集群安装部署

1 安装前准备

1.1 准备服务器

10.10.60.204 master
10.10.60.205 slave1
10.10.60.206 slave2
10.10.60.207 slave3

分别在四台服务器上创建 hadoop、spark、utils 三个用户

1.2 配置 ip 与 hostname

用 root 用户修改每台机器的 hosts
Vi /etc/hosts
添加以下内容：
10.10.60.204 master
10.10.60.205 slave1
10.10.60.206 slave2
10.10.60.207 slave3

应用配置 source /etc/hosts

1.3 关闭防火墙

切换到管理员 su root

连接设备，键入命令“service iptables status”查看防火墙状态

关闭命令“chkconfig iptables off”，重启后生效。

1.4 ssh 免密码登录

- **SSH 检查**

首先确认系统已经安装 SSH，切换到管理员 su root

键入命令:

rpm -qa | grep openssh
rpm -qa | grep rsync
出现如下图信息表示已安装。

假设没有安装 ssh 和 rsync，可以通过下面命令进行安装。

安装 SSH 协议--> yum install ssh

安装 rsync 工具--> yum install rsync

启动服务--> service sshd restart

- **SSH 免密码登录的设置，每台设备的操作都是一样的，以 10.10.60.204 为例：**
  切换用户 su - hadoop，执行 ssh-keygen -t rsa 生成密钥，一直按回车就行

  进入.ssh 目录 cd ~/.ssh，执行：

  cp id_rsa.pub authorized_keys

  本机器测试，执行：

  ssh master

  如果没有提示输入密码表示免密登陆成功

  打包 .ssh 下的所有文件分发到各子节点

  tar -czvf ssh.tar.gz .ssh/

  scp ssh.tar.gz slave1:~/

  scp ssh.tar.gz slave2:~/

  scp ssh.tar.gz slave3:~/

  在各节点上执行 tar -xzvf ssh.tar.gz

- **常见问题**

  如果以上操作做完后，ssh 登录还需要密码，请检查文件和文件夹权限

  .ssh 目录和用户目录/hadoop 的权限必须是 700

  修改权限命令--> chmod 700 hadoop

  .ssh 目录下的 authorized_keys 文件的权限必须是 600

  修改权限命令--> chmod 600 authorized_keys

1.5 软件准备

所有软件已在在服务器/root/package 目录下
hadoop-2.7.4.tar.gz
jdk-8u144-linux-x64.tar.gz
scala-2.11.11.tgz
spark-2.2.0-bin-hadoop2.7.tgz
zookeeper-3.4.10.tar.gz
安装软件并配置环境变量
用 root 将 hadoop 解压到/home/hadoop 目录下
tar -xzvf hadoop-2.7.4.tar.gz
用 root 将 zookeeper 解压到/home/zookeeper 目录下
tar -xzvf zookeeper-3.4.10.tar.gz

用 root 将 spark 解压到/home/spark 目录下
tar -xzvf spark-2.2.0-bin-hadoop2.7.tgz
用 root 将 java 和 scala 解压到/home/utils 目录下

tar -xzvf jdk-8u144-linux-x64.tar.gz
tar -xzvf scala-2.11.11.tgz

1.6 配置环境/etc/profile 添加以下内容

```
export MAVEN_HOME=/home/utils/maven
export JAVA_HOEM=/home/utils/java
export SCALA_HOME=/home/utils/scala
export SPARK_HOME=/home/spark/spark
export HADOOP_HOME=/home/hadoop/hadoop
export ZK_HOME=/home/zookeeper/zookeeper
export PATH=${JAVA_HOME}/bin:${SCALA_HOME}/bin:${SPARK_HOME}/sbin:${HADOOP_HOME}/sbin:${SPARK_HOME}/bin:${HADOOP_HOME}/bin:${ZK_HOME}/bin:$PATH
```

2 配置 zookeeper

2.1 配置 zoo.cfg

```
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/home/zookeeper/data
clientPort=2181
server.1=slave1:2888:3888
server.2=slave2:2888:3888
server.3=slave3:2888:3888
```

3 配置 hadoop

1.5 修改 core-site.xml

```
<configuration>
<!-- 指定 hdfs 的 nameservice 为 bigdata -->
<property>
<name>fs.defaultFS</name>
<value>hdfs://bigdata/</value>
</property>
<!-- 指定 hadoop 临时目录 -->
<property>
<name>hadoop.tmp.dir</name>
<value>/home/hadoop/data/tmp</value>
</property>
```

```
<property>
<name>hadoop.data.dir</name>
<value>/home/hadoop/data/data</value>
</property>
<!-- 指定 zookeeper 地址 -->
<property>
<name>ha.zookeeper.quorum</name>
<value>slave1:2181,slave2:2181,slave3:2181</value>
</property>
</configuration>
```

1.6 修改 hadoo-env.sh

```
export JAVA_HOME=/home/utils/java
```

1.7 修改 hdfs-site.xml

```
<configuration>
<!--指定 hdfs 的 nameservice 为 bigdata，需要和 core-site.xml 中的保持一致 -->
<property>
<name>dfs.permissions.enabled</name>
<value>false</value>
</property>
<property>
<name>dfs.nameservices</name>
<value>bigdata</value>
</property>
<!-- bigdata（逻辑名）下面有两个 NameNode，分别是 nn1，nn2 -->
<property>
<name>dfs.ha.namenodes.bigdata</name>
<value>nn1,nn2</value>
</property>
<!-- nn1 的 RPC 通信地址 -->
<property>
<name>dfs.namenode.rpc-address.bigdata.nn1</name>
<value>master:9000</value>
</property>
<!-- nn1 的 http 通信地址 -->
<property>
<name>dfs.namenode.http-address.bigdata.nn1</name>
<value>master:50070</value>
</property>
<!-- nn2 的 RPC 通信地址 -->
<property>
<name>dfs.namenode.rpc-address.bigdata.nn2</name>
<value>slave1:9000</value>
</property>
<!-- nn2 的 http 通信地址 -->
<property>
<name>dfs.namenode.http-address.bigdata.nn2</name>
```

```
<value>slave1:50070</value>
</property>
<!-- 指定 NameNode 的 edits 元数据在 JournalNode 上的存放位置 -->
<property>
<name>dfs.namenode.shared.edits.dir</name>
<value>qjournal://slave1:8485;slave2:8485;slave3:8485/bigdata</value>
</property>
<!-- 指定 JournalNode 在本地磁盘存放数据的位置 -->
<property>
<name>dfs.journalnode.edits.dir</name>
<value>/home/hadoop/data/journaldata</value>
</property>
<!-- 开启 NameNode 失败自动切换 -->
<property>
<name>dfs.ha.automatic-failover.enabled</name>
<value>true</value>
</property>
<!-- 配置失败自动切换实现方式 -->
<property>
<name>dfs.client.failover.proxy.provider.bigdata</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</val
ue>
</property>
<!-- 配置隔离机制方法，多个机制用换行分割，即每个机制暂用一行-->
<property>
<name>dfs.ha.fencing.methods</name>
<value>
sshfence
shell(/bin/true)
</value>
</property>
<!-- 使用 sshfence 隔离机制时需要 ssh 免登陆 -->
<property>
<name>dfs.ha.fencing.ssh.private-key-files</name>
<value>/usr/local/.ssh/id_rsa</value>
</property>
<!-- 配置 sshfence 隔离机制超时时间 -->
<property>
<name>dfs.ha.fencing.ssh.connect-timeout</name>
<value>30000</value>
</property>
</configuration>
```

1.8 修改 yarn-site.xml

```
<configuration>
<!-- 开启 RM 高可用 -->
<property>
<name>yarn.resourcemanager.ha.enabled</name>
<value>true</value>
</property>
<!-- 指定 RM 的 cluster id -->
```

```
<property>
<name>yarn.resourcemanager.cluster-id</name>
<value>yrc</value>
</property>
<!-- 指定 RM 的名字 -->
<property>
<name>yarn.resourcemanager.ha.rm-ids</name>
<value>rm1,rm2</value>
</property>
<!-- 分别指定 RM 的地址 -->
<property>
<name>yarn.resourcemanager.hostname.rm1</name>
<value>master</value>
</property>
<property>
<name>yarn.resourcemanager.hostname.rm2</name>
<value>slave1</value>
</property>
<!-- 指定 zk 集群地址 -->
<property>
<name>yarn.resourcemanager.zk-address</name>
<value>slave1:2181,slave2:2181,slave3:2181</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

<property>
<name>yarn.scheduler.minimum-allocation-mb</name>
<value>1024</value>
</property>

<property>
<name>yarn.scheduler.maximum-allocation-mb</name>
<value>102400</value>
</property>
<property>
<name>yarn.scheduler.minimum-allocation-vcores</name>
<value>8</value>
</property>
<property>
<name>yarn.scheduler.maximum-allocation-vcores</name>
<value>32</value>
</property>
<property>
  <name>yarn.nodemanager.pmem-check-enabled</name>
  <value>false</value>
</property>

<property>
  <name>yarn.nodemanager.vmem-check-enabled</name>
  <value>false</value>
</property>
```

```
 <property>
  <name>yarn.nodemanager.resource.memory-mb</name>
  <value>204800</value>
 </property>

 <property>
  <name>yarn.nodemanager.resource.cpu-vcores</name>
  <value>32</value>
 </property>
</configuration>
```

1.9 修改 yarn-env.sh

```
export JAVA_HOME=/home/utils/java
```

1.10 修改 mapred-site.xml

```
<configuration>
 <!-- 指定 mr 框架为 yarn 方式 -->
 <property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
 </property>
 </configuration>
```

1.11 修改 slave

```
slave1
slave2
slave3
```

4 配置 spark

4.1 配置 spark-default.conf

```
spark.master                    spark://master:7077
spark.eventLog.enabled          true
spark.eventLog.dir              hdfs://master:9000/sparklog
spark.driver.memory             5g
```

4.2 配置 spark-env.sh

HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop

以上操作只在 10.10.60.204 上完成，完成后将对应文件同步到 slave1 slave2 slave3 三个节点

5 启动服务

1.12 启动 Zookeeper 集群

分别在 slave1，slave2，slave3 上启动 Zookeeper·

zkServer.sh start
启动成功后 slave1，slave2，slave3 会有 **QuorumPeerMain** 进程

1.13 创建命名空间（格式化 bigdata）

在 master 上执行：
hdfs zkfc -formatZK

1.14 格式化主 NameNode 节点（namenode1）

在 master 上执行命令：
hdfs namenode -format

1.15 启动主 NameNode 节点

在 master 上执行：
hadoop-daemon.sh start namenode
启动成功后 master 会有 **NameNode** 进程

1.16 格式备 NameNode 节点（namenode2）

在 slave1 上执行命令：
hdfs namenode -bootstrapStandby

1.17 启动备 NameNode 节点（namenode2）

在 slave1 上执行：
hadoop-daemon.sh start namenode
启动成功后 slave1 会有 **NameNode** 进程

1.18 启动集群

在 master 上执行
start-dfs.sh
启动成功后 master，slave1 上会有 **DFSZKFailoverController** 进程
slave1，slave2，slave3 会有 **JournalNode** 进程
master,slave1，slave2，slave3 会有 **DataNode** 进程

1.19 启动 YARN（namenode1）

在 master,slave1 上执行：
start-yarn.sh
mr-jobhistory-daemon.sh start historyserver

启动成功后 master,slave1 其中一台上会有 resourcemanager 进程
master,slave1，slave2，slave3 上会有 nodemanager 进程

1.20 查看 hdfs

hadoop fs -ls hdfs://bigdata/