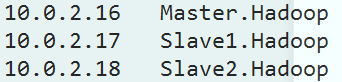
# Hadoop + Mahout集群搭建[服务器]

## 环境准备

1. 新建三台虚拟机(CentOS 7, 64位)，ip和host分别设为如下：



修改hosts文件

|  |
| --- |
| vi /etc/hosts  // 将以下数据复制进入各个主机中  10.0.2.16 Master.Hadoop  10.0.2.17 Slave1.Hadoop  10.0.2.18 Slave2.Hadoop |

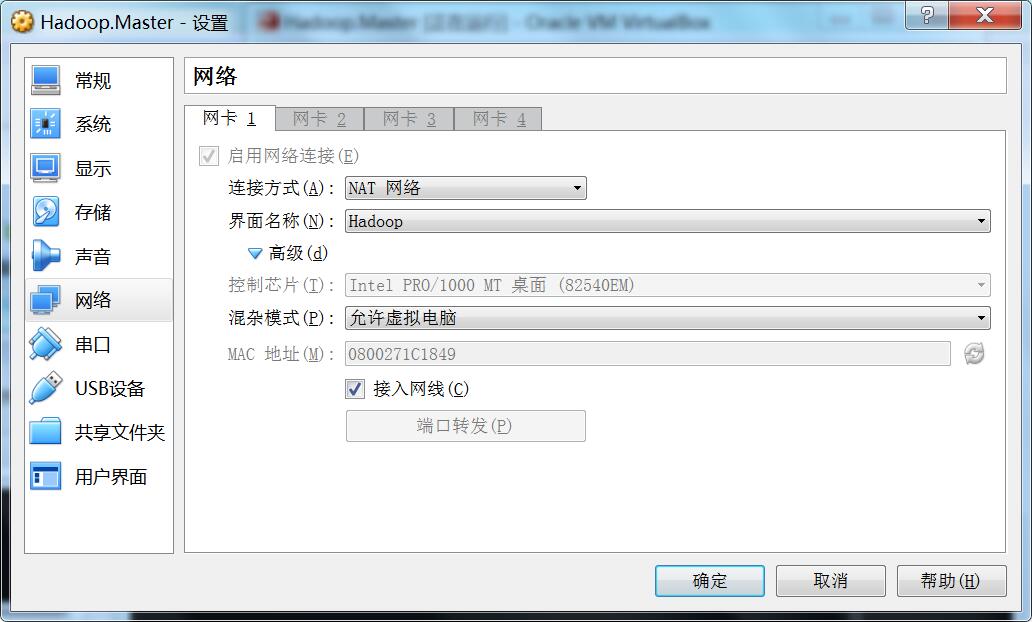
关闭防火墙（一定要！否则网络不通）：

|  |
| --- |
| systemctl stop firewalld |

1. Virtualbox->管理->全局设定->网络->NAT网络,新建一个名为Hadoop的NAT网络



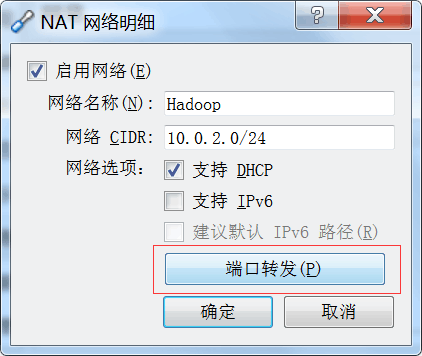
1. 三台虚拟机的网络设置为NAT网络，选择Hadoop网卡并启用混杂模式:



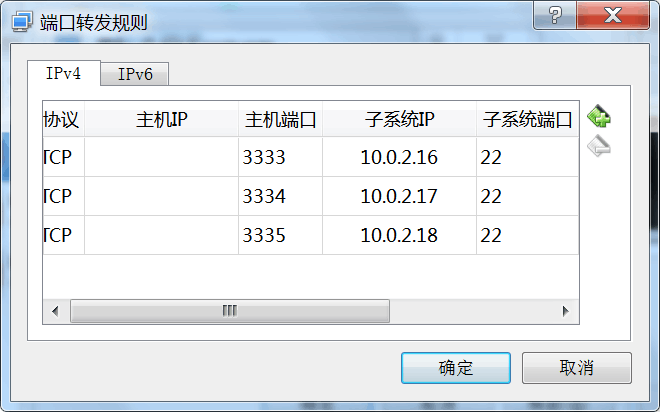
这样三台虚拟机在一个网络中可以互通了。

(以下端口转发设置可选，根据具体网络情况不同自行设置)

4. Virtualbox->管理->全局设定->网络->NAT网络，修改Hadoop网络，设置端口转发：



将3台虚拟机的22端口分别转发到3个端口上：

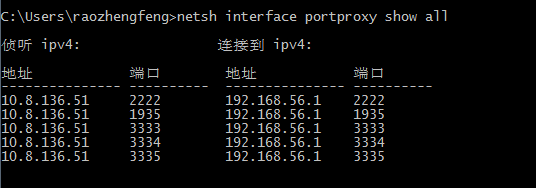


1. Cmd中运行如下命令:

|  |
| --- |
| netsh interface portproxy add v4tov4 listenaddress=10.8.13 6.51 listenport=3333 connectaddress=192.168.56.1 connectport=3333 |

再次将端口转发到外网，这样就可以在其他机器上通过ssh登录到虚拟机中了

查看端口转发设置:



### Hadoop编译安装

版本：Hadoop 2.7.3

安装步骤参考 <http://blog.csdn.net/circyo/article/details/46724335>

(如果是搭建伪分布式，一定要将本机也添加到etc/Hadoop/slaves中，否则将没有可用的datanode)

### Mahout安装

版本：0.12.2

安装步骤略

# Hadoop + Mahout开发环境搭建[Win7]

经过上面步骤已经在服务器中搭建了一个hadoop 2.7.3 + mahout 0.12.2 的集群，接下来我们要在开发机中使用mahout的api提交任务到集群中了

### Mahout编译

由于官方编译的mahout默认使用的是hadoop-client 2.4.1：

Mahout code depends on hadoop-client artifact, with the default version 2.4.1. To build Mahout against to a different hadoop version, hadoop.version property should be set accordingly and passed to the build command. Hadoop1 clients would additionally require hadoop1 profile to be activated.

跟我们服务器上的hadoop版本不一致，可能会存在问题，因此我们需要自己编译mahout：

1. 从[https://github.com/apache/mahout/releases](https://github.com/apache/mahout/releases%20下载0.12.2)  下载0.12.2的源码，解压到目录
2. Cmd中切换到代码根目录，执行如下命令：

|  |
| --- |
| mvn -DskipTests -Dhadoop2.version=2.7.3 clean package |

### 使用Mahout

1. 准备好开发环境:

IntelliJ IDEA 2016.1.1

添加hadoop环境变量:

HADOOP\_HOME：D:\soft\dev\hadoop-2.7.2

HADOOP\_BIN\_PATH：%HADOOP\_HOME%\bin

HADOOP\_PREFIX：%HADOOP\_HOME%

在Path后面加上%HADOOP\_HOME%\bin;%HADOOP\_HOME%\sbin;

17/03/08 18:07:06 ERROR security.UserGroupInformation: PriviledgedActionException as:raozhengfeng cause:java.io.IOException: Failed to set permissions of path: \tmp\hadoop-raozhengfeng\mapred\staging\raozhengfeng58824361\.staging to 0700

Exception in thread "main" java.io.IOException: Failed to set permissions of path: \tmp\hadoop-raozhengfeng\mapred\staging\raozhengfeng58824361\.staging to 0700

at org.apache.hadoop.fs.FileUtil.checkReturnValue(FileUtil.java:691)

./hdfs dfs -chmod -R 777 /tmp

FAILED: RuntimeException org.apache.**[Hadoop](http://lib.csdn.net/base/hadoop" \t "_blank" \o "Hadoop知识库)**.security.AccessControlException: org.apache.hadoop.security.AccessControlException: Permission denied: user=dbs, access=WRITE, inode="/opt/hadoop-1.0.1":hadoop:supergroup:drwxr-xr-x

在 hdfs-site.xml 总添加参数：

 <property>  
        <name>dfs.permissions</name>  
        <value>**false**</value>  
  </property>

然后重启hdfs

2017-03-08 05:57:27,460 WARN org.apache.hadoop.hdfs.protocol.BlockStoragePolicy: Failed to place enough replicas: expected size is 1 but only 0 storage types can be selected (replication=2, selected=[], unavailable=[DISK], removed=[DISK], policy=BlockStoragePolicy{HOT:7, storageTypes=[DISK], creationFallbacks=[], replicationFallbacks=[ARCHIVE]})

2017-03-08 05:57:27,460 WARN org.apache.hadoop.hdfs.server.blockmanagement.BlockPlacementPolicy: Failed to place enough replicas, still in need of 1 to reach 2 (unavailableStorages=[DISK], storagePolicy=BlockStoragePolicy{HOT:7, storageTypes=[DISK], creationFallbacks=[], replicationFallbacks=[ARCHIVE]}, newBlock=true) All required storage types are unavailable: unavailableStorages=[DISK], storagePolicy=BlockStoragePolicy{HOT:7, storageTypes=[DISK], creationFallbacks=[], replicationFallbacks=[ARCHIVE]}

<http://blog.csdn.net/szh1124/article/details/54928314>

Exception in thread "main" java.io.IOException: No FileSystem for scheme: hdfs

at org.apache.hadoop.fs.FileSystem.getFileSystemClass(FileSystem.java:2660)

at org.apache.hadoop.fs.FileSystem.createFileSystem(FileSystem.java:2667)

at org.apache.hadoop.fs.FileSystem.access$200(FileSystem.java:94)

pom.xml中添加

<**dependency**>  
 <**groupId**>org.apache.hadoop</**groupId**>  
 <**artifactId**>hadoop-hdfs</**artifactId**>  
 <**version**>2.7.3</**version**>  
</**dependency**>

18:03:46,281 WARN Thread-4 LocalJobRunner:run:560 - job\_local1339548899\_0001

org.apache.hadoop.security.AccessControlException: Permission denied: user=raozhengfeng, access=WRITE, inode="/tmp/1489053822371/preparePreferenceMatrix/itemIDIndex/\_temporary/0":hadoop:supergroup:drwxr-xr-x

at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:319)

hdfs-site.xml中增加：

<property>  
<name>dfs.permissions</name>  
<value>false</value>  
</property>

Mahout中基于mapreduce的只有item based和ALS算法，没有user based的算法，是因为没有必要。官方原因没找到，只有mahout实战的作者在quora上的回答：

"User-based" recommendation usually means recommending items to users based on user-user similarity. "Item-based" recommendation usually means recommending items to users based on item-item similarity. In both cases you're recommending items to users.  
  
RecommenderJob is item-based recommendation in this sense.  
  
If you turn around the input, you are recommending users to items. The strategy is still "item-based" although the semantics get quite muddled here since you're swapping users and items.

<https://www.quora.com/Apache-Mahout-Generating-user-based-recommendations-using-RecommenderJob>

也就是将输入的user id 和item id对调一下，然后使用item based算法，结果就是user based算法的结果。因为这两种算法的原理本来就一样嘛。。。

也就是说：我们只需要使用item based算法就够了。

Mahout 0.12.2支持的协同过滤算法列表：



