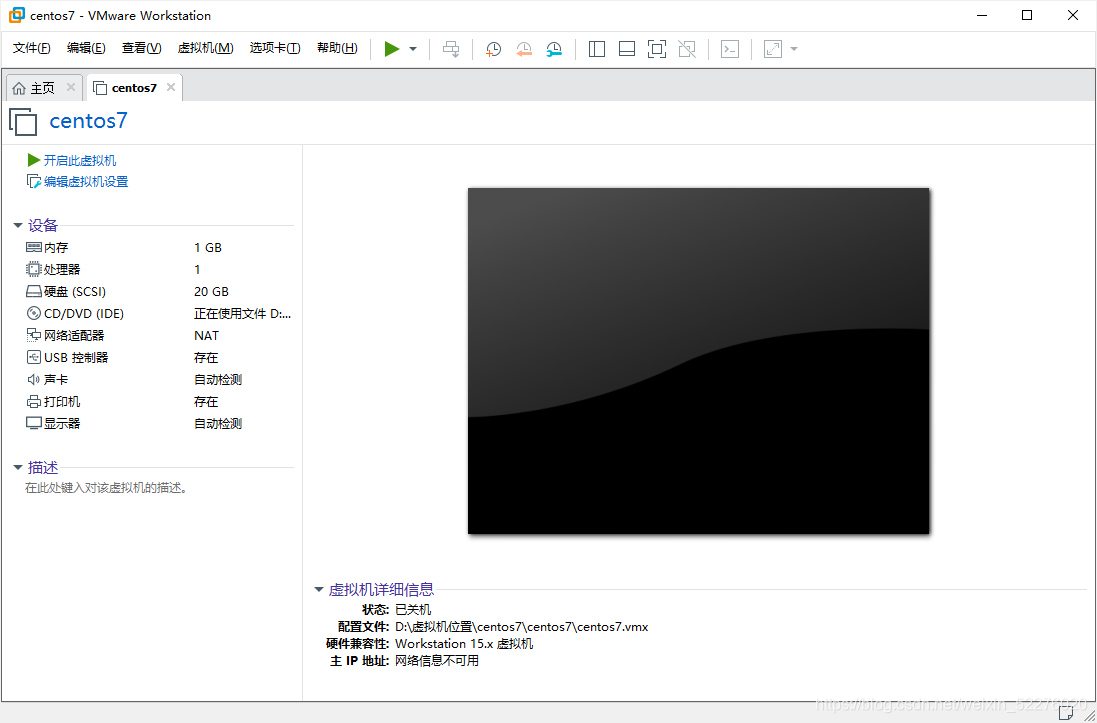
**本文档推荐用word打开（wps也可以，就是各级目录不好展开合上）**

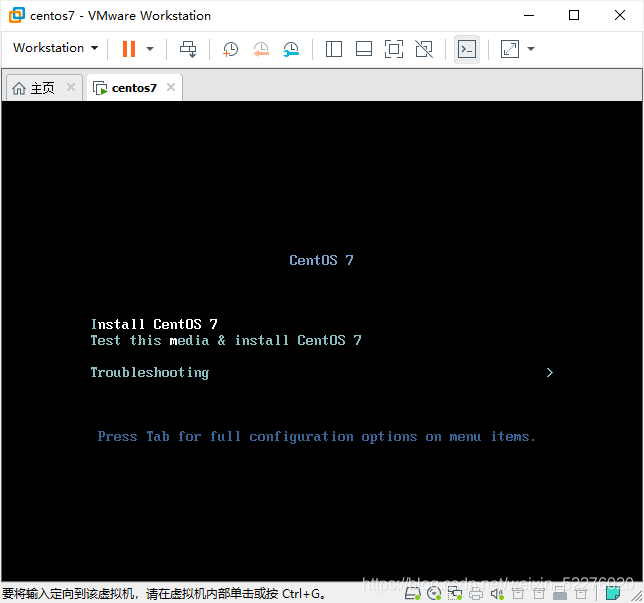
**参考pdf：《大数据技术基础系统部署指南v3》**

**一、最小化安装Centos7.9，vmware16，设置完硬件配置信息后。稍后安装系统**

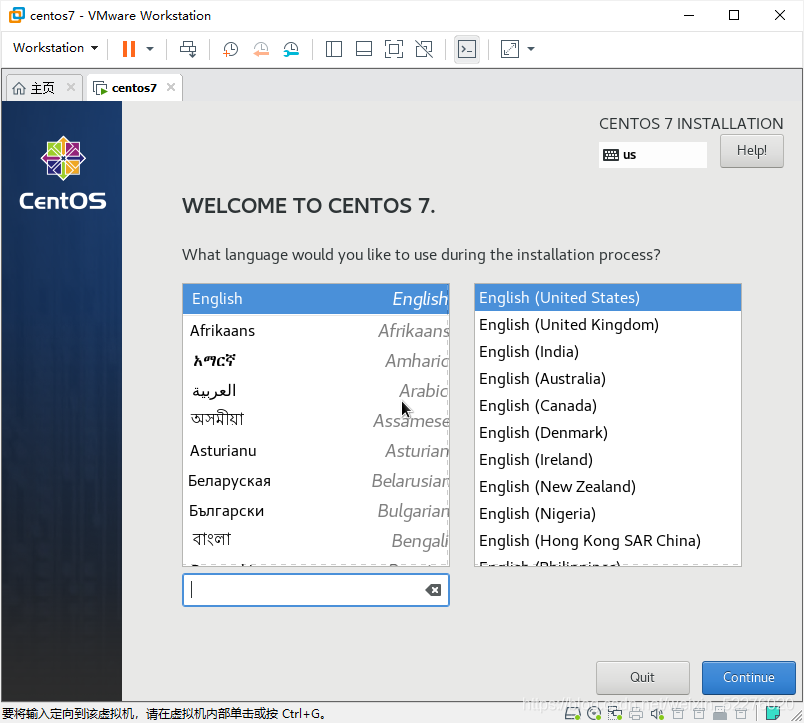
**1.点击打开虚拟机，将安装文件放入虚拟光驱**



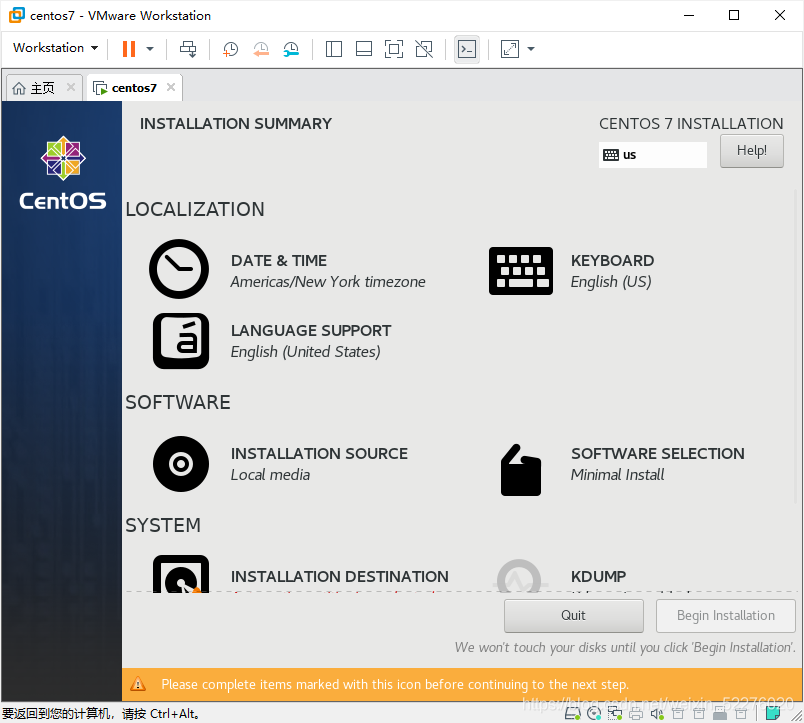
**2.鼠标点击虚拟机界面后，按方向上键，选择Install CentOS7，回车。鼠标点进虚拟机后无法控制鼠标，需要按CTRL+ALT释放鼠标**



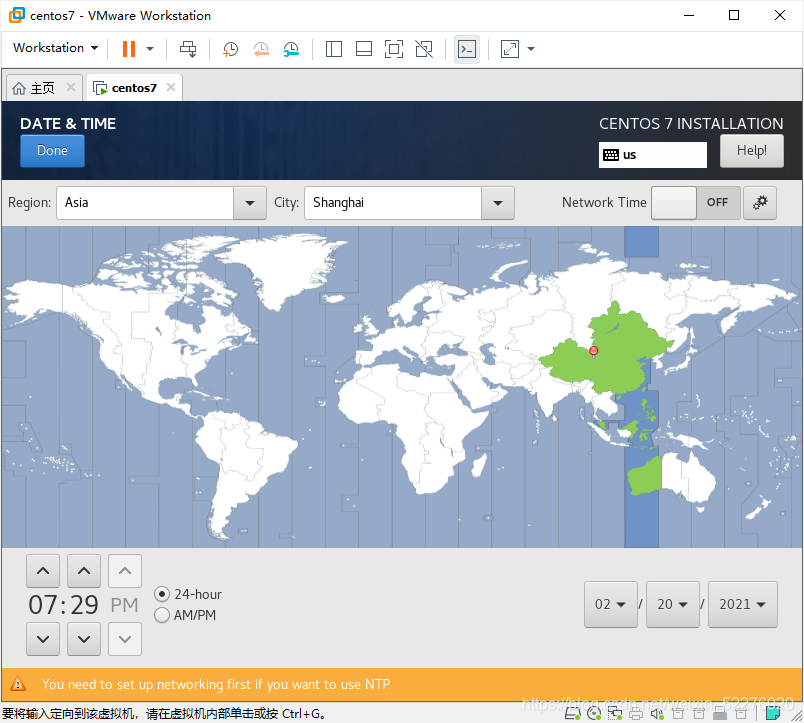
**3.默认English，最好选择中文，可以加装中文字体。，Continue**

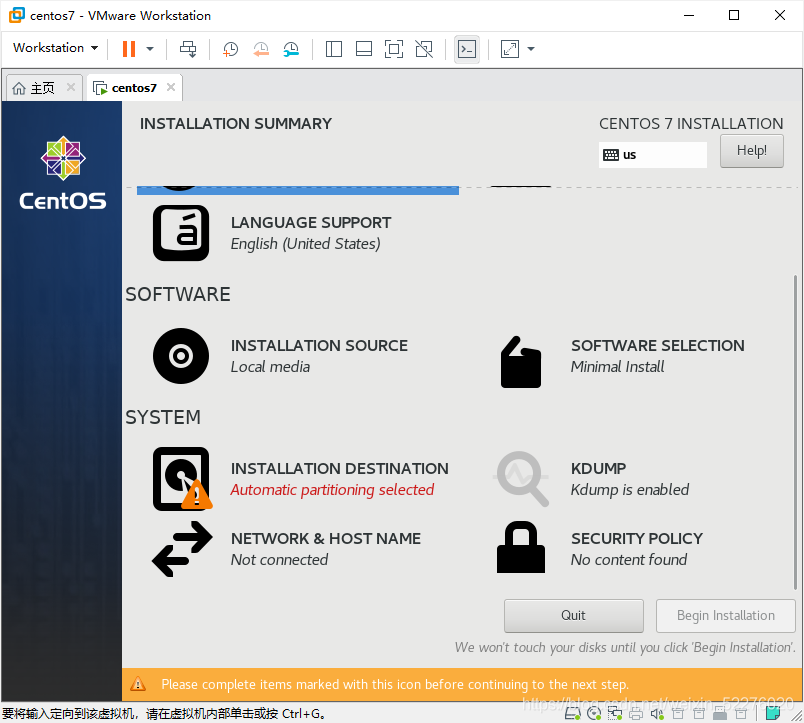


**4.进入Installion summary，点击 DATE & TIME配置时间**

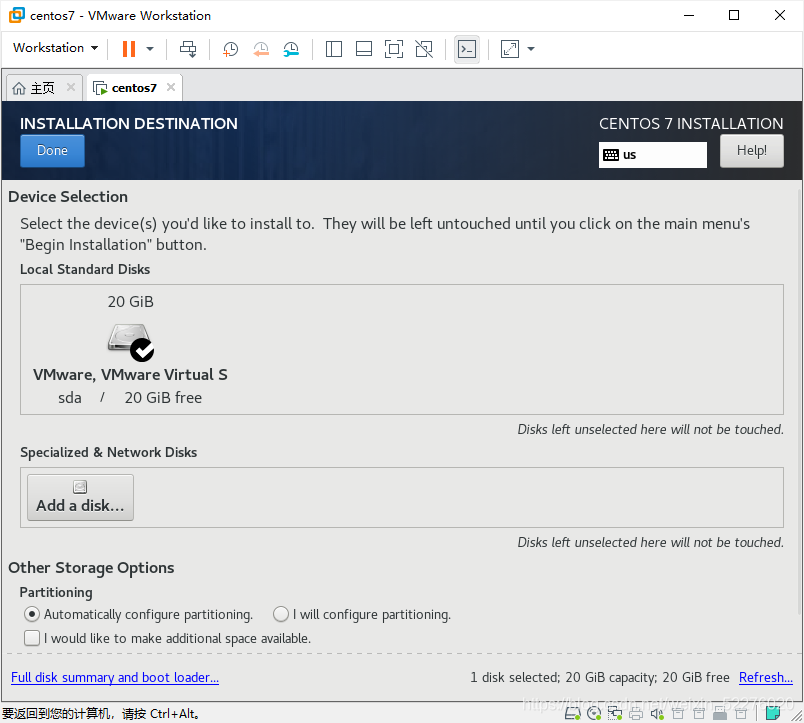


**5.选择城市为上海（只有上海），Done**



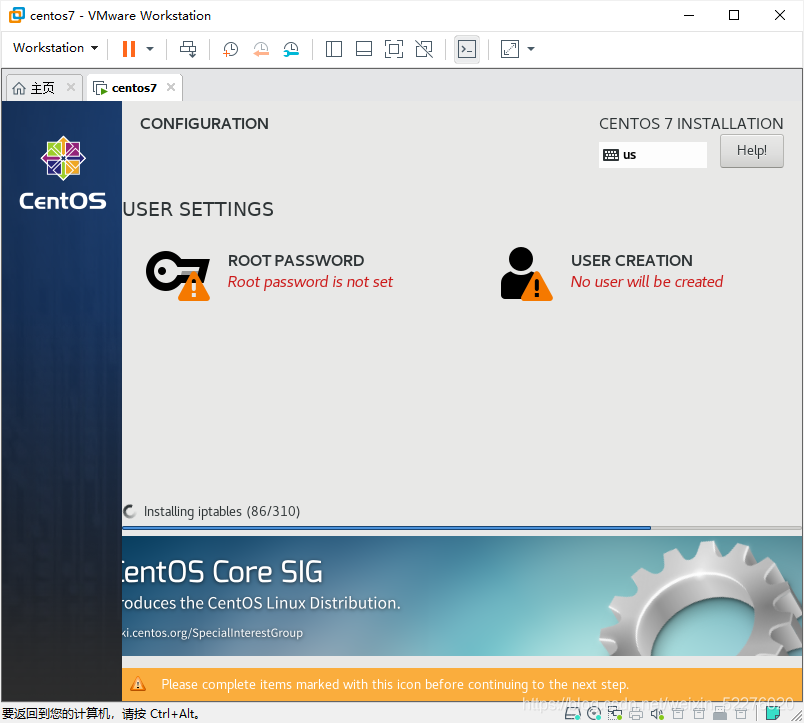
6.点击INSTALLATION DESTINATION，配置硬盘

7.直接点击Done，默认自动分区，如果自定义参考基础linux知识部分



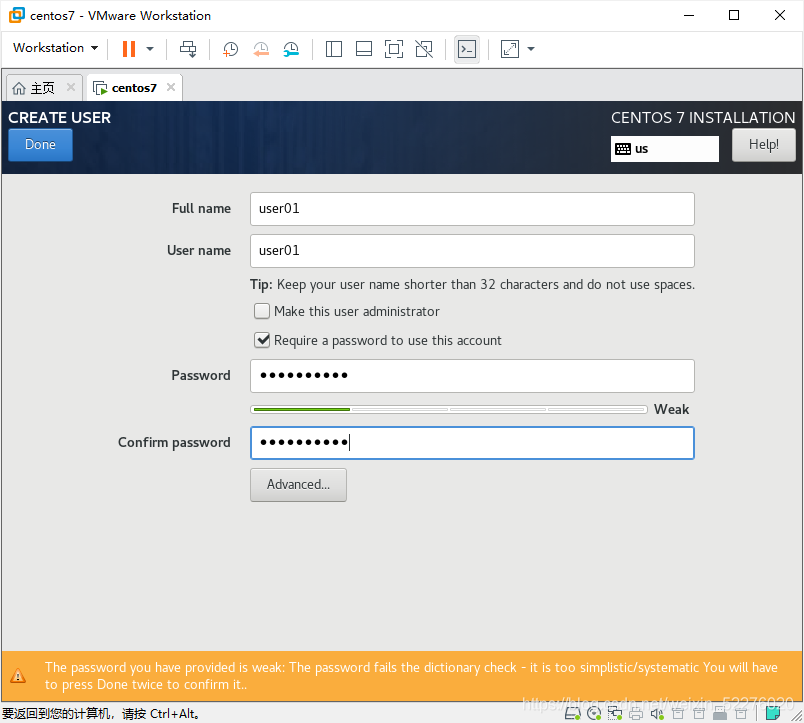
8.点击Begin Install，开始安装，默认Minimal Install（最小化安装）

9.点击ROOT PASSWORD配置root账户密码



10.配置完后点击Done，注意密码强度，如果是弱密码需要连续点击两次Done

11.创建用户，不是必须，默认小键盘锁定，解锁后才能使用，弱密码同样点击两次Done



12.点击Finish configuration完成配置

13.安装完成后，点击Reboot进行重启

**常用混合密码**

1. **配置静态IP**

**查看网络地址，因为是NAT模式联网，需要查看NAT网段。点击虚拟机编辑菜单，虚拟网络编辑器选项，发现VMnet8是NAT模式，它的子网地址是192.168.136.0，点击NAT设置，网关是192.168.136.2，子网掩码是255.255.255.0。每台虚拟机的可能都不一样。**



1. 虚拟机安装centos系统，设置NAT模式和win组成局域网。点击NAT设置，网关是192.168.136.2，子网掩码是255.255.255.0
2. #vim /etc/sysconfig/network-scripts/ifcfg-eth0

vi /etc/sysconfig/network-scripts/ifcfg-ens33 #centos7

作以下改动：

#删除UUID和HWADDR#centos7不需要删除

ONBOOT=yes

BOOTPROTO=static

IPADDR=192.168.136.100 # 必须与你的vmnet8虚拟网卡的IP在同一个网段

NETMASK=255.255.255.0

GATEWAY=192.168.136.2 # 根据虚拟机的信息必须与你的虚拟网卡的IP在同一个网段

DNS1= 114.114.114.114 #192.168.136.2?不知道可否

service network restart或者

systemctl restart network #centos7

测试是否可以和 Windows 主机广播地址通信

ping 192.168.136.0 –b

测试 Window 主机是否可以和虚拟机通信

Ping 192.168.136.100

测试 Linux 虚拟机是否可以连通外网：

Ping [www.baidu.com](http://www.baidu.com)

1. (命令hostnamectl set-hostname node1)

vim /etc/hostname

修改主机名为repo（源）

最好再把/etc/sysconfig/network里面按照centos6修改一下，避免后续其他操作报错

vim /etc/sysconfig/network

NETWORKING=yes

HOSTNAME=repo

systemctl restart systemd-hostnamed

1. vim /etc/hosts 此文件第一列是ip 第二列是主机名 第三列标识别名（可选

★将计划中的集群的各台主机添加到主机的 hosts 列表中的末尾

|  |
| --- |
| vim /etc/hosts |
| 192.168.136.100 repo  192.168.136.101 node-1  192.168.136.102 node-2  192.168.136.103 node-3  192.168.136.104 node-4 |

1. 关闭服务器防火墙

|  |
| --- |
| systemctl status firewalld.service |
| systemctl disable firewalld.service |
| systemctl stop firewalld.service |
| firewall-cmd --state |

第1段代码块：查看当前系统中的防火墙服务的状态，查看是否关闭

若systemctl status firewalld.service显示为inactive状态，后三段代码块可以忽略不执行

第2段代码块：取消开机启动防火墙，确保重启后防火墙为关闭状态

第3段代码块：关闭当前系统中的防火墙服务

第4段代码块：查看当前系统中的防火墙服务的状态，查看是否关闭

#centos6

|  |
| --- |
| service iptables stop |
| chkconfig iptables off |
| service iptables status |

5. 设置 selinux 将 SELINUX 改为 disabled

|  |
| --- |
| vim /etc/selinux/config |

SELINUX=enforcing改成SELINUX=disabled

查看状态

sestatus

## 安装OpenJDK1.8

1.安装 jdk1.8

卸载旧版本的JDK：

|  |
| --- |
| rpm -qa | grep jdk  yum -y remove java-1.4.2-gcj-compat-1.4.2.0-40jpp.115  或者#rpm -e –nodeps java-1.4.2-gcj-compat-1.4.2.0-40jpp.115 |

建立安装文件夹

|  |
| --- |
| mkdir /usr/java  cd /usr/java |

将jdk1.8文件用winscp从windows传入linux目录

|  |
| --- |
| tar -zxvf jdk-8u281-linux-x64.tar.gz  vim /etc/profile |

再文件末尾加入注意hadoop的安装目录正确

|  |
| --- |
| #set java environment  export JAVA\_HOME=/usr/java/jdk1.8.0\_281  export JRE\_HOME=/usr/java/jdk1.8.0\_281/jre  export CLASS\_PATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib  export HADOOP\_HOME=/usr/local/hadoop-3.2.1  export HADOOP\_YARN\_HOME=$HADOOP\_HOME  export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME  export PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin:$HADOOP\_HOME/bin |

让修改生效:

|  |
| --- |
| source /etc/profile |

## 1.2建立Hadoop用户

|  |
| --- |
| # adduser hadoop;pwd hadoop |

此处我直接用root来管理

1. groupadd 命令建立新用户组

|  |
| --- |
| groupadd hadoop |
| groupadd hdfs |
| groupadd hbase |
| groupadd hive |
| groupadd spark |

2. 将组名（如hadoop）添加到用户（如root）所属组中

|  |
| --- |
| usermod -a -G hadoop root |
| usermod -a -G hdfs root |
| usermod -a -G hbase root |
| usermod -a -G hive root |
| usermod -a -G spark root |
| cat /etc/group |

其中cat /etc/group是查询是否添加成功（也可不执行）

## 1.3设置时间自动同步

1.修改时区（时区没问题就不用）

rm -rf /etc/localtime

ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

vim /etc/sysconfig/clock

ZONE="Asia/Shanghai"

UTC=false

ARC=false

2.安装并设置开机自启

yum install -y ntp

systemctl start ntpd #如果不做集群中的时间服务器不用启动服务

systemctl enable ntpd#如果不做集群中的时间服务器不用自启动服务

3.配置开机启动校验

vim /etc/rc.d/rc.local

/usr/sbin/ntpdate ntp1.aliyun.com > /dev/null 2>&1; /sbin/hwclock -w

配置定时任务

4.crontab -e

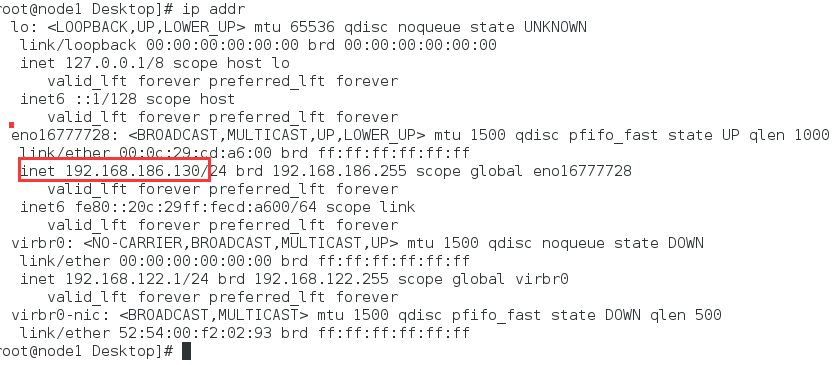
0 \*/1 \* \* \* ntpdate ntp1.aliyun.com > /dev/null 2>&1; /sbin/hwclock -w

## 设置集群环境

**关机并且保存快照。并将repo的快照*完全克隆*成三个虚拟机。node-1…3，并打开虚拟机**

克隆完成之后，右键点击新复制的虚拟机，点击settings，，点击Ｎetwork Adapter，然后点击右边的advanced：点击generate，使新的虚拟机生成新的MAC地址，复制新生成的MAC信息，后面会用

打开新复制的虚拟机，输入命令ip addr，此条命令是查看虚拟机的IP地址以及MAC地址，如下：红色框住的就是机器的IP地址

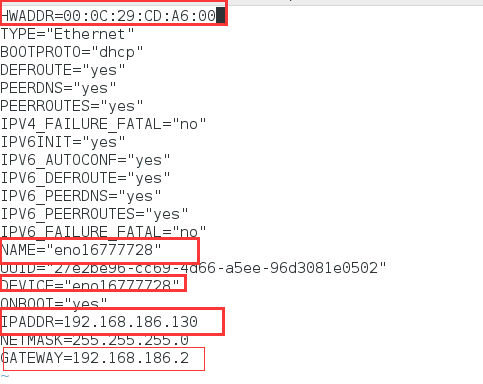


(1) 修改IP和mac地址和UUID

然后输入如下命令：vim /etc/sysconfig/network-scripts/ifcfg-ens33，注意这里的ifcfg-eno16777728是和上图中红点点那里的名称是一样的。然后对这个文件进行编辑，输入如下信息：（注意红色框住的地方改为自己机器相应的信息）

uuidgen 生成新的uuid

HWADDR=



node-1的IPADDR修改为：192.168.9.101

node-2的IPADDR修改为：192.168.9.102

node-3的IPADDR修改为：192.168.9.103

service network restart

(2) 配置主机名

vim /etc/hostname

修改主机名为node-1

最好再把/etc/sysconfig/network里面按照centos6修改一下，避免后续其他操作报错

vim /etc/sysconfig/network

NETWORKING=yes

HOSTNAME=node-1

systemctl restart systemd-hostnamed (3)测试

使用每台虚拟机 ping 其他虚拟机，ping 外网，看是否成功，如果有错误，检查上面提到的三个配置文件，比如 node-1 应该做以下测试：

ping node-2;

ping node-3;

ping www.baidu.com

(2) 使用windows ping 3台虚拟机，检查windows 是否可以和虚拟机通信

测试成功后，关机，为每个虚拟机保存快照，一定要养成保存快照这个习惯

## 1.打开虚拟机

2.配置SSH无密钥登陆。

★各node生成公钥和私钥，分别在各node上执行以下命令：

检查ssh和rsync已经安装了。可以通过下面命令查看结果显示如下：

rpm -qa | grep openssh

rpm -qa | grep rsync

假设没有安装ssh和rsync，可以通过下面命令进行安装

yum install ssh 安装SSH协议

yum -y install rsync （rsync是一个远程数据同步工具，可通过LAN/WAN快速同步多台主机间的文件）

service sshd restart 启动服务

--------------------------------------centos6内容再次不用设置。用下面centos7方法。

#修改配置文件

vim /etc/ssh/sshd\_config

’‘’

PasswordAuthentication no #取消密码认证？似乎设置600后不用取消依然可以无密码。RSAAuthentication yes # 启用 RSA 认证

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

AuthorizedKeysFile .ssh/authorized\_keys # 公钥文件路径（和上面生成的文件同）

‘’‘

vim /etc/ssh/ssh\_config

‘’’

GSSAPIAuthentication no #取消GSS认证

‘’’

service sshd restart 启动服务

执行

|  |
| --- |
| ssh-keygen -t rsa |

7.在node-1主机上，进入.ssh目录下把公钥追加到 authorized\_keys文件中：

cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

chmod 600 ~/.ssh/authorized\_keys

在node-1主机上把authorized\_keys文件复制到下一个node-2主机上

scp ~/.ssh/authorized\_keys [root@node-x:~/.ssh/](mailto:root@node_x:~/.ssh/)

将node-2上的公钥追加到authorized\_keys文件中：

cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

把node2上的authorized\_keys文件复制到node3主机上：

…………

将node-4上的公钥追加到authorized\_keys文件中。

最后用再node1主机上再将node4上的authorized\_keys文件复制回来。在复制到所有主机上，重启各自ssh服务即可完成配置。

scp root@node-4:~/.ssh/authorized\_keys ~/.ssh/

--------------------------------------------------------------cenos7方法更简单

6. ★配置SSH无密钥登陆

|  |
| --- |
| ssh-keygen -t rsa |
| ssh-copy-id node-1 |
| ssh-copy-id node-2#本地的ssh公钥文件安装到远程主机对应的账户下,自动生成目录、文件、权限。并在自己ssh目录下生成known\_hosts，并把远程访问主机ip登记再knownhost下。 |
| ssh-copy-id node-3 |

每执行完一次ssh-copy-id node-1，输入yes，接着再输入密码，然后再执行ssh-copy-id node-2，同理再执行ssh-copy-id node-3

7. ★检查无密钥登录配置成功

|  |
| --- |
| ssh node-1 |
| exit |
| ssh node-2 |
| exit |
| ssh node-3 |
| exit |

为三台虚拟机保存快照。

## 1.4安装Hadoop

**从这里开始只操作node-1一台,无特殊说明node-2、node-3暂不操作，等node-1配置完后后期需要时候会通过scp同步到另外两台服务器上**

1. 安装wget指令

|  |
| --- |
| yum install wget -y |

wget 是一个从网络上自动下载文件的自由工具

2. 下载 hadoop 软件包

|  |
| --- |
| wget https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz |

3. 将 hadoop 安装包放入/usr/local 中并解压

|  |
| --- |
| chown -R root /usr/local |
| tar zxvf hadoop-3.2.1.tar.gz -C /usr/local |
| cd /usr/local |
| ll |

4.检查安装成功

|  |
| --- |
| cd /usr/local/hadoop-3.2.1 |
| bin/hadoop version |

这里直接cd或者cd ~可以返回到底，cd ..可以返回上一个文件目录，ls是仅列出当前目录下的所有文件名，ll则是详细列出，运行不了就cd /usr/local/hadoop-3.2.1到这个目录下，就能运行了，后面很多都是要cd xxx(具体目录)下操作的

## 1.5运行hadoop

**无特殊说明只操作node-1，而node-2、node-3暂不操作**

Hadoop安装分为3种：

①单机模式:单机模式: Hadoop默认模式为非分布式模式(本地模式) ，无需进行其他配置即可运行。非分布式即单Java进程，方便进行调试。

②伪分布式模式: Hadoop可以在单节点上以伪分布式的方式运行，Hadoop 进程以分离的Java进程来运行，节点既作为NameNode也作为DataNode,同时，读取的是HDFS中的文件。

③完全分布式模式:使用多个节点构成集群环境来运行Hadoop。

**实际使用的话是完全分布式，因此不是出于初学目的的话下面可以略过1.5.1和1.5.2有关本地模式和伪分布式的操作，省时间直接进行1.5.3完全分布式最终完成hadoop的3台集群工作任务**

### 本地运行hadoop（单机模式）

1. 运行 hadoop 例子的 jar 包，查看有哪些例程。

|  |
| --- |
| cd /usr/local/hadoop-3.2.1 |
| bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar |

2. 在本地模式下运行grep例程

|  |
| --- |
| mkdir input |
| bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar grep input output 'per[a-z.]+' |

上述mkdir input的意为新建一个名叫input的目录

3. 查看输出 output 目录中的结果

|  |
| --- |
| cd output |
| ls |
| cat part-r-00000 |

### 伪分布式hadoop

进行伪分布式hadoop前在这里可以考虑给node-1、node-2、node-3系统**三个都各自备份**；

操作完伪分布式后可以给node-1**再次备份**然后node-1**回档到第一次备份**（可以两次都不备份不回档直接做），node-2、node-3不用操作

理由：后面的1.5.3完全分布式hadoop修改的文件跟伪分布式有重叠，只是内容不一样，当然直接再改也行；另外养成系统备份的好习惯，出了问题好回档，不用全部重做一遍

#### 1.5.2.1伪分布式运行HDFS

1. 配置core-site.xml

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/etc/hadoop/core-site.xml |
| <property>  <name>fs.defaultFS</name>  <value>hdfs://localhost:9000</value>  <final>true</final>  </property> |

第二段代码块在<configuration>和</configuration>之间添加！！（下面带<property>和</property>的均如此）

上面localhost就可以了，不用改成node-1

2. 配置hdfs-site.xml

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/etc/hadoop/hdfs-site.xml |
| <property>  <name>dfs.replication</name>  <value>1</value>  </property> |

输入了vim后如果跳出[O]pen Read-Only, (E)dit anyway, (R)ecover, (Q)uit, (A)bort的界面，按键盘E编辑就行了

3.初始化名称节点

|  |
| --- |
| cd /usr/local/hadoop-3.2.1 |
| bin/hdfs namenode -format |

4.★ 给当前用户配置为 root,随后开启hdfs服务守护进程

|  |
| --- |
| export HDFS\_NAMENODE\_USER=root |
| export HDFS\_DATANODE\_USER=root |
| export HDFS\_SECONDARYNAMENODE\_USER=root |
| vim /usr/local/hadoop-3.2.1/etc/hadoop/hadoop-env.sh |
| export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-0.el7\_7.x86\_64 |
| sbin/start-dfs.sh |

5. jps查看JVM 中的hadoop进程

|  |
| --- |
| jps |

6. 伪分布式模式运行wordcount例程：先上传文本文件到 hdfs 上的相应的文件目录下

|  |
| --- |
| bin/hdfs dfs -mkdir /user |
| bin/hdfs dfs -mkdir /user/root |
| bin/hdfs dfs -mkdir input |
| bin/hdfs dfs -put etc/hadoop/\*.xml input |

下面一直ls的步骤可跳过

|  |
| --- |
| ls |
| bin/hdfs dfs -ls |
| bin/hdfs dfs -ls /user |
| bin/hdfs dfs -ls /user/root |
| bin/hdfs dfs -ls /user/root/input |

7. 运行grep例程

|  |
| --- |
| bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar grep input output 'dfs[a-z.]+' |

查找当前用户在 hdfs 文件系统中默认目录下的 input 目录中的文本文件中包含有 dfs 的单词，并输出至默认目录/user/root下的 output 目录中

8.运行成功后查看 output 目录中的结果

|  |
| --- |
| bin/hdfs dfs -ls /user/root/output |
| bin/hdfs dfs -cat output/\* |

下面也可以将结果传回到本地文件系统（可跳过）

|  |
| --- |
| bin/hdfs dfs -get output output |
| ls |
| ls output/\* |
| cat output/\* |

9.关闭伪分布式 HDFS

|  |
| --- |
| sbin/stop-dfs.sh |
| jps |

#### 1.5.2.2伪分布式运行YARN

1. 配置mapred-site.xml

|  |
| --- |
| vim etc/hadoop/mapred-site.xml |
| <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>   </property>  <property>  <name>mapreduce.application.classpath</name>  <value>$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*:$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*</value>   </property> |

2.配置yarn-site.xml

|  |
| --- |
| vim etc/hadoop/yarn-site.xml |
| <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  <property>  <name>yarn.nodemanager.env-whitelist</name>  <value>JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLASSPATH\_PREPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_MAPRED\_HOME</value>  </property> |

3.启动 YARN 服务守护进程

|  |
| --- |
| export YARN\_RESOURCEMANAGER\_USER=root |
| export YARN\_NODEMANAGER\_USER=root |
| sbin/start-yarn.sh |

4.关闭伪分布式YARN

|  |
| --- |
| jps |
| sbin/stop-yarn.sh |
| jps |

### 完全分布式hadoop

（做过1.5.2伪分布式的无视这里）：如果跳过了1.52伪分布式，则在这里系统备份node-1、node-2、node-3（养成系统备份的好习惯，出了问题好回档，不用全部重做一遍）

**下面依旧是只操作node-1，无特殊提示node-2、node-3不操作**

1.★ 配置hadoop-env.sh运行环境,最后一行添加

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/etc/hadoop/hadoop-env.sh |
| export JAVA\_HOME=/usr/java/jdk1.8.0\_281  export HDFS\_NAMENODE\_USER=root  export HDFS\_DATANODE\_USER=root  export HDFS\_SECONDARYNAMENODE\_USER=root  启动 hadoop 集群时如果使用 root 用户提示错误，则在 hadoop-env.sh 中最后一行添加下面代码。则免去下面/sbin 中的文件配置。无错误则略过。  export JAVA\_HOME=/usr/java/jdk1.8.0\_281  export HDFS\_NAMENODE\_USER=root  export HDFS\_DATANODE\_USER=root  export HDFS\_SECONDARYNAMENODE\_USER=root  export YARN\_RESOURCEMANAGER\_USER=root  export YARN\_NODEMANAGER\_USER=root |

可将 hadoop 软件所在目录添加到系统环境（在/etc/profile 中添加）中，并把其中的 bin

目录添加到$PATH 中

export HADOOP\_HOME=[hadoop 所在路径]

export PATH=$PATH:${HADOOP\_HOME}/bin

2.★配置core-site.xml(下面<value>hdfs://node-1:9000</value>改成自己的)

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/etc/hadoop/core-site.xml |
| <property>    <!-- 必须设置：默认文件系统（存储层和运算层解耦 -->  <!-- 此处值为uri结构：使用内置的hdfs系统 端口号一般都是9000 -->  <name>fs.defaultFS</name>  <value>hdfs://node-1:9000</value>  <final>true</final>  </property>  <property>  <!-- 必须设置：hadoop在本地的工作目录，用于放hadoop进程的临时数据，可以自己指定 -->  <name>hadoop.tmp.dir</name>  <value>/root/hadoop/tmp</value>  </property> |

<property>和</property>均在<configuration></configuration>之间添加！！

3.★配置hdfs-site.xml（下面<value>node-1:9870</value>改成自己的）

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/etc/hadoop/hdfs-site.xml |
| <property>  <!-- hdfs保存namenode当前数据的路径，默认值需要配环境变量，建议使用自己创建的路径, 方便管理-->  <name>dfs.namenode.name.dir</name>  <value>/root/hadoop/dfs/name</value>  <final>true</final>  </property>  <property>  <!-- hdfs保存datanode当前数据的路径，默认值需要配环境变量，建议使用自己创建的路径, 方便管理-->  <name>dfs.datanode.data.dir</name>  <value>/root/hadoop/dfs/data</value>  <final>true</final>  </property>  <property>  <!-- hdfs存储数据的副本数量（避免一台宕机），可以不设置，默认值是3-->  <name>dfs.replication</name>  <value>3</value>  </property>  <property>   <!-- hdfs监听namenode的web的地址，默认就是9870端口，如果不改端口也可以不设置-->  <name>dfs.namenode.http-address</name>  <value>node-1:9870</value>  </property>  <property>  <!-- 配置为false后，可以允许不要检查权限就生成dfs上的文件，方便倒是方便了，但是你需要防止误删除 -->  <name>dfs.permissions</name>  <value>false</value>  </property>  <property>  <name>dfs.webhdfs.enabled</name>  <value>true</value>  </property> |

4.★配置mapred-site.xml（修改2处：<value>node-1:10020</value>和<value>node-1:19888</value>改成自己的）

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/etc/hadoop/mapred-site.xml |
| <property>  <!-- 必须设置，mapreduce程序使用的资源调度平台，默认值是local,若不改就只能单机运行，不会到集群上了 -->    <name>mapreduce.framework.name</name>  <value>yarn</value>  </property>  <property>  <!-- 这是3.2以上版本需要增加配置的，不配置运行mapreduce任务可能会有问题，记得使用自己的路径 -->  <name>mapreduce.application.classpath</name>  <value>$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*:$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*</value>  </property>  <property>  <name>mapreduce.jobhistory.address</name>  <value>node-1:10020</value>  </property>  <property>  <name>mapreduce.jobhistory.webapp.address</name>  <value>node-1:19888</value>  </property> |

5.★配置yarn-site.xml（<value>node-1</value>改成自己的）

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/etc/hadoop/yarn-site.xml |
| <property>  <!-- 必须配置指定YARN的老大(ResourceManager)在哪一台主机 -->  <name>yarn.resourcemanager.hostname</name>  <value>node-1</value>  </property>  <property>  <!-- 必须配置提供mapreduce程序获取数据的方式默认为空 -->  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  <property>  <name>yarn.nodemanager.env-whitelist</name>  <value>JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLASSPATH\_PREPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_MAPRED\_HOME</value>  </property>  <property>  <!--忽略虚拟内存的检查，如果你是安装在虚拟机上，这个配置很有用，配上去之后后续操作不容易出问题。 -->  <name>yarn.nodemanager.vmem-check-enabled</name>  <value>false</value>  </property> |

6.★修改 workers,**将localhost删除**，node-1、node-2、node-3添加进去

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/etc/hadoop/workers |
| node-1  node-2  node-3 |

cat查看添加是否成功（可跳过）

|  |
| --- |
| cat /usr/local/hadoop-3.2.1/etc/hadoop/workers |

7. 配置 hadoop 中/sbin 中的各项文件

①修改start-dfs.sh，文件顶部添加以下参数

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/sbin/start-dfs.sh |
| HDFS\_DATANODE\_USER=root  HDFS\_DATANODE\_SECURE\_USER=hdfs  HDFS\_NAMENODE\_USER=root  HDFS\_SECONDARYNAMENODE\_USER=root |

②修改stop-dfs.sh，文件顶部添加以下参数

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/sbin/stop-dfs.sh |
| HDFS\_DATANODE\_USER=root  HDFS\_DATANODE\_SECURE\_USER=hdfs  HDFS\_NAMENODE\_USER=root  HDFS\_SECONDARYNAMENODE\_USER=root |

③修改start-yarn.sh，顶部添加以下参数

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/sbin/start-yarn.sh |
| YARN\_RESOURCEMANAGER\_USER=root  HADOOP\_SECURE\_DN\_USER=yarn  YARN\_NODEMANAGER\_USER=root |

④修改stop-yarn.sh，顶部添加以下参数

|  |
| --- |
| vim /usr/local/hadoop-3.2.1/sbin/stop-yarn.sh |
| YARN\_RESOURCEMANAGER\_USER=root  HADOOP\_SECURE\_DN\_USER=yarn  YARN\_NODEMANAGER\_USER=root |

⑤★scp复制 hadoop-3.2.1 到node-2、node-3上（这里就解释了为什么一直以来只操作node-1，而node-2和node-3不用操作，因为一台配置好了同步到其它台就ok了）

|  |
| --- |
| for A in {2..3}; do scp -r /usr/local/hadoop-3.2.1 node-$A:/usr/local;done |

注意上面代码2、3和node-，根据自己情况具体修改，因为现在只在操作node-1，所以要scp复制同步到node-2、node-3上

8.启动集群

①初始化集群的名称节点（**format只执行一次，别重复执行！**）

|  |
| --- |
| /usr/local/hadoop-3.2.1/bin/hdfs namenode -format |

**（正常情况不执行）**如果不小心多次format出问题了，执行下面六步：

|  |
| --- |
| cd /usr/local/hadoop-3.2.1 |
| sbin/stop-all.sh |
| rm -rf /root/hadoop/dfs/name |
| rm -rf /root/hadoop/dfs/data |
| rm -rf hadoop.tmp.dir |
| /usr/local/hadoop-3.2.1/bin/hdfs namenode -format |
| sbin/start-all.sh |

②启动 hadoop 集群

|  |
| --- |
| cd /usr/local/hadoop-3.2.1 |
| sbin/start-all.sh |

如果warning说logs does not exist.无视掉就好，logs日志可以没有的

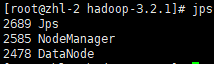
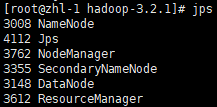
**一旦使用了sbin/start-all.sh后要切记**：任何时候**关闭虚拟机前先sbin/stop-all.sh**，否则下次可能报错，集群失败，又要重新格式化啥的很麻烦

9.检查集群情况

①在node-1、node-2、node-3中同时输入

|  |
| --- |
| jps |

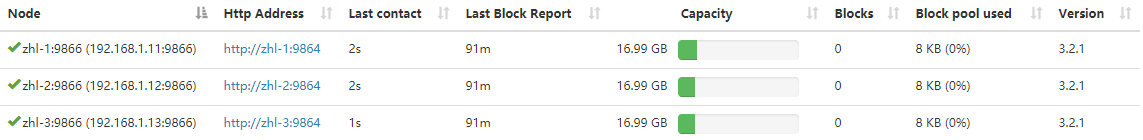
查看每台服务器上JVM 中 hadoop 进程运行情况，了解hadoop各组件在服务器集群中的分布情况



②★在**浏览器**中输入

|  |
| --- |
| <http://192.168.1.11:9870/> |

点击顶部“Datanodes”🡪下翻找到：

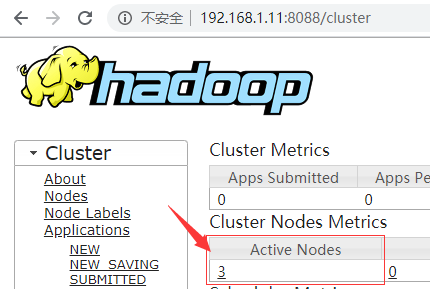


监控datanode 的状态（出现了3个绿√代表hadoop集群已经配置成功）

③★在**浏览器**中输入

|  |
| --- |
| [http://192.168.1.101:8088/](http://192.168.1.11:8088/) |

能看到集群管理页面

配置是成功的

④**只操作node-1**，关闭 hadoop 集群（**该步骤非常重要，start后关机前一定要先stop集群！！**）

|  |
| --- |
| cd /usr/local/hadoop-3.2.1 |
| sbin/stop-all.sh |

这里就3台hadoop完全式分布集群安装就已经成功了，下面的都是用这个集群去运行例程，可以跳过不做，这里总结下：上面在操作的时候，一直在node-1上操作，最后scp同步到node-2、node-3时这么多的配置文件里的node-1要改成node-2或node-3吗？答案是不用的，其他的里面都会是node-1，这才是正常的。原先一直用3台xshell6一起操作，导致后面一直被坑出问题，其实只用操作一台，注意避坑。

10.例程1**（省时间可不做跳过，建议做一下）**：完全分布式运行wordcount

①在此之前先调整时间，避免后面报错

时间同步调整博客：<https://blog.csdn.net/selectdb/article/details/81735104>

|  |
| --- |
| yum install ntp -y |
| systemctl enable ntpd.service |
| vim /etc/sysconfig/ntpd |

上面的vim添加-g -x参数

|  |
| --- |
| systemctl restart ntpd.service |
| ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime |
| date |

②上传文本文件到 HDFS 中

|  |
| --- |
| cd |
| **/usr/local/hadoop-3.2.1/sbin/start-all.sh** |
| hdfs dfs -mkdir /test |
| hdfs dfs -put /etc/profile /test/input |
| hdfs dfs -ls /test/input |

hdfs dfs -ls /test/input用于查看(非必要步骤可跳)

mkdir用于新建

/etc/profile是随意选的一个文件

/test和/test/input是自己可以随意起的文件名

**注意上面第三步又开启了集群，后面不用时要关掉**

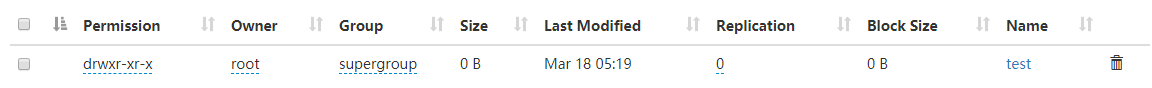
（**正常情况下不用执行**）如果前面有过多次format的行为，第四步会报错，要先离开安全模式，再执行/usr/local/hadoop-3.2.1/bin/hdfs dfs -put test1 /tmp

|  |
| --- |
| hadoop dfsadmin -safemode leave |

★接下来再次进入：

|  |
| --- |
| <http://192.168.1.11:9870/> |

然后点击右上角Utilities🡪Browse the file system

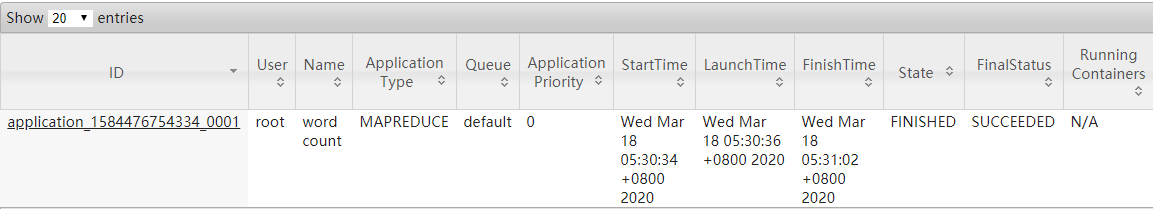


③运行 wordcount 例程

|  |
| --- |
| hadoop jar $HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar wordcount /test/input /test/output |

★接下来再次进入：

|  |
| --- |
| <http://192.168.1.11:8088/> |



看到多出了这个，wordcount例程到此完美结束

10.★例程2**（省时间可不做跳过，建议做一下）**：完全分布式运行蒙特卡洛方法计算圆周率

|  |
| --- |
| hadoop jar $HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar pi 10 1000 |



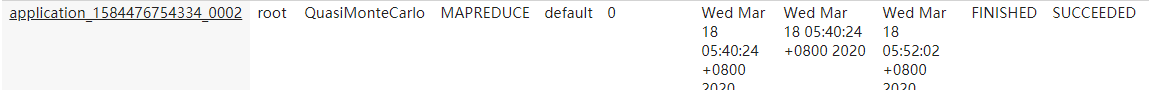
最后Π结果得出3.1408，想要更精确可以把数字调高，就是运行会变慢

运行时会发现有

这里的map和reduce有进度条原因是这样的：hadoop的[分布式计算](https://www.baidu.com/s?wd=%E5%88%86%E5%B8%83%E5%BC%8F%E8%AE%A1%E7%AE%97&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank)框架分为两个阶段，第一个是map阶段，第二个是reduce阶段。map阶段负责对输入文件进行切分处理，然后汇总再分组给reduce进行处理，以达到高效的[分布式计算](https://www.baidu.com/s?wd=%E5%88%86%E5%B8%83%E5%BC%8F%E8%AE%A1%E7%AE%97&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank)效率

★同样进入：

|  |
| --- |
| <http://192.168.1.11:8088/> |



看到多出了这个，蒙特卡洛方法计算圆周率例程到此完美结束

至此，hadoop全部配置、例程结束，别忘了输入: :

|  |
| --- |
| /usr/local/hadoop-3.2.1/sbin/stop-all.sh |

# Zookeeper

**注：无特殊说明只操作node-1，而node-2、node-3不操作。**

**以下所有步骤后加★意为下面代码可能需要具体修改（加红部分），没检查确认前不要看都不看直接完全copy代码段**

## 2.1 Zookeeper部署

1. 从 zookeeper 官网下载最新版本的安装包。http://zookeeper.apache.org/

|  |
| --- |
| cd |
| wget https://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/stable/apache-zookeeper-3.5.9-bin.tar.gz |

2. 解压安装 zookeeper，随后查看

|  |
| --- |
| tar zxvf apache-zookeeper-3.5.9-bin.tar.gz -C /usr/local/ |
| ls /usr/local/ |

3. 修改配置文件 zoo.cfg

|  |
| --- |
| ls /usr/local/apache-zookeeper-3.5.9-bin/conf/ |
| cp /usr/local/apache-zookeeper-3.5.9-bin/conf/zoo\_sample.cfg /usr/local/apache-zookeeper-3.5.9-bin/conf/zoo.cfg |
| ls /usr/local/apache-zookeeper-3.5.9-bin/conf/ |
| vim /usr/local/apache-zookeeper-3.5.9-bin/conf/zoo.cfg |
| dataDir=/usr/local/apache-zookeeper-3.5.9-bin/zkData |

第二段代码：cp zoo\_sample.cfg zoo.cfg将zoo\_sample.cfg复制到同目录下，并将名称改为zoo.cfg

最后vim是将原有的dataDir**替换**掉

4.新建集群数据目录

|  |
| --- |
| mkdir /usr/local/apache-zookeeper-3.5.9-bin/zkData |
| ls /usr/local/apache-zookeeper-3.5.9-bin/ |

5.启动Zookeeper服务进程

|  |
| --- |
| ll /usr/local/apache-zookeeper-3.5.9-bin/bin |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh start |
| jps |

6.查看Zookeeper服务状态

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh status |

7.启动 Zookeeper 客户端连接服务

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkCli.sh |

8.查看Zookeeper的目录

|  |
| --- |
| ls / |

9. 退出Zookeeper客户端

|  |
| --- |
| quit |

10.退出Zookeeper服务

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh stop |
| jps |

11. 修改集群配置文件

在 zookeeper 的数据目录 zkData 中创建 myid 文件：

|  |
| --- |
| touch /usr/local/apache-zookeeper-3.5.9-bin/zkData/myid |

touch 命令主要用于创建普通文件，如果文件存在，表示修改当前文件时间

★然后修改 zoo.cfg 配置文件(第一行加入即可)：

|  |
| --- |
| vim /usr/local/apache-zookeeper-3.5.9-bin/conf/zoo.cfg |
| #cluster configuration  server.1=node-1:2888:3888  server.2=node-2:2888:3888  server.3=node-3:2888:3888 |

**这里有一个天坑！**注意node-1:2888:3888和node-2:2888:3888和node-3:2888:3888最后一个8后面不要留空格，不要留空格，不要留空格，服务端口号后留了空格非常难检查出来，后面会直接导致zookeeper在解析端口号的时候出现异常，从而使：

这里13. 分别在各个服务器上启动 Zookeeper 服务

出现FAILED而不是STARTED 启动失败！

12. ★分布式部署 Zookeeper(3台)

将node-1配置好的软件包复制到node-2、node-3上（只操作node-1）

|  |
| --- |
| for i in {2..3};do scp -r /usr/local/apache-zookeeper-3.5.9-bin node-$i:/usr/local;done |

可以在node-2、node-3中输入：

|  |
| --- |
| ls /usr/local |

查看到：apache-zookeeper-3.5.9-bin 则复制成功

★**在下一步开启zookeeper前请务必注意！**

**仅操作node-1：**

|  |
| --- |
| vim /usr/local/apache-zookeeper-3.5.9-bin/zkData/myid |
| 1 |

**仅操作node-2：**

|  |
| --- |
| vim /usr/local/apache-zookeeper-3.5.9-bin/zkData/myid |
| 2 |

**仅操作node-3：**

|  |
| --- |
| vim /usr/local/apache-zookeeper-3.5.9-bin/zkData/myid |
| 3 |

13.分别在各个服务器上启动 Zookeeper 服务（若启动失败回看11. ★修改集群配置文件 后面警告）

**（jps如果显示有了QuorumPeerMain就不执行）node-1、node-2、node-3一起操作：**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh start |

14.查看各个服务状态

**node-1、node-2、node-3一起操作：**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh status |

15. 测试故障重新选举 Leader

在步骤14中**node-1、node-2、node-3谁显示leader先操作谁，其余两个原follower之后顺序随意**

这里是**node-3是leader，则仅操作node-3**：

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh restart |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh status |

接下来**仅操作node-1：**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh status |

**仅操作node-2：**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh status |

这里显示node-2变成了leader

结论：原先的leader在restart即故障后，即使重新恢复正常，也不再是leader了，而是由剩下的正常的follower之一继承了leader之位（3台情况下node-3不是leader后node-2会继承）

16. 测试HA(High Available缩写，指高可用性集群)

关闭node-3的Zookeeper 服务，Zookeeper 集群服务仍然可以工作（半数以上服务器有效）

**仅操作node-3：**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh stop |

**接下来仅操作node-1**，查看zookeeper集群是否仍能工作

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh status |

再关闭node-2的Zookeeper服务，整个Zookeeper集群服务失效（低于半数服务器失效）

**仅操作node-2:**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh stop |

**接下来仅操作node-1**，查看zookeeper集群是否仍能工作

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh status |

虽然集群失效了，但是仍然开启的服务器用jps能够观察的到

**（正常不执行）最后node-1、node-2、node-3一起操作：**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh stop |

zookeeper的集群部署和测试到此完美结束，下面是zookeeper的使用（建议做一下，节省时间可跳过，直接进行spark，不影响）

## 2.2 Zookeeper使用（省时间可跳过直接spark，不影响）

### 2.2.1 Zookeeper shell

1.

node-1、node-2、node-3一起打开集群，后面无特殊说明仅在node-1上操作：

**（jps如果显示有了QuorumPeerMain就不执行start）**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh start |

启动客户端连接 Zookeeper 集群

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkCli.sh |
| ls / |

其中ls查看当前 znode 中所包含的内容

2. 测试shell命令：help列出所有可用命令及用法。

|  |
| --- |
| help |
| ls -s / |

①创建普通节点（创建时要写入数据）

|  |
| --- |
| create /testroot |
| ls / |
| ls -s /testroot |
| create /testroot/test1 |
| ls /testroot |
| create /testroot/test2 |
| ls /testroot |

②更新和读取节点目录中的数据

|  |
| --- |
| set /testroot/test1 "Hello test1" |
| set /testroot/test2 "Hello test2" |
| get /testroot |
| get /testroot/test1 |
| get /testroot/test2 |

quit关闭客户端，/usr/local/apache-zookeeper-3.5.9-bin/bin/zkCli.sh + 回车 则是启动客户端

③创建带有序号的znode目录

|  |
| --- |
| create -s /testroot/test3 |
| ls /testroot |

④监听znode的值的变化，启动另一个客户端（注册一次watch只能监听一次）

|  |
| --- |
| get -w /testroot |
| set /testroot "Hello,h02" |

⑤监听znode子目录的变化(注册一次watch只能监听一次)

|  |
| --- |
| ls -w /testroot |
| create -s /testroot/test3 |

⑥删除znode目录

|  |
| --- |
| delete /testroot/test1 |
| ls /testroot |

⑦查看znode详细信息

|  |
| --- |
| stat / |
| stat /testroot |

⑧递归删除 znode 目录

|  |
| --- |
| deleteall /testroot |
| ls / |
| quit |

**（正常不执行）最后node-1、node-2、node-3一起操作：**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh stop |

**这时候node-1、node-2、node-3均备份**

zookeeper集群

zookeeper集群和shell使用已完成，API编程未完成（要IDEA环境），下一步spark

### 2.2.2 Zookeeper API 编程应用（待更新）

(具体安装看视频)输入浏览器下载IntelliJ IDEA:

|  |
| --- |
| <https://www.jetbrains.com/> |

网站打不开解决办法：

<https://blog.csdn.net/n2278556874/article/details/101477069>

破解Ultimate版：<https://npegeek.com/>

##### Ultimate激活码（截至2021.2.18，过时了自己去上面网站找）

|  |
| --- |
| E70JHCOV2H--qlgtO4xVGHX/r45fIKMaR6B9pWQtucrCYVsz0o00crcAiYN1k/kSMygggYl187B0u0jeXQCe4BmQIItKL79x6NwoPn43inreVhZ88f4+Cbl+V/KGeAYeybon+7YoTs8FY4+31ANW/LwBPxkPnlErxYdQ6oc/k6mnxIOm5Nf8WjKRfYYIl5Bhmdt1gHMGgFsocCcTLLiqDUGEcPj5tUIJXwwYaeKAR3YGXm/P73QpnYR/BcGaodBN3jprQRxsS5Ia5y06rrDAJcPSZuttAFpAit/4o/gq2XzhrjaBCtOMxNzk3XEAT82glTlWQOQx6KnRq6D7WUXzd81g44aP+Dca5Q==- |

# Spark

**注：无特殊说明只操作node-1，而node-2、node-3不操作。**

**以下所有步骤后加★意为下面代码可能需要具体修改（加红部分），没检查确认前不要看都不看直接完全copy代码段**

**zookeeper结束后node-1、node-2、node-3均备份**

## 3.1 Spark部署

1. 从spark官网下载最新版本的安装包。http://spark.apache.org/

|  |
| --- |
| cd |
| wget https://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-3.0.2/spark-3.0.2-bin-hadoop3.2.tgz |

2. 解压安装spark，随后查看

|  |
| --- |
| tar -zxvf spark-3.0.2-bin-hadoop3.2.tgz -C /usr/local/ |
| ls /usr/local/ |

3.建立软链接

**node-1、node-2、node-3一起操作：**

|  |
| --- |
| ln -s /usr/local/spark-3.0.2-bin-hadoop3.2 spark |
| ln -s /usr/local/apache-zookeeper-3.5.9-bin zookeeper |
| ll |

相当于：ls /usr/local/spark-3.0.2-bin-hadoop3.2/conf/简化为ls spark/conf/

想删除软链接的时候：rm -rf 目标链接

上面node-2、node-3会爆红，属于正常现象（爆红就是不存在那个目录）

后面步骤会通过scp同步到node-2、node-3上，爆红会消失

4. ★修改配置文件spark-env.sh和slaves

|  |
| --- |
| ls spark/conf/ |
| cp spark/conf/spark-env.sh.template spark/conf/spark-env.sh |
| cp spark/conf/slaves.template spark/conf/slaves |
| ls spark/conf/ |
| vim spark/conf/spark-env.sh |
| export JAVA\_HOME=/usr/java/jdk1.8.0\_281  export SPARK\_WORKER\_CORES=1  export SPARK\_WORKER\_INSTANCES=1  export SPARK\_MASTER\_HOST=node-1  export SPARK\_MASTER\_PORT=7077 |
| vim spark/conf/slaves |
| node-1  node-2  node-3 |

**vim spark/conf/slaves时把localhost删掉**

虚拟机内存默认2G+，若比较小（如1G），需要把conf/spark\_env.sh 中把 SPARK 的

WORKER 的工作内存和 executor 内存降低成 900M 或者更低：

**（正常不运行）**

|  |
| --- |
| vim spark/conf/spark-env.sh |
| export SPARK\_EXECUTOR\_MEMORY=512M  export SPARK\_WORKER\_MEMORY=750M  export SPARK\_DRIVER\_MEMORY=512M |

5. ★将软件包传输至其他节点上

|  |
| --- |
| for i in {2..3};do scp -r /usr/local/spark-3.0.2-bin-hadoop3.2 node-$i:/usr/local;done |

这里把/usr/local/spark-3.0.2-bin-hadoop3.2改成spark没用……

**node-2、node-3一起操作:**

|  |
| --- |
| ll |

可以发现爆红消失

**node-1、node-2、node-3一起操作:**

|  |
| --- |
| ls spark |

6.启动spark 集群(standalone 模式)

下面又**仅操作node-1**：

|  |
| --- |
| spark/sbin/start-all.sh |

查看

**node-1、node-2、node-3一起操作:**

|  |
| --- |
| jps |

★浏览器输入：

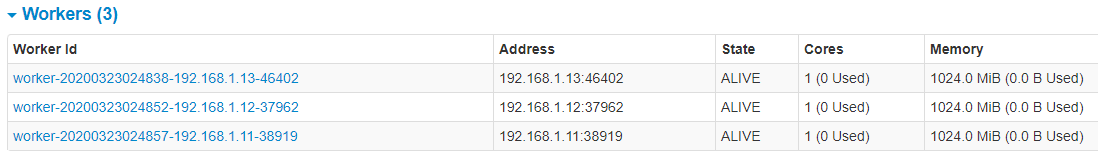
|  |
| --- |
| http://192.168.136.101:8080/ |

如果8081打不开，则输入

**（正常不运行）**

|  |
| --- |
| tail -n 100 spark/logs/spark-root-org.apache.spark.deploy.master.Master-1-node-1.out |

上面tail命令查看日志最后100行，可以看出端口尝试过程，可能是8080，或者8082等等



至此spark安装完成

7. 启用 HA 功能（**zookeeper必须已经配置完！否则无法进行**）

**node-1、node-2、node-3一起操作:**

**或者**

**for i in {2..3};do scp -r /usr/local/spark-3.0.2-bin-hadoop3.2/conf/spark-env.sh node-$i: /usr/local/spark-3.0.2-bin-hadoop3.2/conf;done**

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

将export SPARK\_MASTER\_HOST=node-1和export SPARK\_MASTER\_PORT=7077前面加#注释掉，后面加一行

export SPARK\_DAEMON\_JAVA\_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER -Dspark.deploy.zookeeper.url=node-1:2181,node-2:2181,node-3:2181 -Dspark.deploy.zookeeper.dir=/spark"

即最终变成

|  |
| --- |
| export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-0.el7\_7.x86\_64  export SPARK\_WORKER\_CORES=1  export SPARK\_WORKER\_INSTANCES=1  #export SPARK\_MASTER\_HOST=node-1  #export SPARK\_MASTER\_PORT=7077  export SPARK\_DAEMON\_JAVA\_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER -Dspark.deploy.zookeeper.url=node-1:2181,node-2:2181,node-3:2181 -Dspark.deploy.zookeeper.dir=/spark"  #清理worker  export SPARK\_WORKER\_OPTS="  -Dspark.worker.cleanup.enabled=true # 是否开启自动清理  -Dspark.worker.cleanup.interval=1800 # 清理周期，每隔多长时间清理一次，单位秒  -Dspark.worker.cleanup.appDataTtl=3600" # 保留最近多长时间的数据 |

**（jps三台均显示QuorumPeerMain则不用）node-1、node-2、node-3一起操作**

|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh start |

**下面这两步别跳**，spark关了重新启动一下，不然后面这里的

会只显示[zookeeper]

|  |
| --- |
| spark/sbin/stop-all.sh |
| spark/sbin/start-all.sh |

**在HA设置下从那台主机启动spark，哪台主机就是默认master**

**仅操作node-1：**

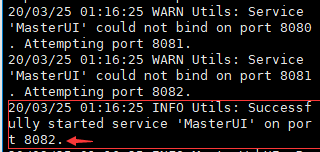
|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkCli.sh |
| ls / |
| ls /spark -s |
| quit |

其中输入ls /后，必须要同时看到spark和zookeeper才能算成功



★**仅操作node-2：**

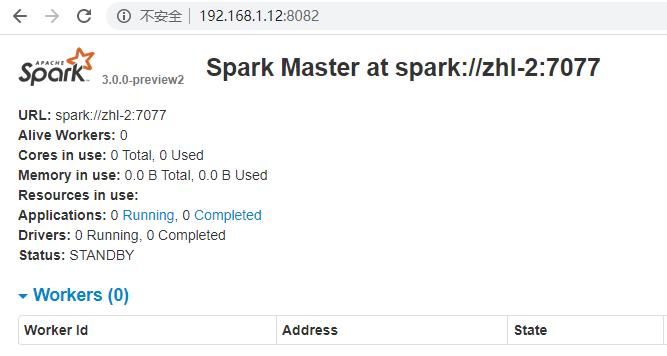
|  |
| --- |
| spark/sbin/start-master.sh |
| tail -n 100 spark/logs/spark-root-org.apache.spark.deploy.master.Master-1-node-2.out |



通过日志找到了node-2使用了8082端口

打开一个浏览器，输入：

|  |
| --- |
| <http://192.168.136.102:8082/> |



与node-1形成了鲜明对比，这里没有一个worker，**这里折腾了这么久的意思就是就是要node-1是主机，node-2是备用主机，然后后面会测试让node-1宕机后恢复正常，然后看看node-2的反应，进而来检验spark的原理**

**node-1、node-2、node-3一起操作:**

|  |
| --- |
| jps |

打开第二个浏览器输入：

|  |
| --- |
| <http://192.168.136.101:8081/> |

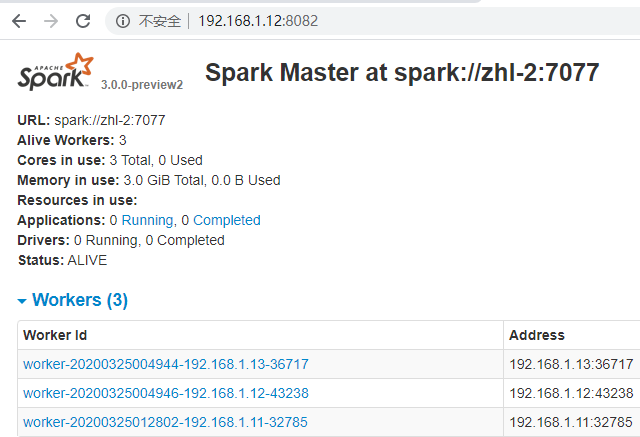


★测试node-1的master进程宕机（此时node-2是备用主机）

|  |
| --- |
| jps |
| kill -9 5424 |

kill -9 进程号 要用jps具体查看具体修改

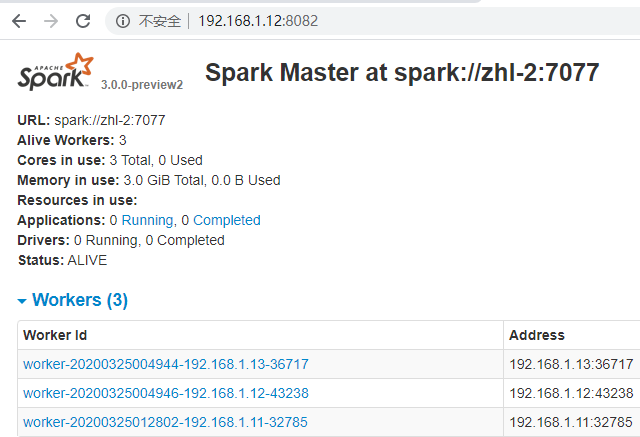
不断刷新浏览器页面，此时node-1的页面已经无法打开，node-2的要多刷新几次（等一会儿）



**也就是此时node-2（备用主机）完全继承了原先主机node-1的功能！**所以正常的想法接下来恢复node-1（原主机），看看继承的功能会不会还回去

然后重新启动node-1的master进程

|  |
| --- |
| spark/sbin/start-master.sh |

**也就是说此时原先的备用主机（node-2 STANDBY）变成了实质上的主机，而原先的主机（node-1 ALIVE）变成了实质上的备用主机，继承的功能并不会随原主机的恢复而还回去！这就是spark集群的特点。**

8. ★运行例程蒙特卡洛计算圆周率

|  |
| --- |
| spark/bin/spark-submit --master spark://node-1:7077,node-2:7077,node-3:7077 --class org.apache.spark.examples.SparkPi spark/examples/jars/spark-examples\_2.12-3.0.2.jar 10000 |

10000太大嫌慢了，改小点，运行起来快，但是Π小数点后的精度会下降



这是数值为10000时候的结果（其实spark运行起来明显发现比hadoop mapreduce快很多）

这里有这样一组数据供参考：相同的运算次数的情况下，spark用时4.6s，而Hadoop MapReduce 用时35s

有人对上面这段代码或许会问：master和worker有什么区别，运行例程必须要在master上吗？

**答：不是这样的。master只是管理作业的和产生运行调度表的，实际运行是在各个worker中excutor进程完成的，不用管哪个master是alive（实质上的主机）了,系统会自动找到alive的master并提交任务的。**

**更进一步来讲，上面7. 启用HA功能只是为了测试方便，所以只弄了两台node-1、node-2，本来这段代码中间应该是--master spark://node-1:7077,node-2:7077 但是真实情况是会连node-3也要写进去的（其中一台是alive，其他全是standby），所有都写的好处在于不去指定，让系统自己找，因为作为客户端大多数时候是没有权限去查看spark的状态的。**给一个zk的服务器上下线的例子也是让大家看看zk如何为其他组件提供HA功能的简单原理。

此时观察（即谁是现在实质上的主机master就观察谁，现在是node-2）：

|  |
| --- |
| <http://192.168.1.12:8082/> |



上图是正在运行例程中



上图是例程已经运行完毕

★这里示范以下强行设定内存2G执行会怎么样（内存不够）

|  |
| --- |
| spark/bin/spark-submit --master spark://node-1:7077,node-2:7077,node-3:7077 --class org.apache.spark.examples.SparkPi --executor-memory 2G spark/examples/jars/spark-examples\_2.12-3.0.0-preview2.jar 1000 |

WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources

没有足够的资源，所以报错

到此为止，spark的部署和例程示范完美结束。

**（正常下不执行）**如果关机则关机前记得：

|  |
| --- |
| spark/sbin/stop-all.sh |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh stop |
| /usr/local/hadoop-3.2.1/sbin/stop-all.sh |

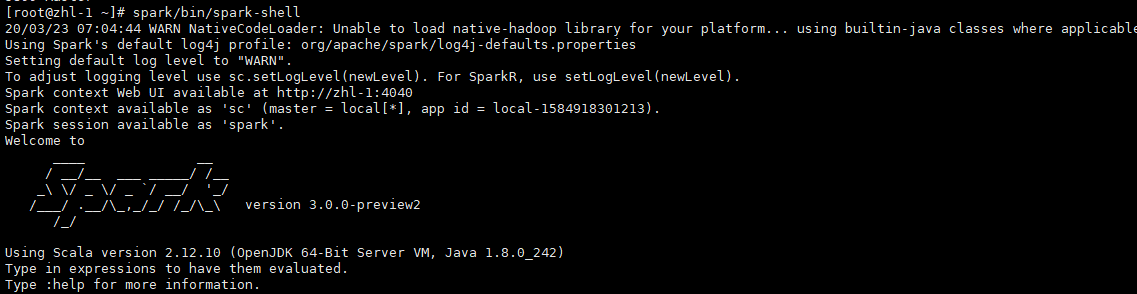
**上面黄色部分node-1、node-2、node-3一起操作**

## 3.2 Spark使用

### 3.2.1 Spark shell

1. Spark 本地运行模式(local)

|  |
| --- |
| spark/bin/spark-shell |



**ctrl+z退出后仅操作node-1**：

|  |
| --- |
| jps |



本机运行模式下，使用的是SparkSubmit进程来运行 Spark

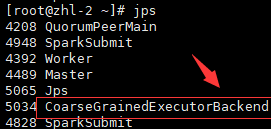
2. Spark 集群运行模式（standalone）

|  |
| --- |
| spark/bin/spark-shell --master spark://node-1:7077,node-2:7077,node-3:7077 |

跟本地区别就在于后面多加了--master spark：xxxxx 体现在：

随便选一台比如**仅运行node-2**看看：

|  |
| --- |
| jps |



Standalone集群模式，使用的是CoarseGrainedExecutorBackend进程来运行Spark任务

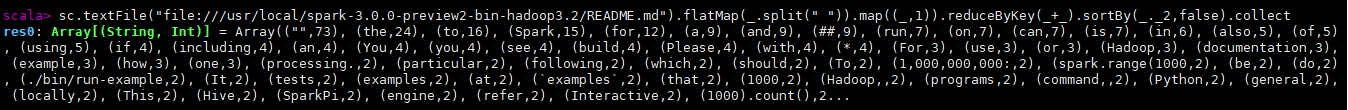
1. 提交 job

在后面输入：

|  |
| --- |
| sc.textFile("file:///usr/local/spark-3.0.2-bin-hadoop3.2/README.md").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).sortBy(\_.\_2,false).collect |

运行完成退出spark shell 命令。：quit。如果不退出，会继续占用资源。发现警告WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources

就到浏览器把多余占用的kill掉



最后成功运行

然后读取分布式文件系统 HDFS 中的文件（hadoop必须也已经配置完成，否则无法进行）

确保node-1、node-2、node-3已经处于开启状态（**第一行若运行3台均操作，其余行若运行只操作node-2**）：

**（一般来讲第三行hadoop还没开启，仅运行第三行，其余正常不运行）**

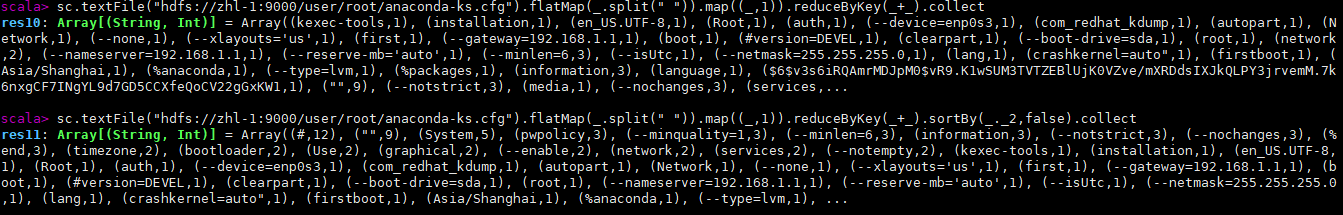
|  |
| --- |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh start |
| spark/sbin/start-all.sh |
| /usr/local/hadoop-3.2.1/sbin/start-all.sh |

**标黄仅操作node-2**

|  |
| --- |
| hdfs dfs -mkdir /test/root |
| hdfs dfs -put /root/anaconda-ks.cfg /test/root/ |
| spark/bin/spark-shell --master spark://node-1:7077,node-2:7077,node-3:7077 |
| sc.textFile("hdfs://node-1:9000/user/root/anaconda-ks.cfg").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).collect |
| sc.textFile("hdfs://node-1:9000/user/root/anaconda-ks.cfg").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).sortBy(\_.\_2,false).collect |

hdfs dfs -put /root/anaconda-ks.cfg /user/root/将 /root/anaconda-ks.cfg复制到/user/root/路径下

hdfs dfs -rm -r 路径 可以删除对应路径hdfs文件



如果报错：java.net.ConnectException: Call From node-1/192.168.1.11 to node-1:9000 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: <http://wiki.apache.org/hadoop/ConnectionRefused>

那就是还没把hadoop开起

最后关机前关闭**（黄色node-1、node-2、node-3均运行）**

|  |
| --- |
| spark/sbin/stop-all.sh |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh stop |
| /usr/local/hadoop-3.2.1/sbin/stop-all.sh |

到此spark完美结束

**这时候node-1、node-2、node-3均备份**

spark集群

spark集群和shell使用已完成，API编程未完成（要IDEA环境），下一步IDEA

### 3.2.2 Spark API 编程应用（待更新）

启动spark集群**（黄色node-1、node-2、node-3均运行）**

|  |
| --- |
| /usr/local/hadoop-3.2.1/sbin/start-all.sh |
| /usr/local/apache-zookeeper-3.5.9-bin/bin/zkServer.sh start |
| spark/sbin/start-all.sh |