实验环境搭建——hadoop 完全分布

软件准备：jdk1.8 安装包 + Hadoop2.8.5 安装包 + zookeeper3.4.12 安装包
硬件准备：至少 3 台物理机或虚拟机

前期准备

1.配置主机名

/etc/hosts 中添加 IP 和主机名，例如

xx.xx.xx.xx hadoop1

xx.xx.xx.xx hadoop2

xx.xx.xx.xx hadoop3

2.关闭防火墙

$ sudo ufw disable

查看防火墙确认状态为 inactive

$ sudo ufw status

3.安装 SSH 和配置免密登录 SSH

确认 linux 系统中是否已安装 openssh-server。若没有安装，使用下面这个命令安装。

$ sudo apt-get install openssh-server

装完之后，在本地生成无密码密钥对。

$ ssh-keygen -t rsa

运行中的询问都直接回车。id_rsa（私钥）和 id_rsa.pub（公钥），默认存储在"/home/用户名/.ssh"目录下。接着在本机上做如下配置，把 id_rsa.pub 追加到授权的 key 里面去。

$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

在其他两个节点上也安装 SSH 并生成公钥，分别将公钥复制到第一个节点上追加到 key 中

$ scp ~/.ssh/id_rsa.pub hadoop1:~/.ssh/id_rsa2.pub

$ cat ~/.ssh/id_rsa2.pub >> ~/.ssh/authorized_keys

将第一个节点的 authorized_keys 复制到其他两个节点

$ scp ~/.ssh/authorized_keys hadoop2:~/.ssh/authorized_keys

然后修改每个节点 ssh 配置文件"/etc/ssh/sshd_config"

$ sudo vim /etc/ssh/sshd_config

检查下面一行前面”#”注释是否取消掉：

PubkeyAuthentication yes # 启用公钥私钥配对认证方式

重启 ssh 服务，使配置生效。

$ service ssh restart

使用如下命令验证是否可以免密登录各节点，第一次登录需要密码

$ ssh <主机名称>

集群部署情况

	hadoop1	hadoop2	hadoop3
是否为 namenode	是	是	否
是否为 datanode	是	是	是
是否为 journalnode	是	是	是

是否为 zookeeper	是	是	是
是否为 zkfc	是	是	否
是否为 resourcemanager	是	否	否
是否为 nodemanager	是	是	是

注：java 和 hadoop 的安装过程与伪分布式一样，但是 hadoop 的配置信息不同。因此这里不再介绍 java 和 hadoop 的安装过程。通过 java -version 和 hadoop version 查看是否安装成功。

zookeeper 安装

先解压 zookeeper 安装包，使用命令

$ tar zxvf <zookeeper 安装包>

在当前目录中会生成 zookeeper 目录

然后把 zookeeper 目录移动到指定的路径下，这里移动到用户目录的 bigdata/下，使用命令

$ tar zxvf <zookeeper 目录>/ ~/bigdata/<zookeeper 目录>/

然后在用户目录下的.bashrc 文件中新增 zookeeper 环境变量

$ vim ~/.bashrc

在文件最后添加环境变量：

```
export ZOOKEEPER=/home/hadoop/bigdata/zookeeper-3.4.12
export PATH=$PATH:$ZOOKEEPER/bin
```

保存文件，使用下面命令使环境变量生效。

$ source ~/.bashrc

测试 zookeeper，使用下述命令出现 zookeeper 安装路径，表明安装完成。

$ zkServer.sh

配置 zookeeper

首先到 zookeeper 安装目录下的 conf 目录，把 zoo_sample.cfg 文件复制一份，命名成 zoo.cfg，

$ cp zoo_sample.cfg zoo.cfg

修改 zoo.cfg 配置，新增以下配置内容：

1.自定义 dataDir 和 dataLogDir 的路径。

```
dataDir=/home/hadoop/bigdata/zookeeper-3.4.12/ZKData
dataLogDir=/home/hadoop/bigdata/zookeeper-3.4.12/ZKLogs
```

2. 最后 3 行 server.A=B:C:D，其中 A 是一个数字，表示这是第几号 server。B 是该 server 所在的 IP 地址。C 配置该 server 和集群中的 leader 交换消息所使用的端口。D 配置选举 leader 时所使用的端口。

```
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1
server.1=hadoop1:2888:3888
server.2=hadoop2:2888:3888
server.3=hadoop3:2888:3888
```

然后创建配置中 dataDir 和 dataLogDir 所指的目录，在 dataDir 目录下创建一个 myid 文件，myid 文件内容为 server.A 的 A。（例如：server.1=hadoop1:2888:3888，则 hadoop1 中的 myid 文件内容为 1）。

$ mkdir ZKData $ mkdir ZKLogs

$ cd ZKData $ vim myid

配置完成以后使用如下命令启动 zookeeper。

```
$ zkServer.sh start
```

使用 jps 查看是否启动成功，出现 QuorumPeerMain 这个进程表示成功启动。

其他节点同理，最后通过下述命令可看到每个节点为 leader 还是 follower

```
$ zkServer.sh status
```

配置 hadoop

需要修改的配置文件为：hadoop-env.sh、yarn-env.sh、core-site.xml、mapred-site.xml、yarn-site.xml、hdfs-site.xml、slaves

1. hadoop-env.sh 和 yarn-env.sh 文件增加一行命令，配置 java 环境。

```
# The java implementation to use.
export JAVA_HOME=/usr/local/jdk1.8.0_181
```

2. core-site.xml 文件配置内容如下：

```
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://hadoopCluster</value>
                <description>HDFS namenode url</description>
        </property>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/home/hadoop/bigdata/hadoop-2.8.5/tmp</value>
                <description>hadoop nodes file directory</description>
        </property>
        <property>
                <name>ha.zookeeper.quorum</name>
                <value>hadoop1:2181,hadoop2:2181,hadoop3:2181</value>
                <description>ZooKeeper's url, the number must be odd and can't be smaller than 3.
        </property>
</configuration>
```

3. mapred-site.xml 文件的配置内容如下：

```
<configuration>
        <property>
                 <name>mapreduce.framework.name</name>
                 <value>yarn</value>
        </property>
        <property>
                 <name>mapreduce.map.memory.mb</name>
                 <value>4096</value>
        </property>
        <property>
                 <name>mapreduce.reduce.memory.mb</name>
                 <value>8192</value>
        </property>
</configuration>
```

4. yarn-site.xml 文件的配置如下：

```
<configuration>

<!-- Site specific YARN configuration properties -->
        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value>hadoop1</value>
        </property>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
</configuration>
```

5. hdfs-site.xml 的配置内容如下：

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>2</value>
        <description>the number of the same block</description>
    </property>
    <property>
        <name>dfs.nameservices</name>
        <value>hadoopCluster</value>
        <description>
            the name of HDFS cluster, notes the number of clusters can be larger than one.
        </description>
    </property>
    <property>
        <name>dfs.ha.namenodes.hadoopCluster</name>
        <value>hadoop1,hadoop2</value>
        <description>
            the name of namenodes, notes the number of namenodes can be larger than one.
        </description>
    </property>
    <property>
        <name>dfs.namenode.rpc-address.hadoopCluster.hadoop1</name>
        <value>hadoop1:9000</value>
        <description>the RPC url of the hadoop1 which is a namenode.</description>
    </property>
    <property>
        <name>dfs.namenode.http-address.hadoopCluster.hadoop1</name>
        <value>hadoop1:50070</value>
        <description>the HTTP url of the hadoop1 which is a namenode. </description>
    </property>
    <property>
        <name>dfs.namenode.rpc-address.hadoopCluster.hadoop2</name>
        <value>hadoop2:9000</value>
        <description>the RPC url of the hadoop2 which is a namenode. </description>
    </property>
    <property>
        <name>dfs.namenode.http-address.hadoopCluster.hadoop2</name>
        <value>hadoop2:50070</value>
        <description>the HTTP url of the hadoop2 which is a namenode. </description>
    </property>
    <property>
        <name>dfs.namenode.shared.edits.dir</name>
        <value>qjournal://hadoop1:8485;hadoop2:8485;hadoop3:8485/hadoopCluster</value>
```

```
        <description>A directory on shared storage between the multiple namenodes
            in an HA cluster. This directory will be written by the active and read
            by the standby in order to keep the namespaces synchronized. This directory
            does not need to be listed in dfs.namenode.edits.dir above. It should be
            left empty in a non-HA cluster.
        </description>
    </property>
    <property>
        <name>dfs.ha.automatic-failover.enabled</name>
        <value>true</value>
        <description>
            automatic-failover enabled or not, if enabled, when namenode is failure, another namenode will be active.
        </description>
    </property>
    <property>
        <name>dfs.client.failover.proxy.provider.hadoopCluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
        <description>
            when namenode is failure, which implementation class is responsible for performing the fault switch.
        </description>
    </property>
    <property>
        <name>dfs.ha.fencing.methods</name>
        <value>sshfence</value>
        <description>
            Once you need NameNode to switch, use the SSH method to operate.
        </description>
    </property>
    <property>
        <name>dfs.ha.fencing.ssh.private-key-files</name>
        <value>/home/hadoop/.ssh/id_rsa</value>
        <description>
            If the SSH is used for fault switching, the location of the key stored in the SSH communication is used.
        </description>
    </property>
    <property>
        <name>dfs.journalnode.edits.dir</name>
        <value>/home/hadoop/bigdata/workspace/journal</value>
        <description>
            journalnode's disk address
        </description>
```

```
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/home/hadoop/bigdata/workspace/namenode/name</value>
    </property>
    <property>
        <name>dfs.namenode.data.dir</name>
        <value>/home/hadoop/bigdata/workspace/namenode/data</value>
    </property>
    <property>
        <name>dfs.datanode.dir</name>
        <value>/home/hadoop/bigdata/workspace/dfs</value>
    </property>
</configuration>
```

6. slaves 文件的配置

```
hadoop1
hadoop2
hadoop3
```

测试

接下来测试 hadoop 是否搭建成功。

1. 测试 hdfs

先启动 zkfc，由于是第一次启动，因此在启动之前要先对 zkfc 格式化，命令如下

```
$ hdfs zkfc -formatZK
```

注：zkfc 格式化只需要在一个节点上调用就行。

然后使用下面的命令启动 zkfc，

```
$ hadoop-daemon.sh start zkfc
```

同样在其他需要启动 zkfc 的节点上调用该命令。

再尝试启动 journalnode，命令如下

```
$ hadoop-daemon.sh start journalnode
```

通过 jps 查看 journalnode 是否启动成功，出现 journalnode 进程表明启动成功。

其他节点同理。

再启动 namenode，同样是第一次启动 namenode，需要对 namenode 进行格式化。命令如下

```
$ hdfs namenode -format -clusterId hadoopCluster
```

然后启动 namenode，命令如下

```
$ hadoop-daemon.sh start namenode
```

通过 jps 查看 namenode 是否启动成功，出现 namenode 进程则表明成功。

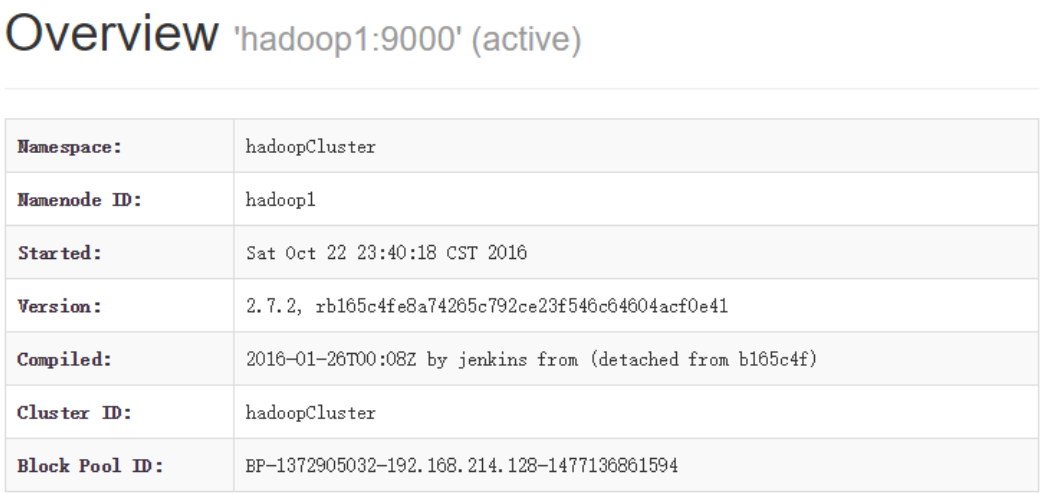
在另一个 namenode 节点上，也需要格式化 namenode，不过此时是从原来 namenode 同步过来。命令如下

```
$ hdfs namenode -bootstrapStandby
```

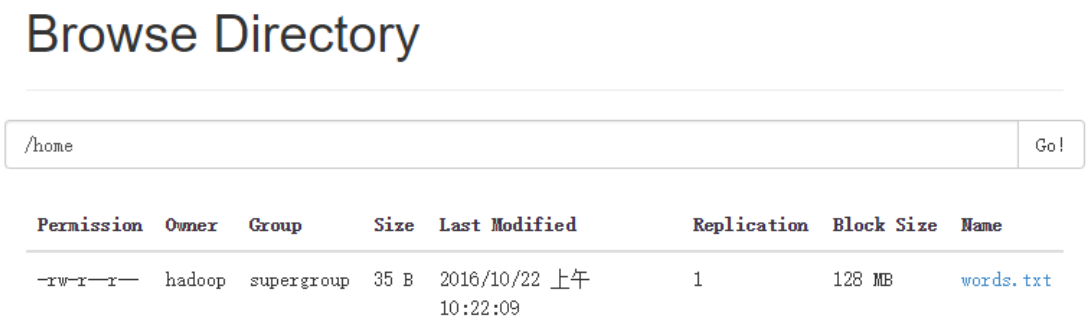
然后启动 namenode，命令如下

$ hadoop-daemon.sh start namenode
通过 jps 查看 namenode 是否启动成功，出现 namenode 进程则表明成功。

启动 datanode，命令如下
$ hadoop-daemons.sh start datanode
会同时启动所有节点的 datanode。然后通过 jps 查看是否成功。
全部节点都成功启动后，尝试使用浏览器访问 hdfs，访问地址为 **你的 ip:50070。**
正常情况下，网页如下图所示。



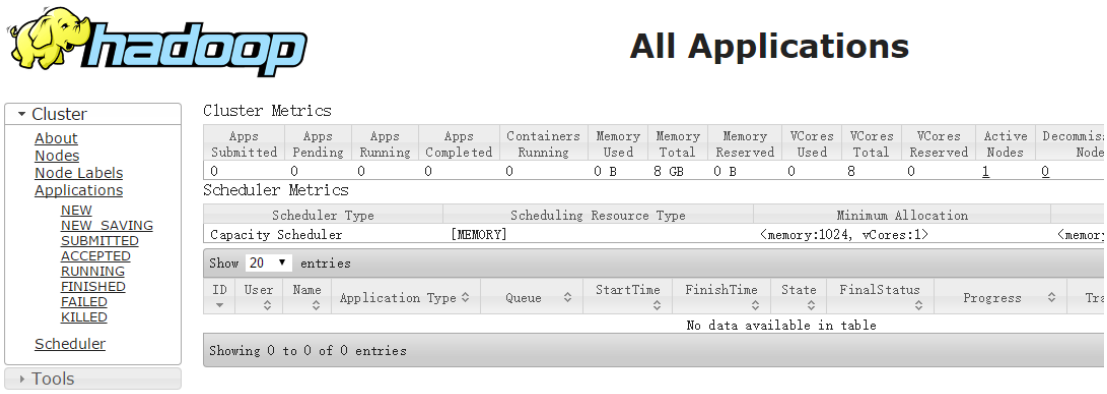
hdfs 上创建一个目录，命令如下图
$ Hadoop fs -mkdir /home
往/home 目录上传自己的文件，命令如下
$ hadoop fs -put words.txt /home



2. 测试 yarn
使用命令在 hadoop1（Resource manager 所在节点）上启动 yarn 的进程。
$ start-yarn.sh

启动后，同样使用 jps 查看 yarn 的进程是否启动成功，看到 nodemanager 和 resourcemanager，表明 yarn 启动成功。

通过浏览器查看，地址为 <你的 ip>:8088，效果如下图



表明 yarn 启动成功，现在可以在 yarn 上面跑 mapreduce 程序了。

这里使用现有的测试程序 wordcount 测试，

```
$ hadoop jar ~/bigdata/hadoop-2.8.5/share/hadoop/mapreduce/hadoop-mapreduce-example-2.8.5.jar wordcount hdfs://hadoop1:9000/home/words.txt hdfs://hadoop1:9000/out
```

跑完以后，结果存在/out/part-r-00000 里，可以查看运行结果

```
$ hadoop fs -cat /out/part-r-00000
```

```
hadoop@hadoop1:~/bigdata/hadoop-2.8.5$ hadoop fs -cat /out/part-r-00000
hello   4
word    2
world   2
```

可以看到 wordcount 的结果已经显示出来了。至此，完全分布的 hadoop 集群搭建完成。

参考文献：

https://hadoop.apache.org/docs/r2.8.5/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html