

# 实验一：环境搭建

Design by W.H Huang | Direct by Prof Feng

## 1 实验目的

本次实验预估耗时较长，因此将给出所有详细步骤，如若不能及时完成可在课后完成。

通过本次实验，你应该完成以下部分：

- 组内合作完成 Hadoop & Spark 单机版环境搭建
- 组内合作完成 Hadoop & Spark 分布式环境搭建

最终需搭建相关详细环境如下：

- 操作系统：centOS 7.6.64
- 图形界面：GNOME
- 语言环境：python 3.6.8
- 相关软件：Hadoop 2.8.5 、 Spark 2.4.4

## 2 实验准备

本次实验将详细介绍三种方式来搭建 Hadoop & Sapak 分布式环境：

- 云服务器分布式搭建
- 伪分布式搭建
- 多台机器分布式搭建

考虑到大家 IP 是动态分配（DHCP），没有使用固定IP。使用第三种方式 多台实际机器搭建 不方便。因此推荐大家使用前两种方式：云服务器分布式搭建、伪分布式搭建进行环境搭建。

## 3 云服务器分布式搭建

出于最简化演示目的，本次搭建将采用两台云服务器进行Hadoop+Spark 详细搭建记录。

☺ 如果小组成员>2，分布式搭建过程大同小异聪明如你应该知道怎么做。

首先记录下小组组员各自服务器的 内网IP&公网IP，例如我的：

主机名	内网IP	外网IP
master	172.30.0.7	129.28.154.240
slave01	172.16.0.4	134.175.210.3

### 3.1 Spark单机版搭建

⚠ 请注意，3.1.1 部分需在小组成员在**所有**云服务器上完成。3.1.2~3.1.4 小节只需在一台云服务器完成即可（作为master节点那台服务器）。

在进行Hadoop、Spark环境搭建前，我们需要进行一些准备工作。

### 3.1.1 准备工作

#### 1 配置用户

该小节主要是创建 Hadoop 用户。

##### 1. 创建用户

```
useradd -m hadoop -s /bin/bash
```

同时设置用户密码：（如 123456）

```
passwd hadoop
```

##### 2. 配置权限

为了方便，给用户 `hadoop` 等同 `root` 权限：

```
visudo # 执行 visudo命令进入vim编辑
```

找到如下位置，添加红框那一行配置权限：

```
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL
hadoop  ALL=(ALL)    ALL
```

##### 3. 切换用户

配置完成后，我们切换到hadoop用户下：

```
su hadoop # 注意，不要使用root用户，以下全部切换到hadoop用户下操作
```

△ 如非特殊说明，接下来所有命令都是Hadoop用户下完成！

#### 2 配置SSH

为什么要配置ssh？

因为集群、单节点模式都需要用到 ssh登陆。同时每次登陆ssh都要输入密码是件蛮麻烦的事，我可以通过生成公钥配置来面密码登陆。

##### 1. 生成密钥

为了生成 `~/.ssh` 目录，我们直接通过执行下面命令会直接生成

```
ssh localhost # 按提示输入yes，然后键入hadoop密码
```

然后开始生成密钥

```
cd ~/.ssh/ # 切换目录到ssh下
ssh-keygen -t rsa # 生成密钥
```

生成密钥过程会有三个提示，不用管全部回车。

## 2. 授权

```
cat id_rsa.pub >> authorized_keys # 加入授权
```

## 3. 修改权限

如果不修改文件 `authorized_keys` 权限为 600，会出现访问拒绝情况

```
chmod 600 ./authorized_keys # 修改文件权限
```

## 4. 测试

```
ssh localhost # ssh登陆
```

不用输入密码，直接登陆成功则说明配置正确。

```
[hadoop@VM_0_7_centos .ssh]$ ssh localhost
Last login: Thu Jan 23 17:05:19 2020 from 127.0.0.1
```

## 3 配置yum源

官方网站下载实在太慢，我们可以先配置一下阿里源来进行下载。

### 1. 切换到 `yum` 仓库

```
cd /etc/yum.repos.d/
```

### 2. 备份下原repo文件

```
sudo mv CentOS-Base.repo CentOS-Base.repo.backup
```

### 3. 下载阿里云repo文件

```
sudo wget -O /etc/yum.repos.d/CentOS-7.repo http://mirrors.aliyun.com/repo/Centos-7.repo
```

防止权限不足使用 `sudo` 命令。

### 4. 设置为默认repo文件

就是把阿里云repo文件名修改为 `CentOS-Base.repo`

```
sudo mv CentOS-7.repo CentOS-Base.repo # 输入y
```

### 5. 生成缓存

```
yum clean all
yum makecache
```

## 4 配置Java环境

最开始下载的是 1.7 版本的JDK，后面出现的问题，重新下载 1.8 版本 JDK。

hadoop2 基于 java 运行环境，所以我们要先配置java 运行环境。

### 1. 安装JDK

执行下面命令，经过实际测试前面几分钟一直显示镜像错误不可用。它会进行自己尝试别的源，等待一会儿就可以下载成功了。

```
sudo yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel
```

△ 此时默认安装位置是 /usr/lib/jvm/java-1.8.0-openjdk

其实，查找安装路径，可以通过以下命令：

```
rpm -ql java-1.8.0-openjdk-devel | grep '/bin/javac'
```

- rpm -ql <RPM包名>：查询指定RPM包包含的文件
- grep <字符串>：搜索包含指定字符的文件

### 2. 配置环境变量

```
vim ~/.bashrc # vim编辑配置文件
```

在文件最后面添加如下单独一行（指向JDK的安装位置），并保存：

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
```

```
# add chrome path
export PATH=$PATH:/opt/google/chrome
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk
```

最后是环境变量生效，执行：

```
source ~/.bashrc
```

### 3. 测试

```
echo $JAVA_HOME # 检验变量值
```

正常会输出 2. 环境变量JDK配置路径。

```
java -version
```

正确配置会输出java版本号。

## 5 安装python

CentOS自带python2版本过低，我们进行python3安装。

### 1. yum查找python3

查找仓库存在的python3安装包

```
yum list python3
```

```
python3.x86_64
```

```
3.6.8-10.el7
```

```
@base
```

## 2. yum 安装python3

```
sudo yum install python3.x86_64
```

如果最开始会显示没有，等一会自动切换阿里源就可以进行安装了，同时还会安装相关依赖。

## 3.1.2 hadoop 安装

本文使用 `wget` 命令来下载 `hadoop`：[了解更多wget](#)

使用的是[北理工镜像站](#)，下载 `hadoop`：

# Index of /apache/hadoop/common/hadoop-2.8.5

Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>	-	-	-
 <a href="#">hadoop-2.8.5-src.tar.gz</a>	2018-09-18 23:47	34M	
 <a href="#">hadoop-2.8.5.tar.gz</a>	2018-09-18 23:47	235M	

### 1. 下载

```
sudo wget -O hadoop-2.8.5.tar.gz https://mirrors.cnnic.cn/apache/hadoop/common/hadoop-2.8.5/hadoop-2.8.5.tar.gz
```

o `wget -O <指定下载文件名> <下载地址>`

### 2. 解压

```
sudo tar -zxf hadoop-2.8.5.tar.gz -C /usr/local
```

把下载好的文件 `hadoop-2.8.5.tar.gz` 解压到 `/usr/local` 目录下

### 3. 修改文件

```
cd /usr/local/ # 切换到解压目录下
sudo mv ./hadoop-2.8.5/ ./hadoop # 将解压的文件hadoop-2.8.5重命名为hadoop
sudo chown -R hadoop:hadoop ./hadoop # 修改文件权限
```

### 4. 测试

```
cd /usr/local/hadoop # 切换到hadoop目录下
./bin/hadoop version # 输出hadoop版本号
```

```
[hadoop@VM_0_7_centos hadoop]$ ./bin/hadoop version
Hadoop 2.8.5
```

## 3.1.3 spark安装

在前我们已经安装了 *hadoop*，现在我们来开始进行 *spark* 安装。

这次下载根据官网推荐使用的清华源。

## 1. 下载

官网下载地址：[官网下载](#)

### Download Apache Spark™

1. Choose a Spark release: 2.4.4 (Aug 30 2019)
2. Choose a package type: Pre-built with user-provided Apache Hadoop
3. Download Spark: [spark-2.4.4-bin-without-hadoop.tgz](#)
4. Verify this release using the 2.4.4 [signatures](#), [checksums](#) and [project release KEYS](#).

- 这样选择的版本可以使用于大部分 *hadoop* 版本

点击上述链接，根据跳转的页面提示选择清华源下载：

```
sudo wget -O spark-2.4.4-bin-without-hadoop.tgz
http://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-2.4.4/spark-2.4.4-bin-without-
hadoop.tgz
```

## 2. 解压

同前解压到 */usr/local* 目录下

```
sudo tar -zxf spark-2.4.4-bin-without-hadoop.tgz -C /usr/local
```

## 3. 设置权限

```
cd /usr/local # 切换到解压目录
sudo mv ./spark-2.4.4-bin-without-hadoop ./spark # 重命名解压文件
sudo chown -R hadoop:hadoop ./spark # 设置用户hadoop为目录spark拥有者
```

## 4. 配置spark环境

先切换到 */usr/local/spark*，（为了防止没权限，下面用 *sudo*）

```
cd /usr/local/spark
cp ./conf/spark-env.sh.template ./conf/spark-env.sh
```

编辑 *spark-env.sh* 文件：

```
vim ./conf/spark-env.sh
```

在第一行添加下面配置信息，使得Spark可以从Hadoop读取数据。

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

## 5. 配置环境变量

```
vim ~/.bashrc
```

在 .bashrc 文件中添加如下内容:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk # 之前配置的java环境变量
export HADOOP_HOME=/usr/local/hadoop # hadoop安装位置
export SPARK_HOME=/usr/local/spark
export PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/lib/py4j-0.10.7-
src.zip:$PYTHONPATH
export PYSARK_PYTHON=python3 # 设置pyspark运行的python版本
export PATH=$HADOOP_HOME/bin:$SPARK_HOME/bin:$PATH
```

最后为了使得环境变量生效, 执行:

```
source ~/.bashrc
```

## 6. 测试是否运行成功

```
cd /usr/local/spark
bin/run-example SparkPi
```

执行会输出很多信息, 也可以选择执行:

```
bin/run-example SparkPi 2>&1 | grep "Pi is"
```

```
[hadoop@VM_0_7_centos spark]$ bin/run-example SparkPi 2>&1 | grep "Pi is"
Pi is roughly 3.1462557312786563
```

## 3.1.4 测试

### 1. 启动pyspark

```
cd /usr/local/spark
bin/pyspark
```

```
[hadoop@VM_0_7_centos spark]$ bin/pyspark
Python 3.6.8 (default, Aug 7 2019, 17:28:10)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
Type "help", "copyright", "credits" or "license" for more information.
20/01/23 17:33:19 WARN util.Utils: Your hostname, VM_0_7_centos resolves to a loopback add
ress: 127.0.0.1; using 172.30.0.7 instead (on interface eth0)
20/01/23 17:33:19 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another addre
ss
20/01/23 17:33:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for you
r platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel
).
Welcome to

  ____      _
 / ___|  _ \| | | |
 \___ \| |_) | |_| |
  ___) | |_) | | | |
 |____|_|_|\___|_|_|_|

 version 2.4.4
```

## 2. 简单测试

```
>>> 8 * 2 + 5
```

```
>>> 2019+1*1  
2020
```

使用 `exit()` 命令可退出。

## 3.2 Hadoop+Spark 分布式环境搭建

### 3.2.1 准备工作

#### 1 修改主机名

两台服务器一台作为master，一台作为slave。为了以示区分，我们分别修改它们的主机名：

- 在master

```
sudo vim /etc/hostname
```

编辑修改为：`master`

- 分别在 slave01 & slave02

```
sudo vim /etc/hostname
```

分别编辑修改为：`slave01`、`slave02`

最后使用命令 `sudo reboot` 重启，便会生效。

#### 2 修改host

修改hosts目的：可以使用云服务器名字访问，而不直接使用IP地址

首先上自己的云服务器，记录下三台服务器的 内网IP&公网IP

主机名	内网IP	外网IP
master	172.30.0.7	129.28.154.240
slave01	172.16.0.4	134.175.210.3

⚠ 警告，下面有个史前大坑。因为云服务器默认访问本身是用内网IP地址

- 在master上

```
su hadoop  
sudo vim /etc/hosts
```

编辑hosts文件如下（以前的全部删除，改成下面这样）：



```
127.0.0.1 localhost
172.30.0.7 master      # master必须用内网IP
134.175.210.3 slave01  # slave01用外网IP
```

- 在 slave01 上

```
su hadoop
sudo vim /etc/hosts
```

```
127.0.0.1 localhost
129.28.154.240 master    # master必须用外网IP
172.16.0.4    slave01    # slave01用内网IP
```

### 3 SSH互相免密

在之前我们搭建Spark单机版环境时，我们配置ssh可以 *无密码* 本地连接：

```
ssh localhost    # 保证两台服务器都可以本地无密码登陆
```

现在我们还要让 master 主机免密码登陆 slave01、slave02。因此我们要将master主机的 `id_rsa.pub` 分别传递给两台slave主机。

1. 在 `master` 上scp传递公钥

第一次传要输入slave01@hadoop用户密码，例如之前设置为123456

```
scp ~/.ssh/id_rsa.pub hadoop@slave01:/home/hadoop/
```

2. 在slave01上加入验证

```
ls /home/hadoop/    # 查看master传送过来的 id_rsa.pub文件
```

将master公钥加入免验证：

```
cat /home/hadoop/id_rsa.pub >> ~/.ssh/authorized_keys
rm /home/hadoop/id_rsa.pub
```

3. 测试

现在我们切换到master主机上，尝试能否免密登陆：

```
[hadoop@master ~]$ ssh slave01
Last failed login: Thu Jan 23 17:44:52 CST 2020 from 129.28.154.240 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Thu Jan 23 17:44:31 2020 from 127.0.0.1
```

验证可以免密登陆后切换回master主机

```
ssh master    # 要输入master@hadoop用户密码
```

### 3.2.2 Hadoop集群配置

原本我们需要同时在master和slave节点安装配置Hadoop集群，但是我们也可以通过仅配置master节点Hadoop，然后将整个配置好的Hadoop文件传递给各个子节点。

## master节点配置

我们需要修改master主机上hadoop配置文件。

### 1. 切换目录

配置文件在 `/usr/local/hadoop/etc/hadoop` 目录下：

```
cd /usr/local/hadoop/etc/hadoop
```

```
[hadoop@master hadoop]$ ll
total 160
-rw-r--r-- 1 hadoop hadoop 4942 Sep 10 2018 capacity-scheduler.xml
-rw-r--r-- 1 hadoop hadoop 1335 Sep 10 2018 configuration.xsl
-rw-r--r-- 1 hadoop hadoop 318 Sep 10 2018 container-executor.cfg
-rw-r--r-- 1 hadoop hadoop 1085 Jan 23 17:51 core-site.xml
-rw-r--r-- 1 hadoop hadoop 3804 Sep 10 2018 hadoop-env.cmd
-rw-r--r-- 1 hadoop hadoop 4666 Sep 10 2018 hadoop-env.sh
```

### 2. 修改文件 `slaves`

master主机作为 `NameNode`，而 slave01 作为 `DataNode`

```
vim slaves
```

修改如下：

```
slave01
```

### 3. 修改文件 `core-site.xml`

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

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/hadoop/tmp</value>
    <description>Abase for other temporary directories.</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
```

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>file:/usr/local/hadoop/tmp</value>
    <description>Abase for other temporary directories.</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
```

#### 4. 修改 `hdfs-site.xml` :

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

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>mapred.job.tracker</name>
    <value>master:9001</value>
  </property>
  <property>
    <name>dfs.namenode.http-address</name>
    <value>master:50070</value>
  </property>
</configuration>
```

#### 5. 修改 `mapred-site.xml.template`

⚠ 首先复制它产生一个新复制文件并命名为: `mapred-site.xml`

```
cp mapred-site.xml.template mapred-site.xml
```

然后修改文件 `vim mapred-site.xml`:

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

#### 6. 修改 `yarn-site.xml`

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

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

## slave节点配置

当然你也可以尝试在slave节点上重复一遍master节点上的配置，而非通过传送文件。

- 在master节点上执行

```
cd /usr/local/
rm -rf ./hadoop/tmp      # 删除临时文件
rm -rf ./hadoop/logs/*   # 删除日志文件
# 压缩./hadoop文件，并重名为hadoop.master.tar.gz
tar -zcf ~/hadoop.master.tar.gz ./hadoop
```

将压缩好的文件传递给 slave01：

```
cd ~
scp ./hadoop.master.tar.gz slave01:/home/hadoop
```

😊 传递速度有点慢，大概要半小时。等待时间你可以先撰写部分实验报告，或者尝试浏览接下来实验步骤。

- 在slave01节点上  
(如果有) 删除原有hadoop文件夹

```
sudo rm -rf /usr/local/hadoop/
```

解压传过来的文件到指定目录 `/usr/local`：

```
sudo tar -zxf /home/hadoop/hadoop.master.tar.gz -C /usr/local
```

设置解压出来的hadoop文件夹权限：

```
sudo chown -R hadoop /usr/local/hadoop
```

## 集群启动测试

1. master上启动集群

```
cd /usr/local/hadoop
bin/hdfs namenode -format # 注意, 仅在第一次启动集群时使用该命令格式化!
sbin/start-all.sh
```

## 2. 测试

- 在master上

```
jps
```

master节点出现以下4个进程则配置成功:

```
[hadoop@master hadoop]$ jps
13191 NameNode
13869 Jps
13598 ResourceManager
13438 SecondaryNameNode
```

- 在 slave01上

```
jps
```

slave节点出现以下3个进程则配置成功:

```
[hadoop@slave01 ~]$ jps
5384 DataNode
5689 Jps
5549 NodeManager
```

## 常见问题汇总

Q1: slave 节点没有 DataNode 进程 / master 节点没有 namenode 进程 ?

这个问题一般是由于在启动集群多次执行格式化命令:

```
bin/hdfs namenode -format
```

导致 hodoop 目录下 tmp/dfs/name/current 文件下的 VERSION 中的 namespaceId 不一致。

首先我们 在master节点上 停止集群:

```
cd /usr/local/hadoop # 切换到你的hadoop目录下
sbin/stop-all.sh # 关闭集群
```

- slave 节点删除 tmp

删除slave节点的临时 tmp 文件

```
cd /usr/local # 切换到hadoop目录
rm -rf ./hadoop/tmp
```

删除 tmp 文件, 如法炮制在 其它节点 进行一样的操作:

```
rm -rf ./hadoop/tmp # 后面格式化会重新生成, 大胆删除
```

- 在master节点删除 tmp

```
cd /usr/local
rm -rf ./hadoop/tmp
```

- 重新启动集群

在master节点执行以下操作：

```
cd /usr/local/hadoop
bin/hdfs namenode -format # 重新格式化
sbin/start-all.sh
```

- 验证

在 master 节点执行以下操作：

```
cd ~
jps
```

```
[root@master hadoop]# jps
30496 NameNode
30882 ResourceManager
30717 SecondaryNameNode
621 Master
32127 Jps
```

在子节点再次输入 jps 命令：

```
cd ~
jps
```

```
[root@slave02 ~]# jps
5794 Jps
5613 NodeManager
5503 DataNode
```

ok~

Q2：启动集群后发现，slave 节点没有 NodeManager 进程

```
[hadoop@slave01 local]$ jps
30673 DataNode
31172 Jps
```

少了nodemanager进程

⚠ 建议先尝试 Q1 方法，一般能解决大部分问题。

启动集群时可以知道，启动 slave01 节点 notemanager 进程相关日志在（最后不是 .out 是 .log）：

/usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.log

```
[hadoop@master hadoop]$ sbin/start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-namenode-master.out
slave01: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-datanode-slave01.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-secondarynamenode-master.out
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-resourcemanager-master.out
slave01: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.out
```

## 1. 查看日志

在 `slave01` 节点下

```
vim /usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.log
```

日志太多，我们在命令模式下，输入 `:$`，直接跳到最后一行：

```
2020-01-31 18:12:33,973 INFO org.apache.hadoop.service.AbstractService: Service org.apache.hadoop.yarn.server.nodemanager.containermanager.localizer.ResourceLocalizationService failed in state STARTED; cause: org.apache.hadoop.yarn.exceptions.YarnRuntimeException: java.net.BindException: Problem binding to [0.0.0.0:8040] java.net.BindException: Address already in use; For more details see: http://wiki.apache.org/hadoop/BindException
org.apache.hadoop.yarn.exceptions.YarnRuntimeException: java.net.BindException: Problem binding to [0.0.0.0:8040] java.net.BindException: Address already in use; For more details see: http://wiki.apache.org/hadoop/BindException
    at org.apache.hadoop.yarn.factories.impl.pb.RpcServerFactoryPBImpl.getServer(RpcServerFactoryPBImpl.java:138)
```

- 很显然，显示端口 8040 被占用

## 2. 查看谁占用 8040 端口

```
netstat -tln | grep 8040
```

```
[hadoop@slave01 local]$ netstat -tln | grep 8040
tcp6      0      0 :::8040          :::*              LISTEN
```

果然 8040 端口已经被占用

## 3. 释放端口

```
lsof -i :8040 # 查询占用8040端口进程pid
```

```
[hadoop@slave01 local]$ lsof -i :8040
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF  NODE  NAME
java    16961 hadoop 231u IPv6 38843230      0t0  TCP *:ampify (LISTEN)
```

杀死相应进程：

```
kill -9 16961
```

## 4. 测试

重新启动集群

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

再次输入 `jps` 命令，发现 `slave01` 节点 `NodeManager` 进程已经出现！

```
[hadoop@slave01 local]$ jps
30673 DataNode
6419 Jps
6035 NodeManager
```

### 3.2.3 Spark集群配置

以下步骤都建立在是我们三台云服务器已经搭建好Spark单机版环境 & hadoop集群。

#### Spark配置

##### 1. 切换配置目录

```
cd /usr/local/spark/conf
```

##### 2. 配置 `slaves` 文件

```
cp slaves.template slaves # 先把模板文件复制重命名
```

开始编辑 `vim slaves`，将默认内容 `localhost` 替换为以下：

```
slave01
```

##### 3. 配置 `spark-env.sh` 文件

```
cp spark-env.sh.template spark-env.sh
```

开始编辑，添加下面内容：

```
vim spark-env.sh
```

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export SPARK_MASTER_IP=172.30.0.7 # 注意，使用的master内网IP!!
```

##### 4. 复制Spark文件到各个slave节点

```
cd /usr/local/
tar -zcf ~/spark.master.tar.gz ./spark
cd ~
scp ./spark.master.tar.gz slave01:/home/hadoop
```

##### 5. 节点替换文件

以下操作是在 slave节点上：



```
sudo rm -rf /usr/local/spark/          # 删除节点原有Spark文件 (如果有)
sudo tar -zxvf /home/hadoop/spark.master.tar.gz -C /usr/local # 解压到local
sudo chown -R hadoop /usr/local/spark  # 设置spark文件权限拥有者是hadoop
```

## 启动Spark集群

在master主机上执行以下操作

### 1. 先启动hadoop集群

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

### 2. 启动master节点

```
cd /usr/local/spark/
sbin/start-master.sh
```

master上运行 `jps` 命令可以看到:

```
[hadoop@master spark]$ jps
18160 Master
18224 Jps
13191 NameNode
13598 ResourceManager
13438 SecondaryNameNode
```

### 3. 启动所有slave节点

```
sbin/start-slaves.sh
```

slave节点上运行 `jps` 命令可以看到:

```
[hadoop@slave01 ~]$ jps
9527 Jps
5384 DataNode
5549 NodeManager
9471 Worker
```

### 4. web UI查看

在浏览器上输入: `master:8080`, 如果出现下面界面则表示 *Hadoop+Spark* 分布式环境搭建成功!

Applications Places Firefox 中 Thu 22:57

Spark Master at spark://master:7077 - Mozilla Firefox

Spark Master at spark://m x +

master:8080

Spark 2.4.4 Spark Master at spark://master:7077

URL: spark://master:7077  
Alive Workers: 1  
Cores in use: 1 Total, 0 Used  
Memory in use: 1024.0 MB Total, 0.0 B Used  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20200123225332-172.16.0.4-40359	172.16.0.4:40359	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)

🎉🎉 聪明如你终于做到这步了，第一个实验完结，撒花 🎉🎉

## 4 伪分布式搭建

😊 选择伪分布式搭建的同学，**每一个组员**都需要在各自服务器上**独立完成**环境搭建。

### 4.1 Spark单机版搭建

在进行Hadoop、Spark环境搭建前，我们需要进行一些准备工作。

#### 4.1.1 准备工作

##### 1 配置用户

该小节主要是创建 Hadoop 用户。

##### 1. 创建用户

```
useradd -m hadoop -s /bin/bash
```

同时设置用户密码：（如 123456）

```
passwd hadoop
```

##### 2. 配置权限

为了方便，给用户 hadoop 等同 root 权限：

```
visudo # 执行 visudo命令进入vim编辑
```

找到如下位置，添加红框那一行配置权限：

```
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
hadoop  ALL=(ALL)        ALL
```

### 3. 切换用户

配置完成好，我们切换到hadoop用户下：

```
su hadoop
```

△ 如非特殊说明，接下来所有命令都是Hadoop用户下完成！

## 2 配置SSH

为什么要配置ssh？

因为集群、单节点模式都需要用到 ssh登陆。同时每次登陆ssh都要输入密码是件蛮麻烦的事，我可以通过生成公钥配置来面密码登陆。

### 1. 生成密钥

为了生成 ~/.ssh 目录，我们直接通过执行下面命令会直接生成

```
ssh localhost # 按提示输入yes，然后键入hadoop密码
```

然后开始生成密钥

```
cd ~/.ssh/ # 切换目录到ssh下
ssh-keygen -t rsa # 生成密钥
```

生成密钥过程会有三个提示，不用管全部回车。

### 2. 授权

```
cat id_rsa.pub >> authorized_keys # 加入授权
```

### 3. 修改权限

如果不修改文件 `authorized_keys` 权限为 `600`，会出现访问拒绝情况

```
chmod 600 ./authorized_keys # 修改文件权限
```

### 4. 测试

```
ssh localhost # ssh登陆
```

不用输入密码，直接登陆成功则说明配置正确。

```
[hadoop@huang .ssh]$ ssh localhost
Last login: Fri Jan 24 11:16:06 2020 from 127.0.0.1
```

## 3 配置yum源

官方网站下载实在太慢，我们可以先配置一下阿里源来进行下载。

#### 1. 切换到 yum 仓库

```
cd /etc/yum.repos.d/
```

#### 2. 备份下原repo文件

```
sudo mv CentOS-Base.repo CentOS-Base.repo.backup
```

#### 3. 下载阿里云repo文件

```
sudo wget -O /etc/yum.repos.d/CentOS-7.repo http://mirrors.aliyun.com/repo/Centos-7.repo
```

防止权限不足使用 `sudo` 命令。

#### 4. 设置为默认repo文件

就是把阿里云repo文件名修改为 `CentOS-Base.repo`

```
sudo mv CentOS-7.repo CentOS-Base.repo # 输入y
```

#### 5. 生成缓存

```
yum clean all  
yum makecache
```

### 4 配置Java环境

最开始下载的是 1.7 版本的JDK，后面出现的问题，重新下载 1.8 版本 JDK。

*hadoop2* 基于 *java* 运行环境，所以我们要先配置*java* 运行环境。

#### 1. 安装 JDK

执行下面命令，经过实际测试前面几分钟一直显示镜像错误不可用。它会进行自己尝试别的源，等待一会儿就可以下载成功了。

```
sudo yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel
```

△ 此时默认安装位置是 `/usr/lib/jvm/java-1.8.0-openjdk`

其实，查找安装路径，可以通过以下命令：

```
rpm -ql java-1.8.0-openjdk-devel | grep '/bin/javac'
```

- `rpm -ql <RPM包名>`：查询指定RPM包包含的文件
- `grep <字符串>`：搜索包含指定字符的文件

#### 2. 配置环境变量

```
vim ~/.bashrc # vim编辑配置文件
```

在文件最后面添加如下单独一行（指向JDK的安装位置），并保存：

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
```

```
# add chrome path
export PATH=$PATH:/opt/google/chrome
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk
```

最后是环境变量生效，执行：

```
source ~/.bashrc
```

### 3. 测试

```
echo $JAVA_HOME      # 检验变量值
```

正常会输出 2. 环境变量JDK配置路径。

```
java -version
```

正确配置会输出java版本号。

## 5 安装python

CentOS自带python2版本过低，我们进行python3安装。

### 1. yum查找python3

查找仓库存在的python3安装包

```
yum list python3
```

```
python3.x86_64                               3.6.8-10.el7                                @base
```

### 2. yum 安装python3

```
sudo yum install python3.x86_64
```

如果最开始会显示没有，等一会自动切换阿里源就可以进行安装了，同时还会安装相关依赖。

## 4.1.2 hadoop 安装

本文使用 `wget` 命令来下载 `hadoop`：[了解更多wget](#)

使用的是[北理工镜像站](#)，下载 `hadoop`：

# Index of /apache/hadoop/common/hadoop-2.8.5

Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>	-	-	-
 <a href="#">hadoop-2.8.5-src.tar.gz</a>	2018-09-18 23:47	34M	
 <a href="#">hadoop-2.8.5.tar.gz</a>	2018-09-18 23:47	235M	

## 1. 下载

```
sudo wget -O hadoop-2.8.5.tar.gz https://mirrors.cnnic.cn/apache/hadoop/common/hadoop-2.8.5/hadoop-2.8.5.tar.gz
```

- `wget -O <指定下载文件名> <下载地址>`

## 2. 解压

```
sudo tar -zxf hadoop-2.8.5.tar.gz -C /usr/local
```

把下载好的文件 `hadoop-2.8.5.tar.gz` 解压到 `/usr/local` 目录下

## 3. 修改文件

```
cd /usr/local/ # 切换到解压目录下
sudo mv ./hadoop-2.8.5/ ./hadoop # 将解压的文件hadoop-2.8.5重命名为hadoop
sudo chown -R hadoop:hadoop ./hadoop # 修改文件权限
```

## 4. 测试

```
cd /usr/local/hadoop # 切换到hadoop目录下
./bin/hadoop version # 输出hadoop版本号
```

```
[hadoop@huang hadoop]$ ./bin/hadoop version
Hadoop 2.8.5
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 0b8464d75227fcee2c6e7f2410377b3d53d3d5f8
```

## 4.1.3 spark安装

在前我们已经安装了 *hadoop*，现在我们来开始进行 *spark* 安装。

这次下载根据官网推荐使用的清华源。

### 1. 下载

官网下载地址: [官网下载](#)

## Download Apache Spark™

1. Choose a Spark release: `2.4.4 (Aug 30 2019)`
2. Choose a package type: `Pre-built with user-provided Apache Hadoop`
3. Download Spark: `spark-2.4.4-bin-without-hadoop.tgz`
4. Verify this release using the 2.4.4 [signatures](#), [checksums](#) and [project release KEYS](#).

- 这样选择的版本可以使用于大部分 `hadoop` 版本

点击上述链接，根据跳转的页面提示选择清华源下载：

```
sudo wget -O spark-2.4.4-bin-without-hadoop.tgz
http://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-2.4.4/spark-2.4.4-bin-without-hadoop.tgz
```

### 2. 解压

同前解压到 `/usr/local` 目录下

```
sudo tar -zxvf spark-2.4.4-bin-without-hadoop.tgz -C /usr/local
```

### 3. 设置权限

```
cd /usr/local # 切换到解压目录
sudo mv ./spark-2.4.4-bin-without-hadoop ./spark # 重命名解压文件
sudo chown -R hadoop:hadoop ./spark # 设置用户hadoop为目录spark拥有者
```

### 4. 配置spark环境

先切换到 `/usr/local/spark` , (为了防止没权限, 下面用 `sudo` )

```
cd /usr/local/spark
cp ./conf/spark-env.sh.template ./conf/spark-env.sh
```

编辑 `spark-env.sh` 文件 :

```
vim ./conf/spark-env.sh
```

在第一行添加下面配置信息, 使得Spark可以从Hadoop读取数据。

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

### 5. 配置环境变量

```
vim ~/.bashrc
```

在 `.bashrc` 文件中添加如下内容:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk # 之前配置的java环境变量
export HADOOP_HOME=/usr/local/hadoop # hadoop安装位置
export SPARK_HOME=/usr/local/spark
export PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/lib/py4j-0.10.7-
src.zip:$PYTHONPATH
export PYSARK_PYTHON=python3 # 设置pyspark运行的python版本
export PATH=$HADOOP_HOME/bin:$SPARK_HOME/bin:$PATH
```

最后为了使得环境变量生效, 执行:

```
source ~/.bashrc
```

### 6. 测试是否运行成功

```
cd /usr/local/spark
bin/run-example sparkPi
```

执行会输出很多信息，也可以选择执行：

```
bin/run-example SparkPi 2>&1 | grep "Pi is"
```


```
[hadoop@huang spark]$ bin/run-example SparkPi 2>&1 | grep "Pi is"
Pi is roughly 3.135315676578383
```

#### 4.1.4 测试

## 1. 启动pyspark

```
cd /usr/local/spark
bin/pyspark
```

```
[hadoop@huang spark]$ bin/pyspark
Python 3.6.8 (default, Aug  7 2019, 17:28:10)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
Type "help", "copyright", "credits" or "license" for more information.
20/01/24 11:28:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
```



version 2.4.4

## 2. 简单测试

```
>>> 8 * 2 + 5
```

```
>>> 2019+1*1
2020
```

使用 `exit()` 命令可退出。

## 4.2 Hadoop+Spark 分布式环境搭建

### 4.2.1 Hadoop集群配置

## Hadoop文件配置

我们需要修改hadoop配置文件。

## 1. 切换目录

配置文件在 `/usr/local/hadoop/etc/hadoop` 目录下:

```
cd /usr/local/hadoop/etc/hadoop
```

```
[hadoop@master hadoop]$ ll
total 160
-rw-r--r-- 1 hadoop hadoop 4942 Sep 10 2018 capacity-scheduler.xml
-rw-r--r-- 1 hadoop hadoop 1335 Sep 10 2018 configuration.xml
-rw-r--r-- 1 hadoop hadoop 318 Sep 10 2018 container-executor.cfg
-rw-r--r-- 1 hadoop hadoop 1085 Jan 23 17:51 core-site.xml
-rw-r--r-- 1 hadoop hadoop 3804 Sep 10 2018 hadoop-env.cmd
-rw-r--r-- 1 hadoop hadoop 4666 Sep 10 2018 hadoop-env.sh
```



## 2. 修改文件 `core-site.xml`

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

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/hadoop/tmp</value>
    <description>Abase for other temporary directories.</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://0.0.0.0:9000</value>
  </property>
</configuration>
```

⚠ 实际测试必须要 `hdfs://0.0.0.0:9000` 才能使用 `hdfs` 服务。

⚠ 有可能依旧报错: `Error JAVA_HOME is not set and could not be found -`

- 配置 `hadoop-env.sh`

```
cd /usr/local/hadoop/etc/hadoop
vim hadoop-env.sh
```

配置 `JAVA_HOME` 路径如下:

```
The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
```

## 3. 修改 `hdfs-site.xml` :

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

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/tmp/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/tmp/dfs/data</value>
  </property>
</configuration>
```

## 集群启动测试

### 1. 启动集群

```
cd /usr/local/hadoop
bin/hdfs namenode -format    # 注意, 仅在第一次启动集群时使用该命令格式化!
sbin/start-all.sh
```

## 2. 测试

```
jps
```

出现以下6个进程则配置成功:

```
[hadoop@huang hadoop]$ jps
3024 DataNode
3184 SecondaryNameNode
3729 Jps
3331 ResourceManager
3431 NodeManager
2895 NameNode
```

## 4.2.2 Spark集群配置

### Spark配置

#### 1. 切换配置目录

```
cd /usr/local/spark/conf
```

#### 2. 配置 spark-env.sh 文件

```
cp spark-env.sh.template spark-env.sh
```

开始编辑, 添加下面内容:

```
vim spark-env.sh
```

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export SPARK_MASTER_IP=localhost
```

### 启动Spark集群

执行以下操作

#### 1. 先启动hadoop集群

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

#### 2. 启动spark集群

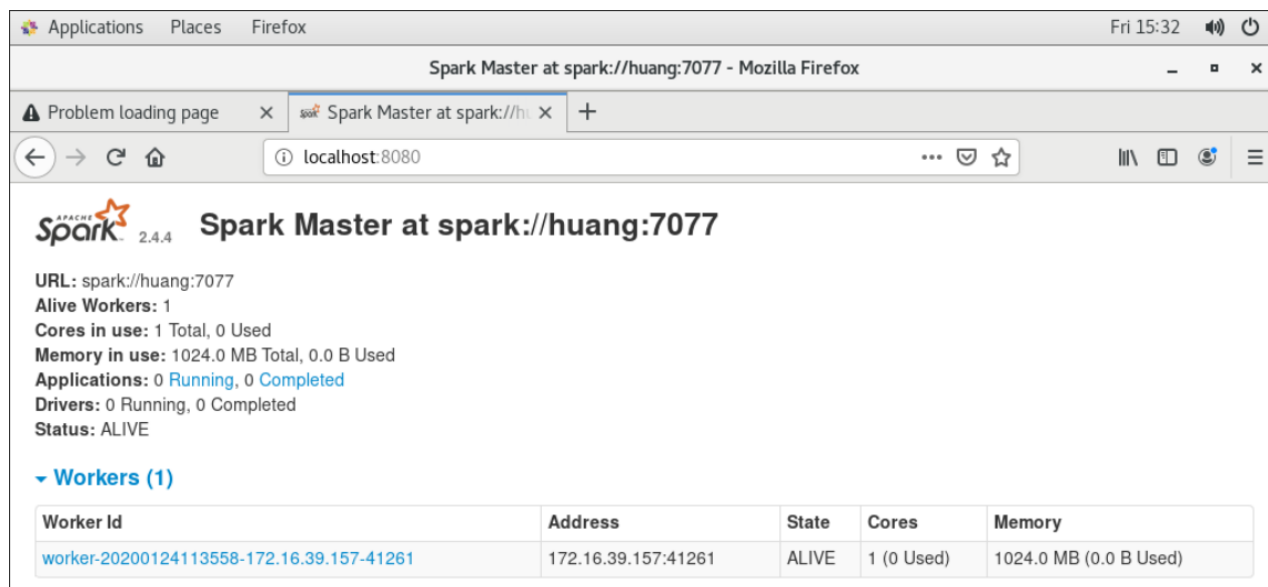
```
cd /usr/local/spark/  
sbin/start-all.sh
```

运行 `jps` 命令可以看到:

```
[root@huang ~]# jps  
3024 DataNode  
3184 SecondaryNameNode  
3331 ResourceManager  
3909 Worker  
23798 Jps  
3431 NodeManager  
3770 Master
```

### 3. web UI查看

在浏览器上输入: `localhost:8080` , 如果出现下面界面则表示 *Hadoop+Spark* 分布式环境搭建成功!



Applications Places Firefox Fri 15:32

Spark Master at spark://huang:7077 - Mozilla Firefox

Problem loading page x Spark Master at spark://hu x +

localhost:8080

**Spark Master at spark://huang:7077**

URL: spark://huang:7077  
Alive Workers: 1  
Cores in use: 1 Total, 0 Used  
Memory in use: 1024.0 MB Total, 0.0 B Used  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20200124113558-172.16.39.157-41261	172.16.39.157:41261	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)

🎉🎉 聪明如你终于做到这步了, 第一个实验完结, 撒花 🎉🎉