一、配置部署环境

1.1 修改主机名

依次登陆所有节点，分别修改主机名为master、worker1、worker2、worker3

```
hostnamectl set-hostname 主机名 --static
```

(注：可能需要重新打开SSH连接才能看到主机名变更)

1.2 修改hosts文件

依次登陆所有节点，修改“/etc/hosts”文件

```
vim /etc/hosts
```

添加集群所有节点的“地址-主机名“映射关系

```
IPaddress1 master
IPaddress2 worker1
IPaddress3 worker2
IPaddress4 worker3
```

1.3 关闭防火墙

在所有节点上关闭防火墙

```
systemctl stop firewalld.service
systemctl disable firewalld.service
```

1.4 配置SSH免密登陆

依次在**所有节点**上执行以下步骤：

a.生成密钥，并按照提示按三次回车

```
ssh-keygen -t rsa
```

b.配置SSH免密登录

```
ssh-copy-id -i ~/.ssh/id_rsa.pub root@master
ssh-copy-id -i ~/.ssh/id_rsa.pub root@worker1
ssh-copy-id -i ~/.ssh/id_rsa.pub root@worker2
ssh-copy-id -i ~/.ssh/id_rsa.pub root@worker3
```

1.5 安装OpenJDK

在所有节点上执行以下步骤：

a.下载和解压

```
wget https://github.com/nju-tcy/flink_files/raw/master/OpenJDK8U-
jdk_x64_linux_hotspot_8u252b09.tar.gz
tar -zxf OpenJDK8U-jdk_x64_linux_hotspot_8u252b09.tar.gz -C /usr/local
```

b.添加环境变量

```
vim /etc/profile
```

```
export JAVA_HOME=/usr/local/jdk8u252-b09
export PATH=$JAVA_HOME/bin:$PATH
```

c.生效环境变量

```
source /etc/profile
```

d.验证OpenJDK是否安装成功

```
java -version
```

```
[root@master ~]# java -version
openjdk version "1.8.0_252"
OpenJDK Runtime Environment (AdoptOpenJDK)(build 1.8.0_252-b09)
OpenJDK 64-Bit Server VM (AdoptOpenJDK)(build 25.252-b09, mixed mode)
```

二、部署Zookeeper

2.1 在worker1上解压Zookeeper

```
cd /usr/local
wget https://github.com/nju-tcy/flink_files/raw/master/apache-zookeeper-3.6.2-
bin.tar.gz
tar -zxvf apache-zookeeper-3.6.2-bin.tar.gz
```

建立软链接

```
ln -s apache-zookeeper-3.6.2-bin zookeeper
```

2.2 添加Zookeeper到环境变量

a.添加环境变量

```
vim /etc/profile
```

```
export ZOOKEEPER_HOME=/usr/local/zookeeper
export PATH=$ZOOKEEPER_HOME/bin:$PATH
```

b.生效环境变量

```
source /etc/profile
```

2.3 修改Zookeeper配置文件

a.切换到配置文件所在目录

```
cd /usr/local/zookeeper/conf
```

b.根据配置文件模版拷贝配置文件

```
cp zoo_sample.cfg zoo.cfg
```

c.修改配置文件

```
vim zoo.cfg
```

修改数据目录

```
dataDir=/usr/local/zookeeper/tmp
```

添加以下内容

```
server.1=worker1:2888:3888
server.2=worker2:2888:3888
server.3=worker3:2888:3888

admin.enableServer=true
quorumListenOnAllIPs=true
```

```
# 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=/usr/local/zookeeper/tmp
# 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

## Metrics Providers
#
# https://prometheus.io Metrics Exporter
#metricsProvider.className=org.apache.zookeeper.metrics.prometheus.PrometheusMetricsProvider
#metricsProvider.httpPort=7000
#metricsProvider.exportJvmInfo=true
server.1=worker1:2888:3888
server.2=worker2:2888:3888
server.3=worker3:2888:3888

admin.enableServer=true
quorumListenOnAllIPs=true
```

d.创建tmp文件夹作为dataDir

```
mkdir /usr/local/zookeeper/tmp
```

e.创建myid文件并写入myid的内容

```
touch /usr/local/zookeeper/tmp/myid
echo 1 > /usr/local/zookeeper/tmp/myid
```

2.4 同步配置到其他worker节点

a.将Zookeeper拷贝到其他worker节点

```
scp -r /usr/local/apache-zookeeper-3.6.2-bin root@worker2:/usr/local
scp -r /usr/local/apache-zookeeper-3.6.2-bin root@worker3:/usr/local
```

b.在worker2、worker3上创建软链接并修改myid内容

worker2

```
cd /usr/local
ln -s apache-zookeeper-3.6.2-bin zookeeper
echo 2 > /usr/local/zookeeper/tmp/myid
```

worker3

```
cd /usr/local
ln -s apache-zookeeper-3.6.2-bin zookeeper
echo 3 > /usr/local/zookeeper/tmp/myid
```

2.5 运行验证

a.分别在worker1、worker2、worker3上启动Zookeeper

```
/usr/local/zookeeper/bin/zkServer.sh start
```

（停止Zookeeper的命令是）

```
/usr/local/zookeeper/bin/zkServer.sh stop
```

b.分别查看Zookeeper状态

```
/usr/local/zookeeper/bin/zkServer.sh status
```

```
[root@worker1 local]# /usr/local/zookeeper/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[root@worker2 local]# /usr/local/zookeeper/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: leader
```

```
[root@worker3 local]# /usr/local/zookeeper/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
```

三、部署Hadoop

3.1 在master上下载并解压Hadoop

```
cd /usr/local
wget https://github.com/nju-tcy/flink_files/raw/master/hadoop-3.2.0.tar.gz
tar -zxvf hadoop-3.2.0.tar.gz
```

建立软链接

```
ln -s hadoop-3.2.0 hadoop
```

3.2 添加Hadoop到环境变量

a.添加环境变量

```
vim /etc/profile
```

```
export HADOOP_HOME=/usr/local/hadoop
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

b.使环境变量生效

```
source /etc/profile
```

3.3 修改Hadoop配置文件

a.切换到配置文件目录

```
cd $HADOOP_HOME/etc/hadoop
```

b.修改hadoop-env.sh

```
echo "export JAVA_HOME=/usr/local/jdk8u252-b09" >> hadoop-env.sh
echo "export HDFS_NAMENODE_USER=root" >> hadoop-env.sh
echo "export HDFS_SECONDARYNAMENODE_USER=root" >> hadoop-env.sh
echo "export HDFS_DATANODE_USER=root" >> hadoop-env.sh
```

c.修改yarn-env.sh

```
echo "export YARN_REGISTRYDNS_SECURE_USER=root" >> yarn-env.sh
echo "export YARN_RESOURCEMANAGER_USER=root" >> yarn-env.sh
echo "export YARN_NODEMANAGER_USER=root" >> yarn-env.sh
```

d.修改core-site.xml

```
vim core-site.xml
```

在configuration标签范围内添加以下内容

```
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
</property>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/hadoop_tmp_dir</value>
</property>
<property>
   <name>ipc.client.connect.max.retries</name>
   <value>100</value>
</property>
<property>
   <name>ipc.client.connect.retry.interval</name>
   <value>10000</value>
</property>
<property>
   <name>hadoop.proxyuser.root.hosts</name>
   <value>*</value>
</property>
<property>
   <name>hadoop.proxyuser.root.groups</name>
   <value>*</value>
</property>
```

在**master**节点上创建相应目录

```
mkdir /home/hadoop_tmp_dir
```

e.修改hdfs-site.xml

```
vim hdfs-site.xml
```

添加以下内容

```
<property>
    <name>dfs.replication</name>
    <value>3</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>
    <value>/data/data1/hadoop/nn</value>
</property>
<property>
    <name>dfs.datanode.data.dir</name>
<value>/data/data1/hadoop/dn,/data/data2/hadoop/dn,/data/data3/hadoop/dn,/data/data4/ha
doop/dn,/data/data5/hadoop/dn,/data/data6/hadoop/dn,/data/data7/hadoop/dn,/data/data8/h
adoop/dn,/data/data9/hadoop/dn,/data/data10/hadoop/dn,/data/data11/hadoop/dn,/data/data
12/hadoop/dn</value>
</property>
<property>
    <name>dfs.http.address</name>
    <value>master:50070</value>
</property>
<property>
    <name>dfs.namenode.http-bind-host</name>
    <value>0.0.0.0</value>
</property>
<property>
    <name>dfs.datanode.handler.count</name>
    <value>600</value>
</property>
<property>
    <name>dfs.namenode.handler.count</name>
    <value>600</value>
</property>
<property>
    <name>dfs.namenode.service.handler.count</name>
    <value>600</value>
</property>
<property>
    <name>ipc.server.handler.queue.size</name>
    <value>300</value>
</property>
<property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
</property>
```

在**worker1**、**worker2**、**worker3**节点上创建相应目录

```
mkdir -p /data/data{1,2,3,4,5,6,7,8,9,10,11,12}/hadoop
```

f.修改mapred-site.xml

```
vim mapred-site.xml
```

在configuration标签范围内添加以下内容

```
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
    <final>true</final>
    <description>The runtime framework for executing MapReduce jobs</description>
</property>
<property>
    <name>mapreduce.job.reduce.slowstart.completedmaps</name>
    <value>0.88</value>
</property>
<property>
    <name>mapreduce.application.classpath</name>
    <value>
        /usr/local/hadoop/etc/hadoop,
        /usr/local/hadoop/share/hadoop/common/*,
        /usr/local/hadoop/share/hadoop/common/lib/*,
        /usr/local/hadoop/share/hadoop/hdfs/*,
        /usr/local/hadoop/share/hadoop/hdfs/lib/*,
        /usr/local/hadoop/share/hadoop/mapreduce/*,
        /usr/local/hadoop/share/hadoop/mapreduce/lib/*,
        /usr/local/hadoop/share/hadoop/yarn/*,
        /usr/local/hadoop/share/hadoop/yarn/lib/*
    </value>
</property>
<property>
    <name>mapreduce.map.memory.mb</name>
    <value>6144</value>
</property>
<property>
    <name>mapreduce.reduce.memory.mb</name>
    <value>6144</value>
 </property>
 <property>
    <name>mapreduce.map.java.opts</name>
    <value>-Xmx5530m</value>
</property>
<property>
    <name>mapreduce.reduce.java.opts</name>
    <value>-Xmx2765m</value>
</property>
<property>
    <name>mapred.child.java.opts</name>
    <value>-Xmx2048m -Xms2048m</value>
```

```
</property>
<property>
    <name>mapred.reduce.parallel.copies</name>
    <value>20</value>
</property>
<property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<property>
    <name>mapreduce.job.counters.max</name>
    <value>1000</value>
</property>
```

g.修改yarn-site.xml

```
vim yarn-site.xml
```

在configuration标签范围添加以下内容

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
    <final>true</final>
</property>
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>server1</value>
</property>
<property>
    <name>yarn.resourcemanager.bind-host</name>
    <value>0.0.0.0</value>
</property>
<property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>371200</value>
```

```
</property>
<property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>371200</value>
</property>
<property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>1024</value>
</property>
<property>
    <name>yarn.nodemanager.resource.cpu-vcores</name>
    <value>64</value>
</property>
<property>
    <name>yarn.scheduler.maximum-allocation-vcores</name>
    <value>64</value>
</property>
<property>
    <name>yarn.scheduler.minimum-allocation-vcores</name>
    <value>1</value>
</property>
<property>
    <name>yarn.log-aggregation-enable</name>
    <value>true</value>
</property>
<property>
    <name>yarn.client.nodemanager-connect.max-wait-ms</name>
    <value>300000</value>
</property>
<property>
    <name>yarn.nodemanager.vmem-pmem-ratio</name>
    <value>2.1</value>
</property>
<property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
</property>
<property>
    <name>yarn.nodemanager.pmem-check-enabled</name>
    <value>false</value>
</property>
<property>
    <name>yarn.application.classpath</name>
    <value>
        /usr/local/hadoop/etc/hadoop,
        /usr/local/hadoop/share/hadoop/common/*,
        /usr/local/hadoop/share/hadoop/common/lib/*,
        /usr/local/hadoop/share/hadoop/hdfs/*,
        /usr/local/hadoop/share/hadoop/hdfs/lib/*,
```

```
        /usr/local/hadoop/share/hadoop/mapreduce/*,
        /usr/local/hadoop/share/hadoop/mapreduce/lib/*,
        /usr/local/hadoop/share/hadoop/yarn/*,
        /usr/local/hadoop/share/hadoop/yarn/lib/*
    </value>
</property>
<property>
    <name>yarn.nodemanager.local-dirs</name>
<value>/data/data1/hadoop/yarn/local,/data/data2/hadoop/yarn/local,/data/data3/hadoop/yarn/local,/data/data4/hadoop/yarn/local,/data/data5/hadoop/yarn/local,/data/data6/hadoop/yarn/local,/data/data7/hadoop/yarn/local,/data/data8/hadoop/yarn/local,/data/data9/hadoop/yarn/local,/data/data10/hadoop/yarn/local,/data/data11/hadoop/yarn/local,/data/data12/hadoop/yarn/local</value>
    </property>
<property>
    <name>yarn.nodemanager.log-dirs</name>
<value>/data/data1/hadoop/yarn/log,/data/data2/hadoop/yarn/log,/data/data3/hadoop/yarn/log,/data/data4/hadoop/yarn/log,/data/data5/hadoop/yarn/log,/data/data6/hadoop/yarn/log,/data/data7/hadoop/yarn/log,/data/data8/hadoop/yarn/log,/data/data9/hadoop/yarn/log,/data/data10/hadoop/yarn/log,/data/data11/hadoop/yarn/log,/data/data12/hadoop/yarn/log</value>
</property>
<property>
    <name>yarn.timeline-service.enabled</name>
    <value>true</value>
</property>
<property>
    <name>yarn.timeline-service.hostname</name>
    <value>server1</value>
</property>
<property>
    <name>yarn.timeline-service.http-cross-origin.enabled</name>
    <value>true</value>
</property>
<property>
    <name>yarn.resourcemanager.system-metrics-publisher.enabled</name>
    <value>true</value>
</property>
```

在**worker1**、**worker2**、**worker3**节点上创建相应目录

```
mkdir -p /data/data{1,2,3,4,5,6,7,8,9,10,11,12}/hadoop/yarn
```

h.修改workers

```
vim workers
```

修改其内容为

```
worker1
worker2
worker3
```

3.4 同步配置到worker节点

a.在master节点创建journaldata目录

```
mkdir -p /usr/local/hadoop-3.2.0/journaldata
```

b.拷贝hadoop-3.2.0到worker1、worker2、worker3节点的"/usr/local"目录

```
scp -r /usr/local/hadoop-3.2.0 root@worker1:/usr/local
scp -r /usr/local/hadoop-3.2.0 root@worker2:/usr/local
scp -r /usr/local/hadoop-3.2.0 root@worker3:/usr/local
```

c.分别在worker1、worker2、worker3上为hadoop-3.2.0建立软链接

```
cd /usr/local
ln -s hadoop-3.2.0 hadoop
```

3.5 启动Hadoop集群

a.启动ZooKeeper集群

在worker1，worker2，worker3节点上启动ZooKeeper

```
/usr/local/zookeeper/bin/zkServer.sh start
```

b.启动JournalNode

在worker1，worker2，worker3节点上启动JournalNode

```
/usr/local/hadoop/sbin/hadoop-daemon.sh start journalnode
```

c.格式化hdfs

在**mater**节点上格式化hdfs

```
hdfs namenode -format
```

（注意：多次格式化hdfs会导致master节点与worker节点的cluster_id不一致，使worker节点上datanode进程无法正常启动）

d.在mater节点上格式化zkfc

```
hdfs zkfc -formatZK
```

e.启动hdfs

在**mater**节点上格启动hdfs

```
/usr/local/hadoop/sbin/start-dfs.sh
```

f.启动yarn

在**mater**节点上格式化yarn

```
/usr/local/hadoop/sbin/start-yarn.sh
```

g.检查进程是否正常启动

master

```
[root@master local]# jps
136387 Jps
136022 ResourceManager
1109 WrapperSimpleApp
134888 NameNode
135775 SecondaryNameNode
```

worker1、worker2、worker3(仅举例worker1)

```
[root@worker1 local]# jps
21842 NodeManager
18195 QuorumPeerMain
22219 Jps
1084 WrapperSimpleApp
18349 JournalNode
21134 DataNode
```

h.首次部署完毕之后启动集群的命令

在worker1、worker2、worker3上

```
/usr/local/zookeeper/bin/zkServer.sh start
/usr/local/hadoop/sbin/hadoop-daemon.sh start journalnode
```

然后在master上

```
/usr/local/hadoop/sbin/start-all.sh
```

四、部署Flink

4.1 在master节点上下载与解压Flink

```
cd /usr/local
wget https://github.com/nju-tcy/flink_files/raw/master/flink-1.12.2-bin-scala_2.11.tgz
tar -zxvf flink-1.12.2-bin-scala_2.11.tgz
```

建立软链接

```
ln -s flink-1.12.2 flink
```

4.2 添加Flink到环境变量

a.添加Flink到环境变量

```
vim /etc/profile
```

添加以下内容

```
export FLINK_HOME=/usr/local/flink
export PATH=$FLINK_HOME/bin:$PATH
```

b.使环境变量生效

```
source /etc/profile
```

4.3 修改Flink配置文件

a.切换到配置文件所在目录

```
cd $FLINK_HOME/conf
```

b.修改配置文件flink-conf.yaml

添加以下内容

```
env.java.home: /usr/local/jdk8u252-b09
env.hadoop.conf.dir: /usr/local/hadoop/etc/hadoop/
```

4.4运行验证

a.确保Zookeeper和Hadoop已经正常启动

b.在master上启动Flink

```
/usr/local/flink/bin/start-cluster.sh
```

```
[root@master ~]# /usr/local/flink/bin/start-cluster.sh
Starting cluster.
Starting standalonesession daemon on host master.
Starting taskexecutor daemon on host master.
```

（停止Flink的命令是）

```
/usr/local/flink/bin/stop-cluster.sh
```

c.查看进程是否正常启动

在master上

```
[root@master ~]# jps
2048 NameNode
4176 TaskManagerRunner
3909 StandaloneSessionClusterEntrypoint
1126 WrapperSimpleApp
4279 Jps
3192 ResourceManager
2937 SecondaryNameNode
```

五、测试Flink

```
cd /usr/local/flink
```

使用Flink项目自带的案例（单词统计程序）进行测试

```
bin/flink run examples/streaming/WordCount.jar --output out.txt
```

```
[root@master flink]# bin/flink run examples/streaming/WordCount.jar --output out.txt
Executing WordCount example with default input data set.
Use --input to specify file input.
Job has been submitted with JobID 020cadf9e12b227a096b436258881636
Program execution finished
Job with JobID 020cadf9e12b227a096b436258881636 has finished.
Job Runtime: 1746 ms
```

查看结果

```
cat out.txt
```

```
[root@master flink]# cat out.txt
(to,1)
(be,1)
(or,1)
(not,1)
(to,2)
(be,2)
```

六、问题汇总

6.1 免密登录问题

```
[root@master ~]# ssh-copy-id -i ~/.ssh/id_rsa.pub root@worker2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@worker2's password:
sh: .ssh/authorized_keys: Operation not permitted
```

解决办法：在目标节点上

```
chattr -i ~/.ssh/authorized_keys
```

6.2 Zookeeper无法启动的问题

```
[root@shixun zkData]# zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /opt/app/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost.
Error contacting service. It is probably not running.
```

参考链接：https://blog.csdn.net/zyd_994264926326/article/details/119807354

6.3 启动hdfs后master节点上没有namenode进程

参考链接：https://blog.csdn.net/web18484626332/article/details/126606658

6.4 启动hdfs后master节点上没有datanode进程

参考链接：https://blog.csdn.net/qq_45069279/article/details/111559319

（本实验VEISION文件所在目录/data/data1/hadoop/nn/current、/data/data1/hadoop/dn/current）

本文参考部署指南：https://www.hikunpeng.com/document/detail/zh/kunpengbds/ecosystemEnable/Flink/kunpengflink_04_0001.html