/zookeeper

**5.启动zookeeper**

bin目录下

./ zkServer.sh start

**(1)启动客户端**

./ zkCli.sh 连接本地服务器

./zkCli.sh -server ip:port 连接指定IP地址服务器

**(2)连接后help，查看帮助信息**

ls / 列出根目录所有节点

create /test abc 创建节点并赋值

get /test 获取指定节点的值

set /test cb 设置已存在节点的值

rmr /test 递归删除节点

delete /test/test01 删除不存在子节点的节点

**八、 logstash**

**1.下载logstash**

logstash2.4.1版本下载地址：

https://www.elastic.co/downloads/past-releases/logstash-2-4-1

**2.解压到~/app目录**

tar –zxvf logstash-2.4.1.tar.gz –C ~/app

**3.配置文件**

**（1）监控控制台，输出到控制台**

cd logstash-2.4.1

bin/logstash -e 'input { stdin { } } output { stdout {} }'

**（2）监控控制台，输出到控制台json格式**

bin/logstash -e 'input { stdin { } } output { stdout {codec=>json} }'

**（3）监控控制台，输出到文件out.txt**

用-f命令，先写好配置文件test.conf

input {

stdin { }

}

output {

file {

path => "/home/hadoop/out.txt"

}

}

然后运行命令bin/logstash –f test.conf

**（4）监控控制台，以json格式输出到文件out.txt**

用-f命令，先写好配置文件test.conf

input {

stdin { }

}

output {

file {

path => "/home/hadoop/out.txt"

codec => "json"

}

}

然后运行命令bin/logstash –f test.conf

**（5）监控文件in.txt，输出到控制台**

用-f命令，先写好配置文件test.conf

input {

file {

path => "/home/hadoop/in.txt"

}

}

output {

stdout {}

}

然后运行命令bin/logstash –f test.conf

**（6）监控文件in.txt，输出到文件out.txt**

用-f命令，先写好配置文件test.conf

input {

file {

path => "/home/hadoop/in.txt"

}

}

output {

file {

path => "/home/hadoop/out.txt"

}

}

然后运行命令bin/logstash –f test.conf

**（7）监控多个文件in1.txt，in2.txt，输出到文件out.txt**

用-f命令，先写好配置文件test.conf

input {

file {

path => "/home/hadoop/in1.txt"

}

file {

path => "/home/hadoop/in2.txt"

}

}

output {

file {

path => "/home/hadoop/out.txt"

}

}

然后运行命令bin/logstash –f test.conf

例子：监控控制台，输出到控制台，运行以下命令

bin/logstash -e 'input { stdin { } } output { stdout {} }'

输入hello world敲回车，结果如下：

hello world

2013-11-21T01:22:14.405+0000 0.0.0.0 hello world

**4.logstash整合kafka**

Logstash监控文件，输出消息到kafka的topic

（1）先启动zookeeper

./ zkServer.sh start

（2）接着启动kafka

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

（3）接着创建topic

./kafka-topics.sh

--create

--zookeeper hadoop001:2181

--replication-factor 1

--partitions 1

--topic logstash\_topic

（4）接着写logstash配置文件file\_kafka.conf

监控in.txt文件，输出到kafka的logstash\_topic

input {

file {

path => "/home/hadoop/in.txt"

}

}

output {

kafka {

codec => json

topic\_id => "logstash\_topic"

bootstrap\_servers => “hadoop001:9092”

batch\_size => 1

}

}

bootstrap\_servers表示kafka服务器的地址和端口

Batch\_size表示批处理的大小，一次处理多少消息

（5）然后启动kafka消费者

./kafka-console-consumer.sh

--zookeeper hadoop001:2181

--topic logstash\_topic

（6）然后启动logstash

./logstash –f file\_kafka.conf

（7）往in.txt里面加内容

echo “hello” >> in.txt

**九、storm**

**1.下载**

<https://archive.apache.org/dist/storm/apache-storm-1.1.1/apache-storm-1.1.1.tar.gz>

**2.解压**

tar -zxvf apache-storm-1.1.1.tar.gz –C ~/app

**3.配置环境变量**

在根目录下的~/.bash\_profile文件中添加

export STORM\_HOME=/home/hadoop/app/apache-storm-1.1.1

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

使之生效source ~/.bash\_profile

检验是否生效命令：echo $STORM\_HOME

**4.配置文件**

**(1)单机模式**

配置conf目录下的storm-env.sh文件

导出JAVA\_HOME

# The java implementation to use.

#export JAVA\_HOME=/path/to/jdk/home

export JAVA\_HOME=/home/hadoop/app/jdk1.7.0\_51

**启动storm**

bin目录下

执行storm就能看到详细的命令：

storm dev-zookeeper 前台启动zookeeper

nohup sh storm dev-zookeeper 后台启动zookeeper

storm jar 要运行的jar包路径 主类名 参数 --jars 额外的jar包

storm kill 作业名字 -w 等待时间 杀死作业

storm list 列出正在运行的作业和它们的状态

storm logviewer 启动日志查看服务

storm nimbus 启动主节点

storm supervisor 启动从节点

storm ui 启动UI界面(进程名称叫core)

命令前加nohup sh 就可以以后台（挂起）形式启动

访问ui页面： http://hadoop001:8080

**(2)集群模式**

用3台机器：hadoop000,hadoop001,hadoop002

etc/hosts文件中的IP地址-主机名 映射关系要写好。

每台机器要部署的东西：JAVA\_HOME，ZK\_HOME，STORM\_HOME

hadoop000:zookeeper,nimbus,supervisor

hadoop001:zookeeper,supervisor

hadoop002:zookeeper,supervisor

一个集群中多台机器可以复用同一个zookeeper

**先安装zookeeper分布式环境：**

3台机器的zoo.cfg文件添加：

server.1=hadoop000:2888:3888

server.2=hadoop001:2888:3888

server.3=hadoop002:2888:3888

接着在3台机器的zoo.cfg文件中的那个dataDir目录下，创建一个myid文件，里面分别写一个1,2,3

hadoop000: myid 1

hadoop001: myid 2

hadoop002: myid 3

然后每台机器都要启动zookeeper: ./zkServer.sh start

查看状态./zkServer.sh status

看到Mode:follower表示是从节点，看到Mode:leader表示是主节点，哪台机器是主，哪台机器是从，这是随机分配的。

**接着配置storm集群**

每台机器都要有apache-storm-1.1.1包，配置conf目录下的storm-env.sh文件，导出JAVA\_HOME。

接着配置conf目录下storm.yaml文件（注意规范格式，记得删除前面的#号）

* **配置Zookeeper地址（配置Zookeeper的主机名，注意: 如果Zookeeper集群使用的不是默认端口，那么还需要配置storm.zookeeper.port）**

**storm.zookeeper.servers:  
  - "hadoop000"  
  - "hadoop001"**

**- "hadoop002"**

* **在配置文件里添加一行：(记得冒号后面有一个空格)这个目录要有读写权限**

storm.local.dir: "/home/hadoop/app/tmp/storm"

* **配置nimbus.seeds：用于配置主控节点的地址，可以配置多个。**
* **配置supervisor.slots.ports 配置worker**

supervisor.slots.ports:

- 6700

- 6701

- 6702

- 6703

**接着hadoop000机器启动niumbus：**

nohup sh storm nimbus

启动UI 访问ui页面： http://hadoop000:8080

nohup sh storm ui

启动supervisor

nohup sh storm supervisor

启动logviewer

nohup sh storm logviewer

其它机器hadoop001,hadoop002只需启动supervisor和logviewer。

**运行作业**

在hadoop000主节点这台机器上提交作业

storm jar 要运行的jar包路径 主类名 参数 --jars 额外的jar包

正在运行的作业的名称不能相同

**并行度**

一个worker进程执行的是一个topo的子集

一个worker进程会启动1...n个executor线程来执行一个topo的component

一个运行的topo就是由集群中多台物理机上的多个worker进程组成

executor是一个被worker进程启动的单独线程，每个executor只会运行1个topo的一个component

task是最终运行spout或者bolt代码的最小执行单元

**默认：**

一个supervisor节点最多启动4个worker进程

每一个topo默认占用一个worker进程

每个worker进程会启动一个executor

每个executor启动一个task

**Storm架构**

类似于Hadoop的架构，主从(Master/Slave)

Nimbus: 主

集群的主节点，负责任务(task)的指派和分发、资源的分配

Supervisor: 从

可以启动多个Worker，具体几个呢？可以通过配置来指定

一个Topo可以运行在多个Worker之上，也可以通过配置来指定

集群的从节点，(负责干活的)，负责执行任务的具体部分

启动和停止自己管理的Worker进程

无状态，在他们上面的信息(元数据)会存储在ZK中

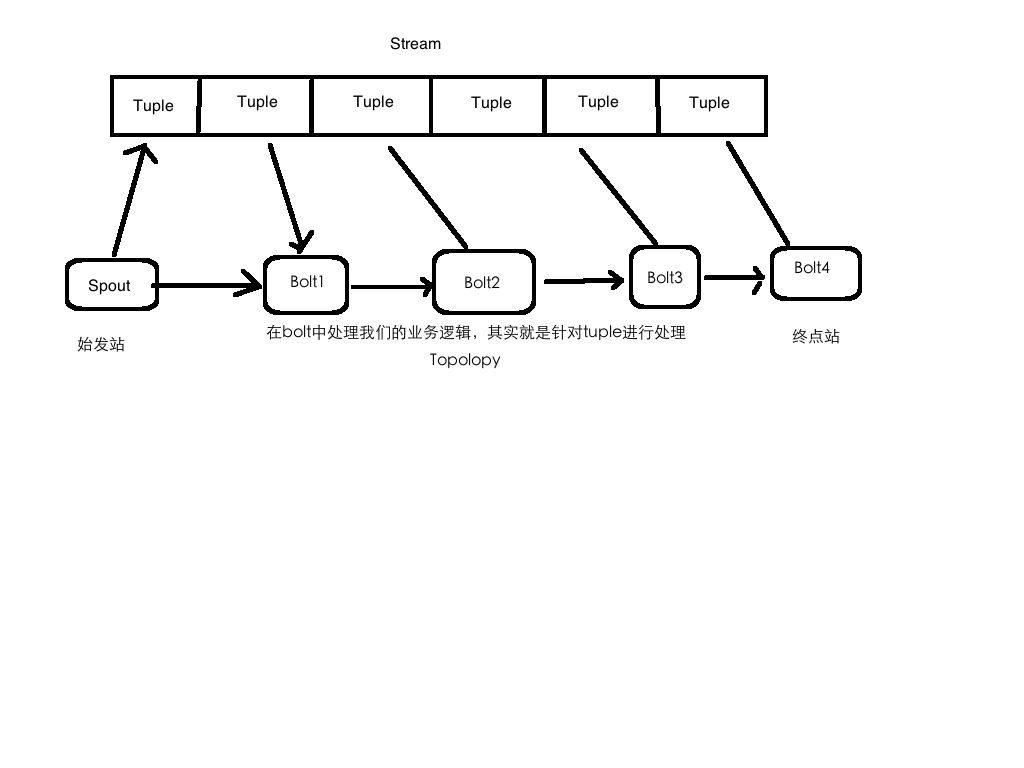
Worker: 运行具体组件逻辑(Spout/Bolt)的进程

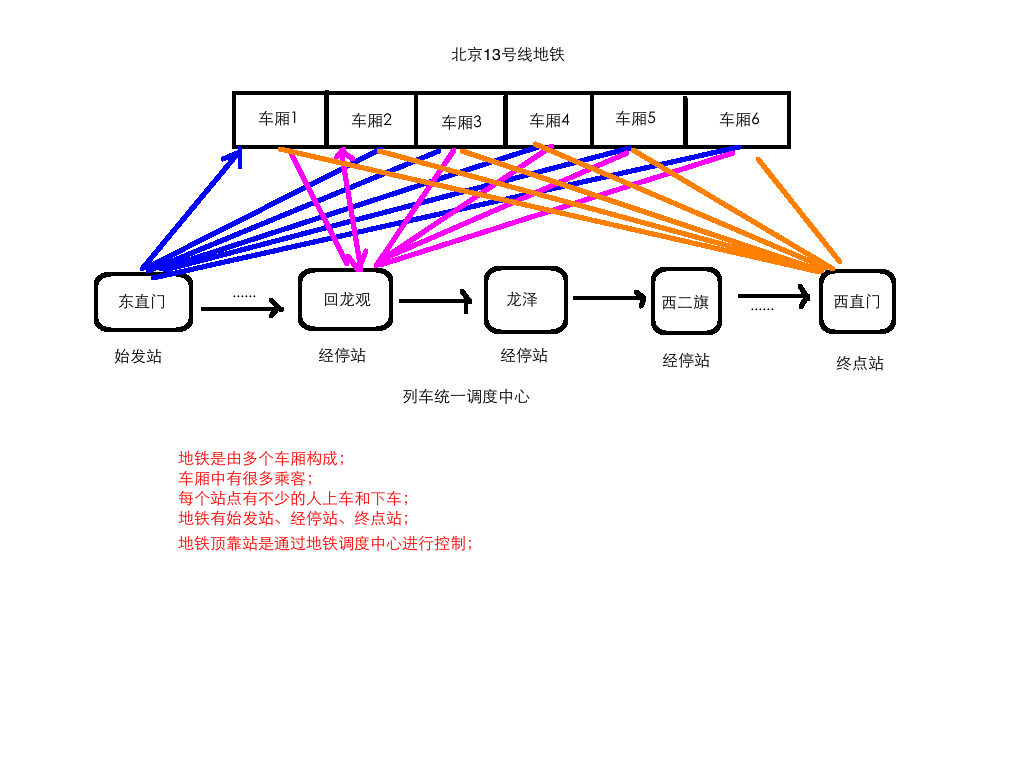
task：

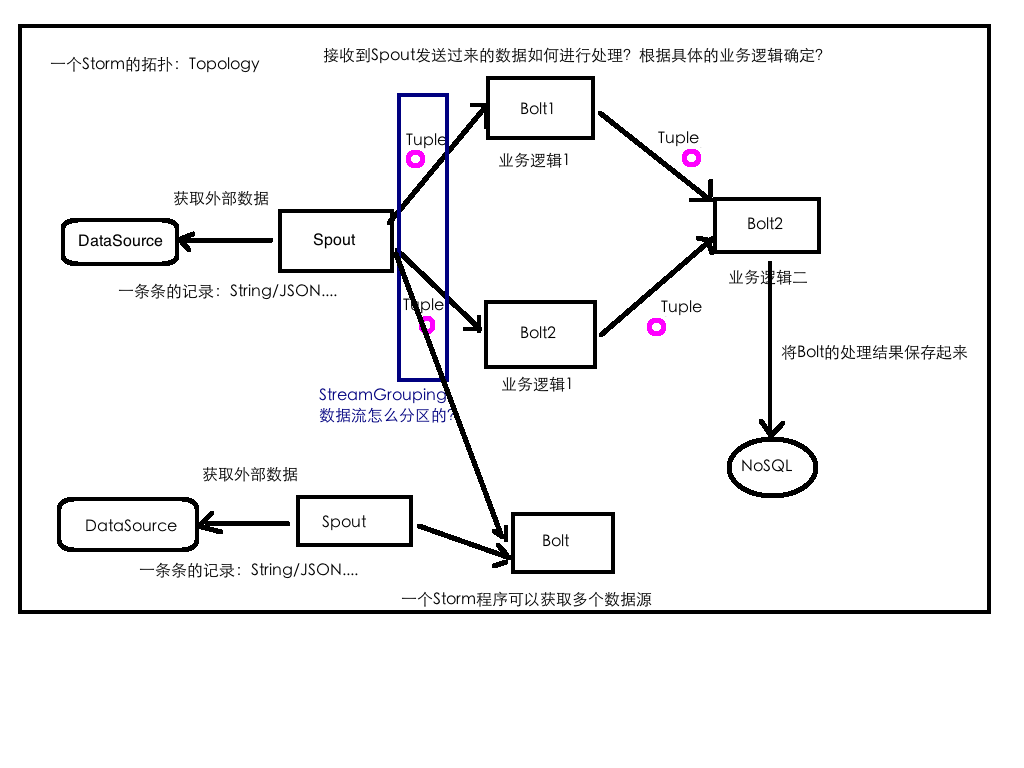
Spout和Bolt

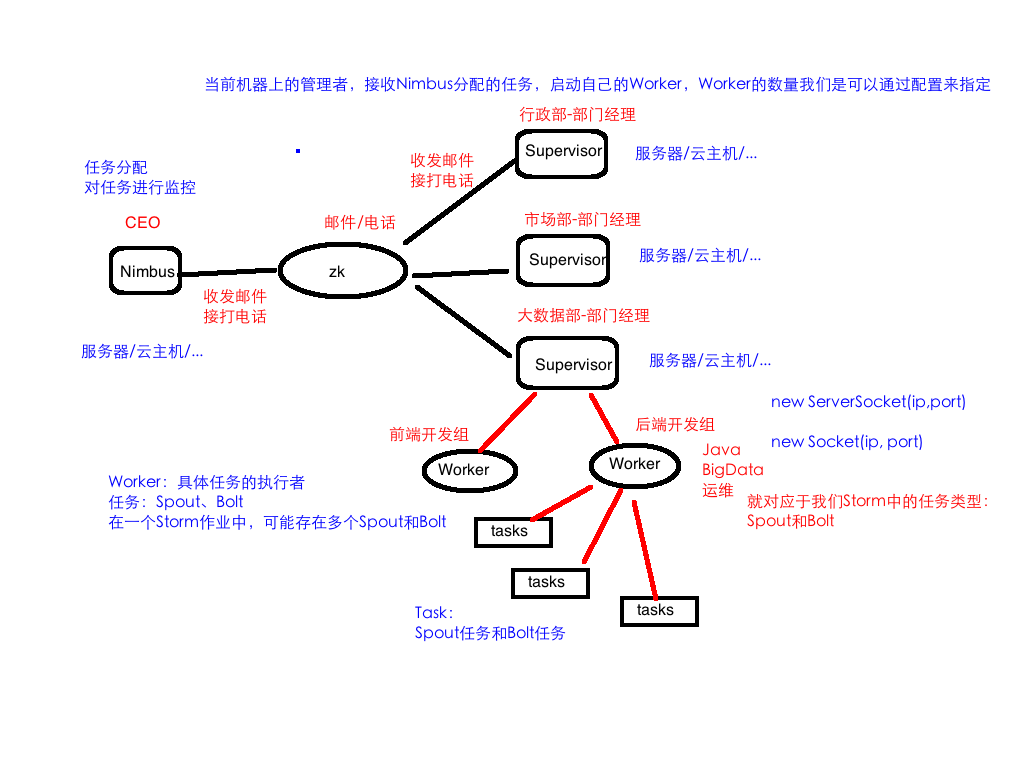
Worker中每一个Spout和Bolt的线程称为一个Task

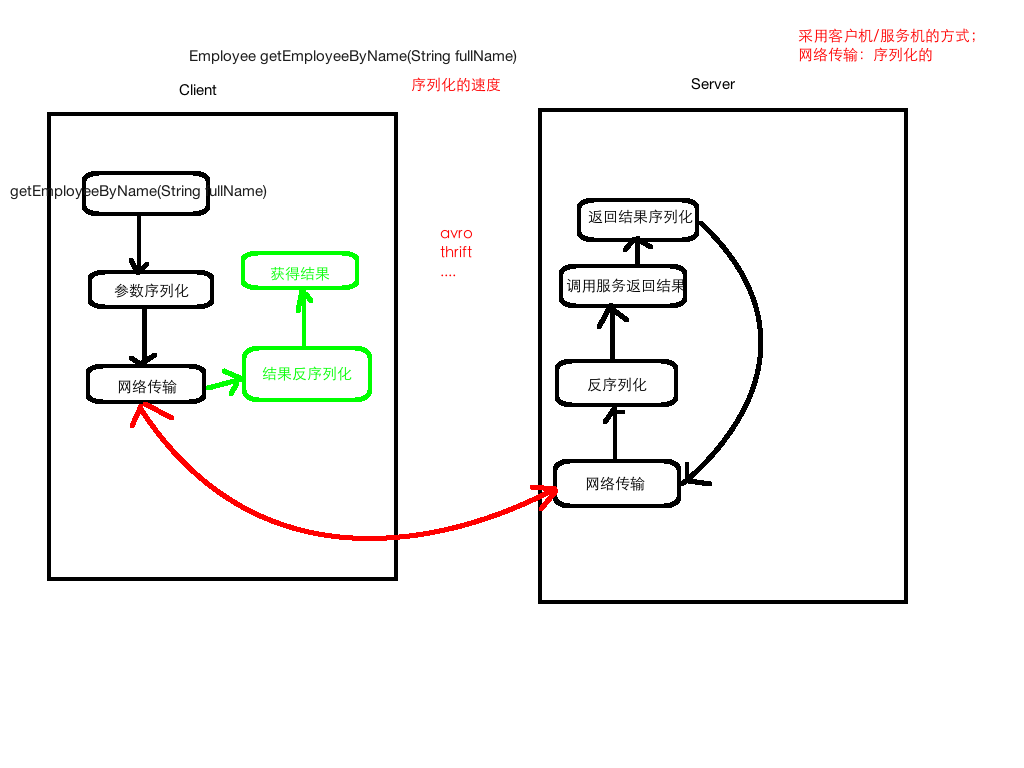
executor： spout和bolt可能会共享一个线程



****

****

****

****

**十、maven**

**1.下载**

https://maven.apache.org/download.cgi

**2.解压**

tar -zxvf maven-3.3.9.tar.gz –C ~/app

**3.配置环境变量**

在根目录下的~/.bash\_profile文件中添加

export MAVEN\_HOME=/home/hadoop/app/maven-3.3.9

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

使之生效source ~/.bash\_profile

检验是否生效命令：echo $MAVEN\_HOME

**4.配置文件**

配置Maven本地仓库

conf目录下的settings.xml文件

<localRepository>/home/hadoop/.m2</localRepository>

**5.用maven打包文件**

mvn clean package -DskipTests 跳过测试，打包

mvn compile编译

**十一、flink**

**1.下载**

https://archive.apache.org/dist/flink/flink-1.5.0/flink-1.5.0-bin-hadoop26-scala\_2.11.tgz

**2.解压**

tar –zxvf flink-1.5.0-bin-hadoop26-scala\_2.11.tgz –C ~/app

**3.配置**

**4.启动**

bin目录下

./start-local.sh

jps有JobManager进程

Web界面http://hadoop001:8081

执行wordcount样例

./bin/flink run ./examples/batch/wordcount.jar

--输入文件的地址

--输出文件的地址

**十二、zeppelin**

**1.下载**

https://archive.apache.org/dist/zeppelin/zeppelin-0.7.0/zeppelin-0.7.0.tgz

**2.解压**

tar –zxvf zeppelin-0.7.0.tgz –C ~/app

**3.配置**

**4.启动**

定位到 Zeppelin 安装目录的bin文件夹下，使用以下命令启动进程：

./zeppelin-daemon.sh start

若需要停止，可以使用以下命令停止进程：

./zeppelin-daemon.sh stop