大数据开发技术期末总结

180110910603

陈彬捷

# 课程介绍

大数据开发课程主要从了解大数据概念、特征开始，再介绍大数据Java开发和Hadoop的环境配置，较为全面地讲解了HDFS分布式存储，MapReduce分布式计算框架，Spark平台开发和Scala编程语言.归根结底，我总结了一下本学期的大数据开发技术课程老师主要从五个方面进行了教学和指导：

1. HDFS使用操作；
2. MapReduce开发；
3. Spark开发；
4. Scala语言；
5. 大数据经典程序分析.

# 系统搭建步骤

## 准备文件

* VMware Workstation，下载地址：

<https://my.vmware.com/cn/web/vmware/info/slug/desktop_end_user_computing/vmware_workstation_pro/15_0>

* Centos7，下载地址：

<http://mirrors.aliyun.com/centos/7.7.1908/isos/x86_64/>

* 下载CentOS-7-x86 64-DVD-1908.iso.
* Spark，这里使用的版本是Spark2.4.5，下载地址：

<http://spark.apache.org/downloads.html>

* Hadoop，这里使用的版本是Hadoop2.7，下载地址：

<https://www.apache.org/dyn/closer.cgi/hadoop/common>

* Java，这里使用的版本是Java7.
* Scala，这里使用的版本是scala-2.11.12.
* CTeX，这里使用的版本是CTeX\_2.9.2.

以上所有资源链接：https://pan.baidu.com/s/12R7weK8SHBWjdLdjs7sv7g

提取码：0w5w

## Hadoop集群搭建

### **安装软件VMware Workstation**

进入VMware Workstation官网.

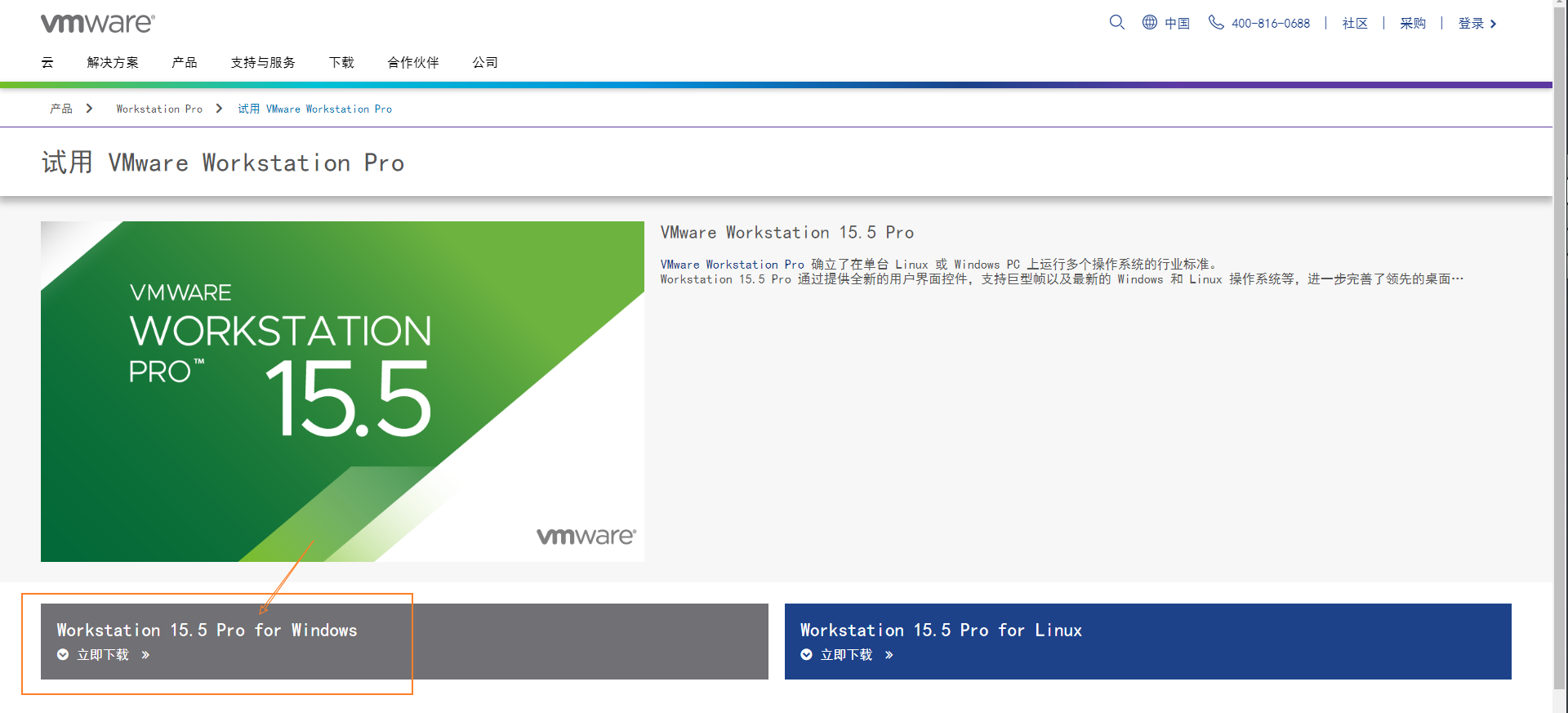


图 2‑1 VMware Workstation官网截图

单击之后即可下载安装VMware Workstation.

若要下载其他版本，则点击“下载”菜单栏，选择“产品下载”中的“WordstationPro”.

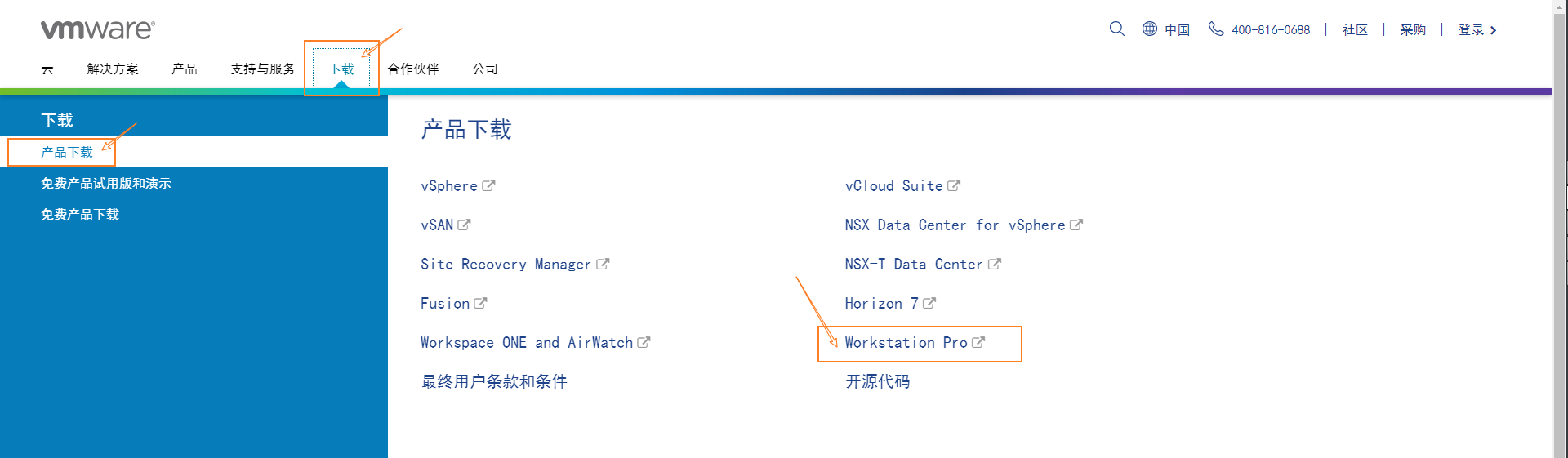


图 2‑2 点击 VMware Workstation “产品下载”目录

点击后可以选择需要下载的版本进行下载.



图 2‑3 选择对应的文件进行下载

下载完成之后，双击安装包，进入安装页面.



图 2‑4 VMware Workstation产品安装首页面

点击“下一步”.

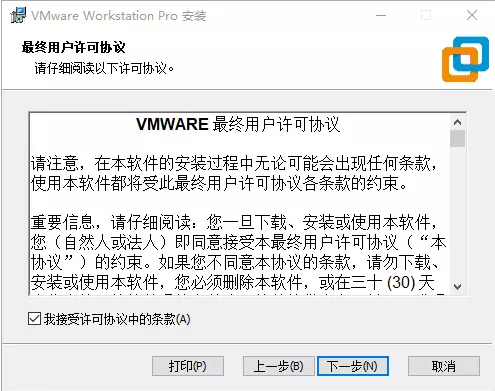


图 2‑5 VMware Workstation接受许可协议

勾选接受协议，点击“下一步”.

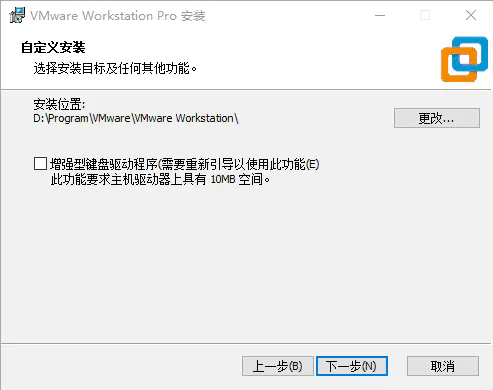


图 2‑6 VMware Workstation更改安装位置

更改安装位置，选择非C盘的文件夹，

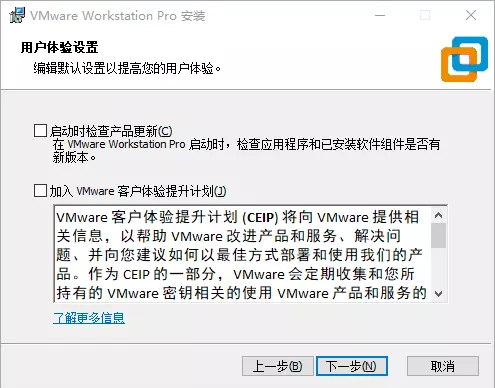


图 2‑7 VMware Workstation用户体验设置

不勾选俩个复选框，直接点击“下一步”

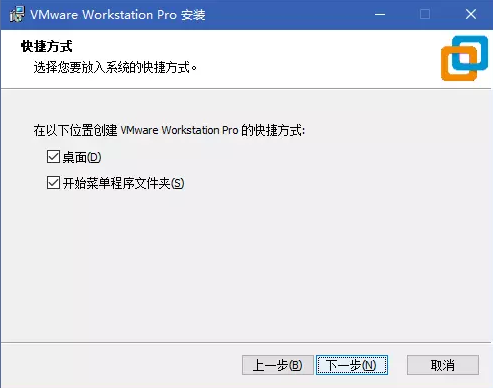


图 2‑8 VMware Workstation产品安装快捷方式

点击“下一步”，等待安装完成.

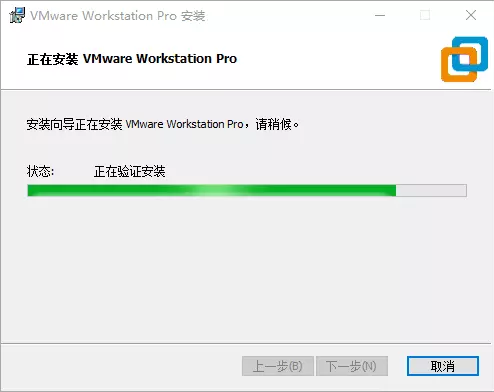


图 2‑9 VMware Workstation安装进程

安装完成之后输入许可证密钥：

*YG5H2-ANZ0H-M8ERY-TXZZZ-YKRV8*

*UG5J2-0ME12-M89WY-NPWXX-WQH88*

*UA5DR-2ZD4H-089FY-6YQ5T-YPRX6*

### **安装CentOS7的图形化界面**

首先右键以管理员身份运行VMware.



图 2‑10 右键以管理员身份运行VMware

点击创建新的虚拟机.



图 2‑11 创建新的虚拟机



图 2‑12 新建虚拟机向导

选择“典型（推荐）”，然后点击“下一步”.

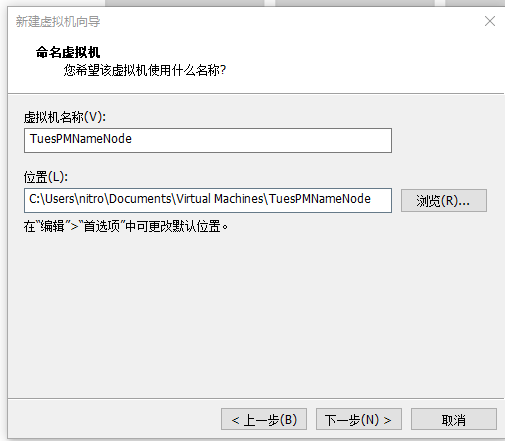


图 2‑13 虚拟机命名

给虚拟机起个名字，选择安装位置.第一个考虑是把虚拟机与宿主机放到不同的物理硬盘上，第二个考虑是硬盘读写速度.建议硬盘空余空间>40G.点击“下一步”.



图 2‑14 虚拟机定义磁盘容量

建议最大磁盘大小>40G，原因：因为这个节点除了要安装集群文件，还要安装开发系统.勾选“将虚拟磁盘存储为单个文件”，这样运行速度比较快.为了安装速度比较快，点击“自定义硬件”，根据自己电脑配置选择内存大小和CPU核心数.

* 电脑运行内存8G，建议内存选择2G(2-48MB），其他节点1G
* 处理器数量4个(因为双线程显示8个)，建议处理器数量2，其他处理器数量1个



图 2‑15 自定义硬件参数

一定要勾选**“虚拟化Intel VT-X/EPT 或AMD-V/RVI(V)”**.

设置完成，关闭，开始安装进程.

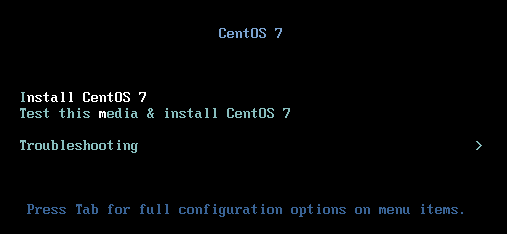


图 2‑16 CentOS 7 安装过程

安装完成之后，鼠标点击界面，按键盘上“i”键，如果要释放鼠标，按“alt+ctrl”组合键.

选择安装过程中的语言，点击“continue”.

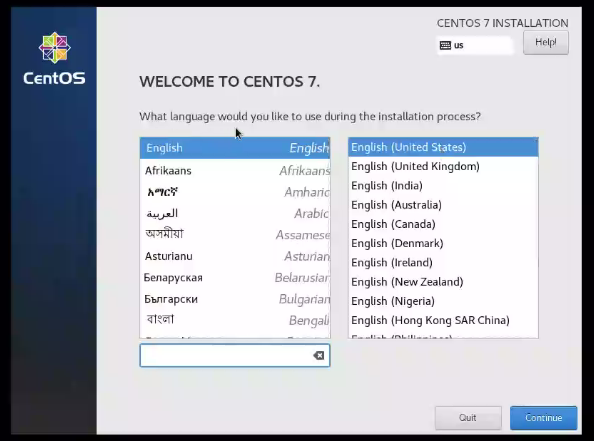


图 2‑17 CentOS 7 安装语言选择

出现这个界面后，需要等待系统进行安装前的检测，稍等片刻看哪个项目还有感叹号.

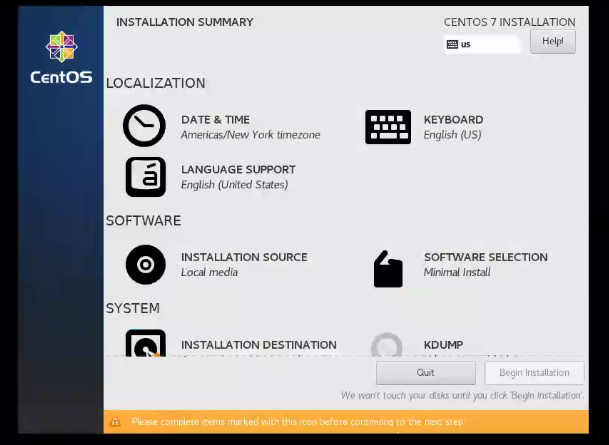
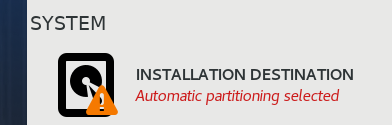


图 2‑18 等待系统检测-a



图 2‑19 等待系统检测-b

点击这个有感叹号的项目  进行分区选项.

不需任何改动，全部使用默认选项，点击“Done”.

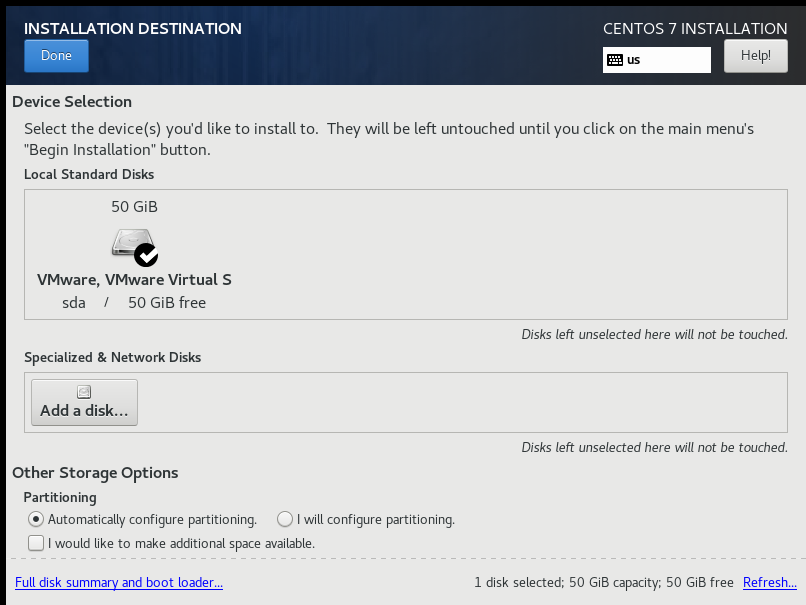


图 2‑20 分区保持默认选项

点击图示位置，选择安装软件的内容.

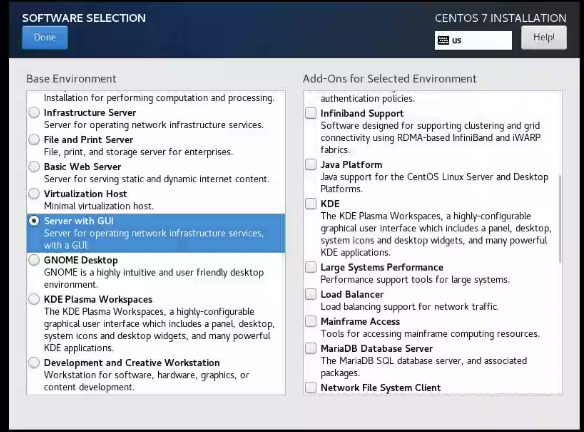


图 2‑21 选择软件内容

NameNode选择“Server with GUI”，然后点击“Done”.

系统会重新进行检测，当”Begin Installation”可以点击时，点击进行安装.

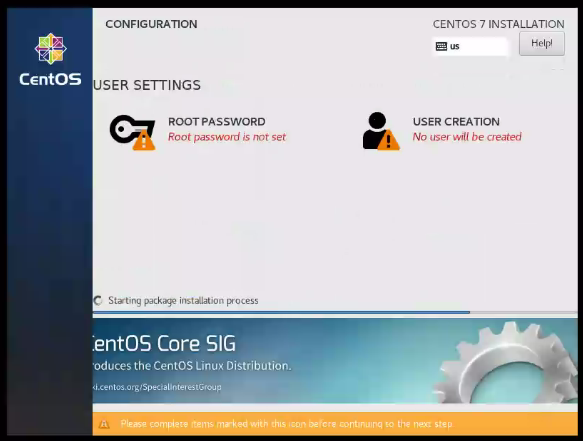


图 2‑22 系统重新检测

点击“ROOT PASSWORD”，设置root密码，一定要记住（记不住只能重装了，所以必须记住），设置完之后点击”USER CREATION”，设置用户名和密码，要勾选“Make this user administrator”，将不能够对系统进行相应设置，点击了可以给该用户相应权限，设置完之后点击“Done”，

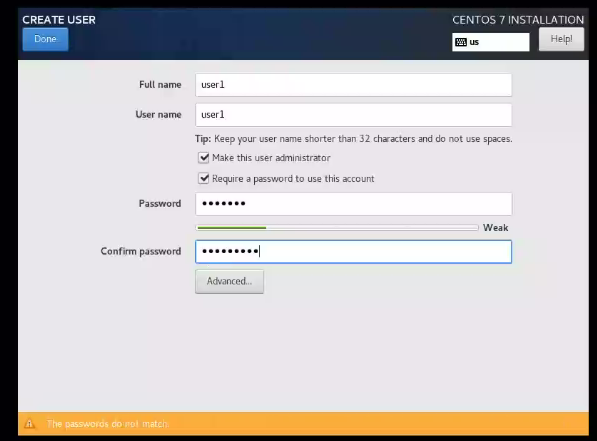


图 2‑23 设置用户姓名和密码

系统进行安装.系统安装大概需要五分钟，安装完成后点击“Reboot”按钮重启.

点击，接受协议.

完成之后开启虚拟机，然后点击最下面的“我已完成安装”按钮，vmware将自动安装一些工具.

安装完成后，配置网络.

### **安装CentOS7的最小安装版**

（在自己电脑上安装集群的特殊操作不常用，如果是使用四台电脑联立集群的话，请跳过此步骤）

大致步骤同第（2）步，选择硬盘容量不小于30G.

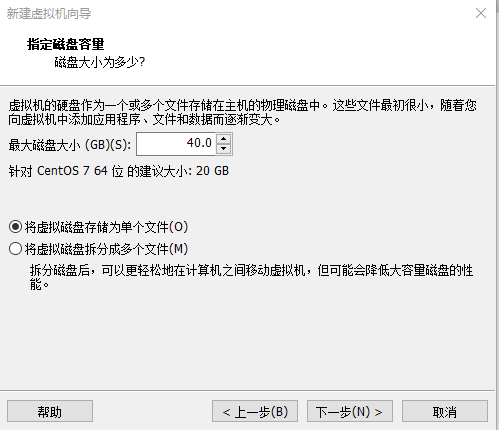


图 2‑24 指定虚拟机磁盘容量

自定义硬件中，勾选“虚拟化Intel VT-x/EPT或AMD-V/RVI(V) ”.

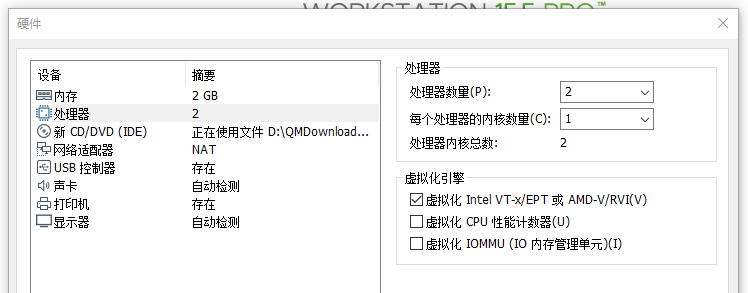


图 2‑25 自定义硬件参数设置

在此界面中，“SOFTWARE SELECTION”保持默认，因为默认就是最小安装.

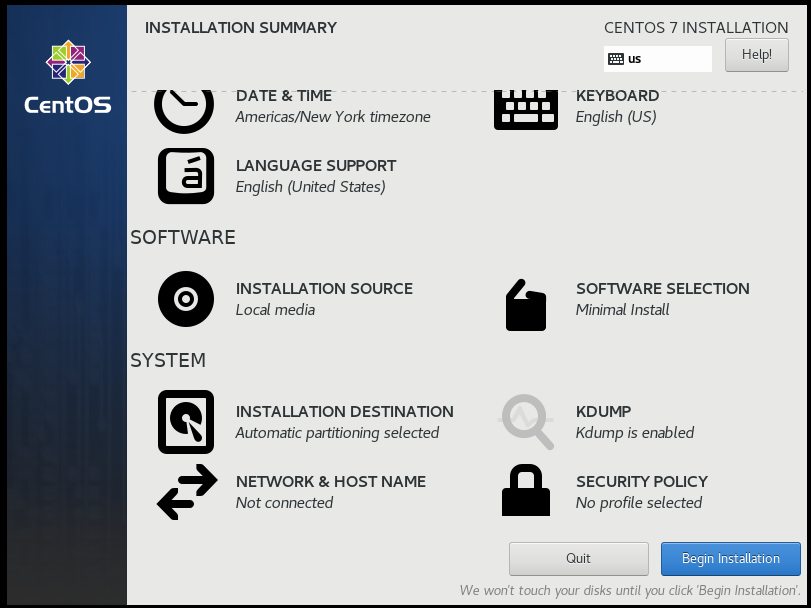


图 2‑26 CentOS 7 安装设置

安装完成后，配置网络.

进入/etc/sysconfig/network-scripts目录，使用命令:

cd /etc/sysconfig/network-scripts



图 2‑27 网卡号文件信息

使用ls命令查看与网卡号相关的文件.使用ip a命令查看网卡号.

一般命令模式下，按i或s键进入编辑模式，按ESC进入一般命令模式.在一般命令模式下，按冒号键进入底行命令模式.

在相应目录下，执行sudo vi ifcfg-ens33，进入编辑页面，按i键，进入编辑模式，把“ONBOOT=no”改成“ONBOOT=yes”，然后按ESC进入一般命令模式，再按冒号，进入底行命令模式，输入wq，回车退出.

重启虚机：sudo reboot

ONBOOT=no表示启动时不激活，改成yes后，就变成启动时激活.

重启后用ping命令检查是否能够上网，例如：ping [www.zufe.edu.cn](http://www.zufe.edu.cn)

Vmware虚拟机网络配置：

打开编辑菜单里的“虚拟网络编辑器”，正常情况是这样的：



图 2‑28 虚拟网络编辑器

如果异常，确定是否以管理员身份启动的vmware，如果是，点击“还原默认设置”.

如果要在NAT模式修改配置IP地址，选中NAT模式：

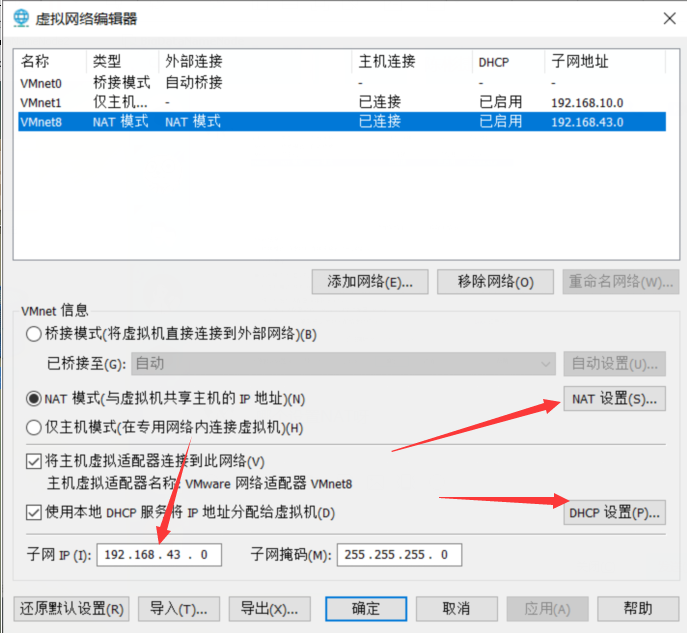


图 2‑29 选中NAT模式

在子网IP栏内输入自己想要的网络地址， 如：10.0.0.0

设置静态ip（在GUI下），打开如下界面：

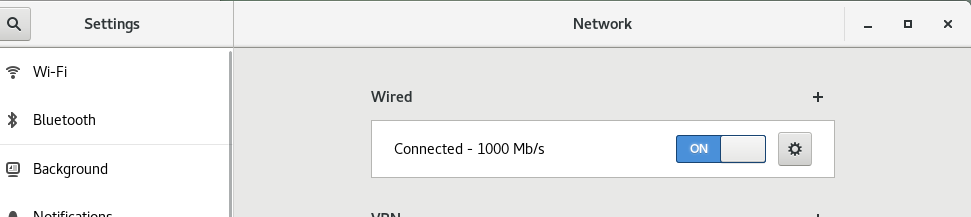


图 2‑30 设置静态IP

点击齿轮，进入设置页面，选中IPV4：

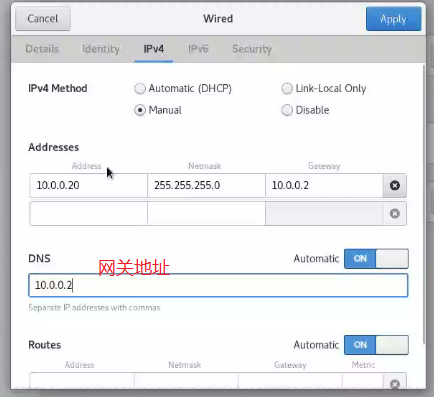


图 2‑31 进入设置页面，选中IPV4

按图示输入，点击“Apply”，回到“Network”界面，关闭网络再打开，ip地址就生效了.

打开网络配置文件，内容如下：

*TYPE=Ethernet*

*PROXY\_METHOD=none*

*BROWSER\_ONLY=no*

*BOOTPROTO=none*

*DEFROUTE=yes*

*IPV4\_FAILURE\_FATAL=no*

*IPV6INIT=yes*

*IPV6\_AUTOCONF=yes*

*IPV6\_DEFROUTE=yes*

*IPV6\_FAILURE\_FATAL=no*

*IPV6\_ADDR\_GEN\_MODE=stable-privacy*

*NAME=ens33*

*UUID=485a313b-0242-4d61-b54f-4ce88077a7fd*

*DEVICE=ens33*

*ONBOOT=yes*

*IPADDR=10.0.0.20*

*PREFIX=24*

*GATEWAY=10.0.0.2*

*DNS1=10.0.0.2*

观察文件内容，红色部分是有变化的.

重新打开虚拟机，验证是否能够上网，查看IP地址.输入命令ip a，得到如下所示结果：

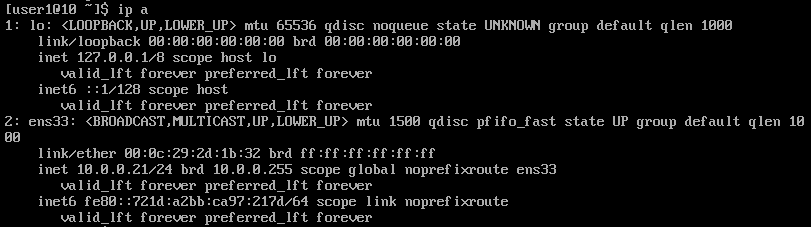


图 2‑32 验证虚拟机是否能够上网

然后，进行系统更新.系统更新命令：sudo yum update.

修改主机名

编辑/etc下的hostname文件，将文件内容全部删除，把自己想要的主机名输入，注意不要有空行.

命令：cd /etc，sudo vi hostname；

编辑完文件后，执行：sudo reboot，进行重启.

### **确定Spark、Hadoop、Java版本**

进入Spark官网：<http://spark.apache.org/>，官网首页没有版本信息，根据经验，点击“download”，进入下载页面：<http://spark.apache.org/downloads.html>，页面信息如下：

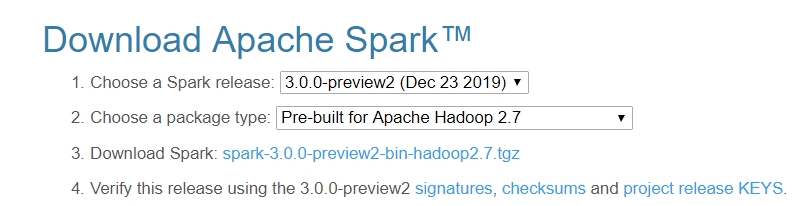


图 2‑33 Spark“download”页面

从此页面上看，似乎是应该下载Spark3.0.0，但此版本标识preview2，应该不是稳定版，作为初次学习之用，必须安装稳定版.需要确定此信息.在此页面：

<http://spark.apache.org/news/spark-3.0.0-preview2.html>，

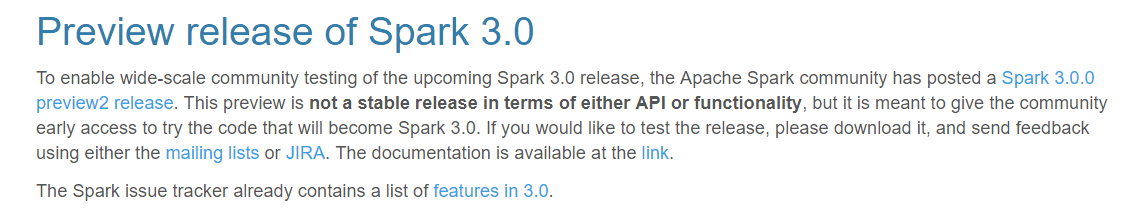


图 2‑34 Spark3.0.0版本介绍

从此页面上可以看到，Spark2.4.5对应Hadoop2.7，下面核实Spark2.4.5是否是稳定版.

点击页面下部的Spark2.4.5的链接，进入到页面：

<http://spark.apache.org/releases/spark-release-2-4-5.html>



图 2‑35 Spark2.4.5版本介绍

页面上明确Spark2.4.5是正在维护的稳定版.

结论：选择Spark2.4.5，Hadoop2.7.

进入Hadoop主页：<http://hadoop.apache.org/>.

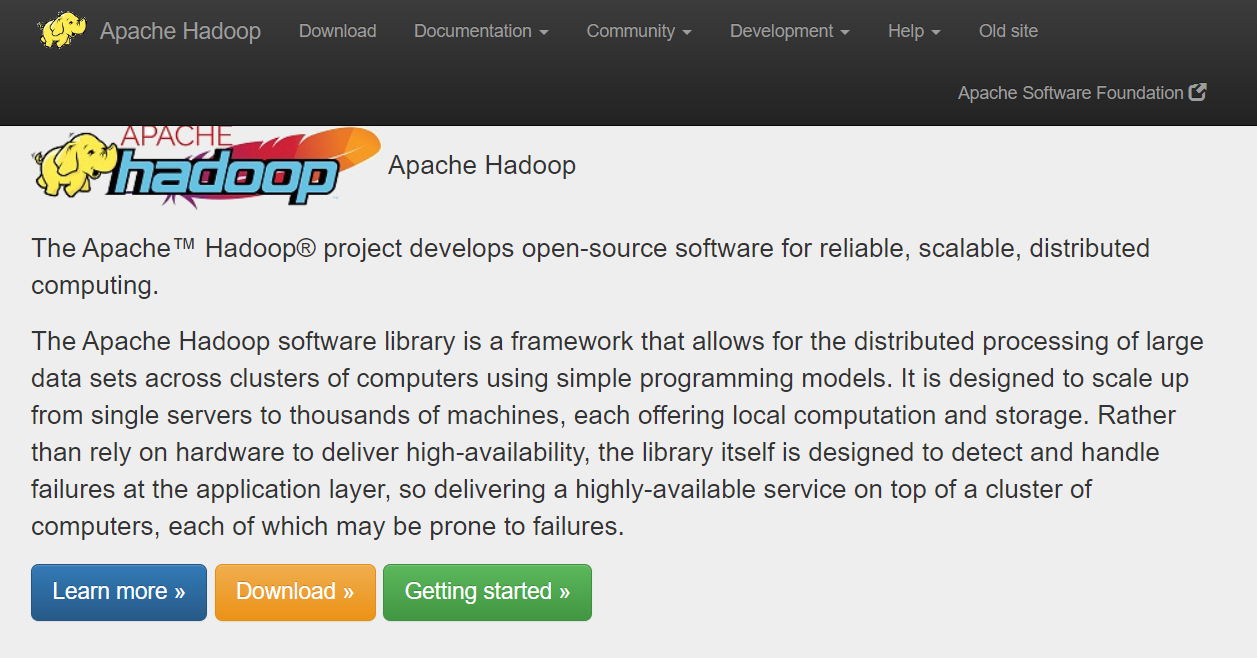


图 2‑36 Hadoop主页

按经验，点击“download”按钮进入下载页面：<https://hadoop.apache.org/releases.html>.

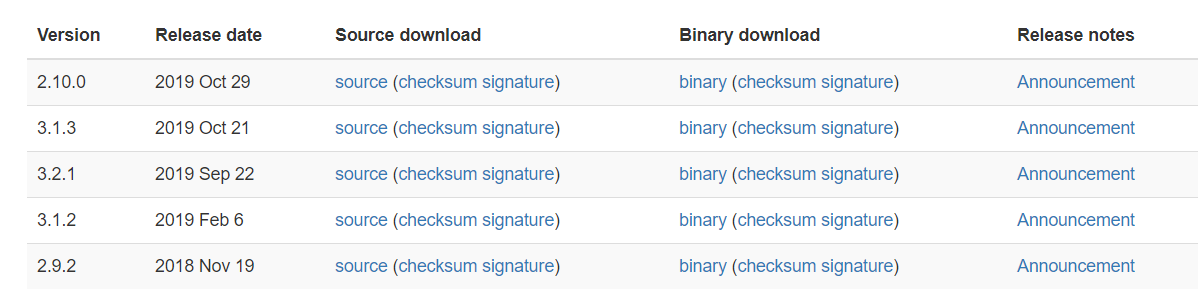


图 2‑37 Hadoop“download”界面

下载页面中没有我们需要的2.7版本，也没有相关信息，所以回到主页.

点击“Getting start”按钮，进入页面：

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html>

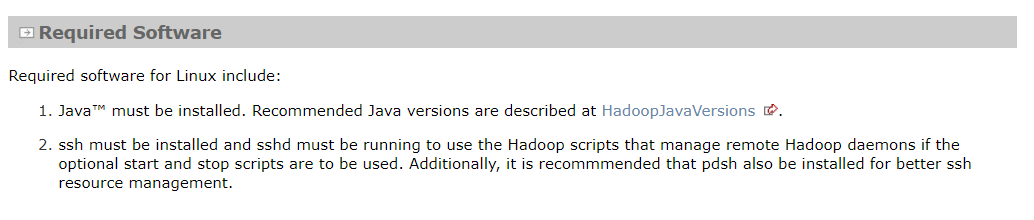


图 2‑38 Hadoop Required Software 介绍

在此项目上有Java版本说明

点击链接：

<https://cwiki.apache.org/confluence/display/HADOOP2/HadoopJavaVersions>

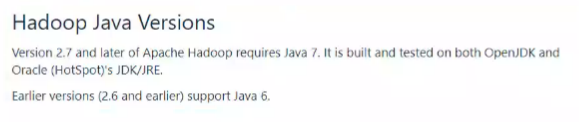


图 2‑39 Java版本说明

从页面说明上，需要Java7版本

得出最终结论：Spark2.4.5，Hadoop2.7，Java7

下载Hadoop2.7，由于下载页面上没有此版本，打开页面上的“mirror sit”链接，进入镜像网站<https://www.apache.org/dyn/closer.cgi/hadoop/common> 下载2.7版本

### **准备Hadoop的后续节点**

后续节点由前面步骤创建的节点直接拷贝而来，如果是自己电脑上的集群就拷贝CentOS最小安装版，所以开始之前，先前创建的节点必须执行sudo poweroff命令关机.

然后，复制整个虚拟机目录，修改目录名.

从vmware主页开始，点击“打开虚拟机”.

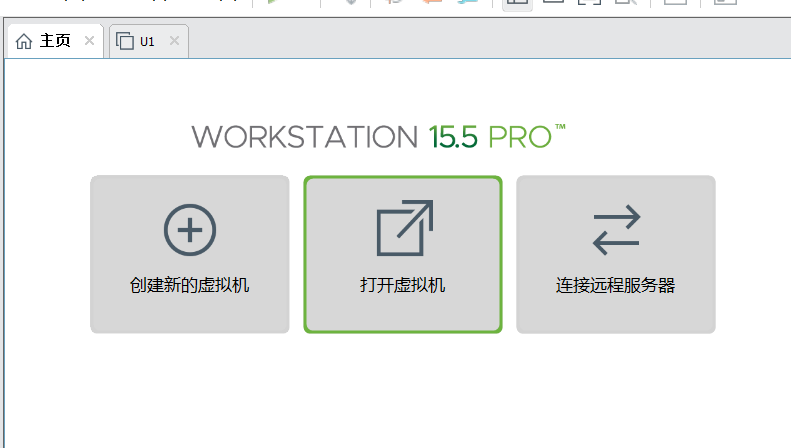


图 2‑40 打开虚拟机

打开后修改虚拟机名字，修改ip地址，修改主机名.修改完成后重启：sudo reboot.

如果是四台物理机的话就要拷贝虚拟机映像.在开启虚拟机之前，设置虚拟机内存是4G，网络连接是桥接模式.

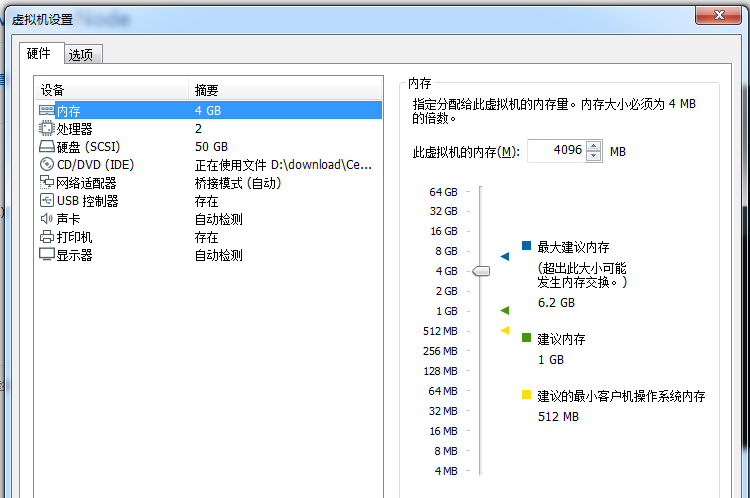


图 2‑41 虚拟机设置-a

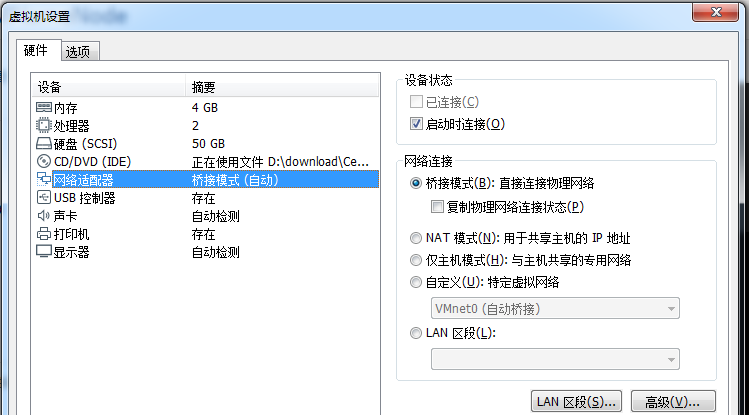


图 2‑42 虚拟机设置-b

把桥接网卡绑定到具体的物理网卡，不要使用自动绑定.

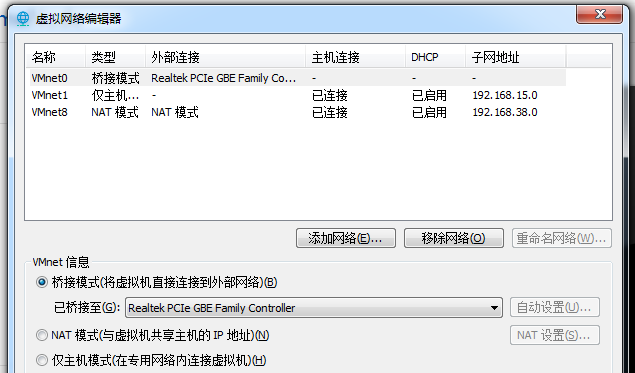


图 2‑43 选择“桥接模式”

设置完成后，启动虚拟机，设置网络连接.

编辑网络配置文件：/etc/sysconfig/network-scripts/ifcfg-ens33，

输入命令：sudo vi /etc/sysconfig/network-scripts/ifcfg-ens33

IP地址设置如下：

*IPADDR=172.21.12. //你自己的ip*

*PREFIX=24*

*GATEWAY=172.21.12.254*

*DNS1=172.16.3.8*

验证各个节点是否能够上网.

1. **设置hostname和hosts**

（如果是在自己电脑上，则只需要俩个CentOS最小安装版虚拟机，步骤相同）

修改集群各个机器的名字和域名解析.

编辑/etc下的hostname文件，将文件内容全部删除，把自己想要的主机名输入，注意不要有空行.

命令：cd /etc，sudo vi hostname；

编辑完文件后，执行：sudo reboot，进行重启.

节点的准备情况（示例）如下：

10.0.0.20 master

10.0.0.21 slave0

10.0.0.22 slave1

10.0.0.23 slave2

确认网络连通正常，使用ssh相互登陆，例如从master节点执行：ssh 10.0.0.21，然后执行exit；再执行：ssh 10.0.0.22，然后执行exit；再执行：ssh 10.0.0.23，然后执行exit.

命令解释：ssh 用户名@IP地址

域名解析，编辑/etc下hosts文件，不要把原来内容删除，直接添加如下内容：

10.0.0.20 master

10.0.0.21 slave0

10.0.0.22 slave1

10.0.0.23 slave2

所有节点都添加同样的内容.

然后验证域名设置是否成功，验证方法：

* 从master节点执行ssh slave0，然后执行exit，再执行：ssh slave1，然后执行exit，再执行：ssh slave2；
* 从slave0节点执行ssh master，然后执行exit，再执行：ssh slave1，然后执行exit，再执行：ssh slave2；
* 从slave1节点执行ssh master，然后执行exit，再执行：ssh slave0，然后执行exit，再执行：ssh slave2；
* 从slave2节点执行ssh master，然后执行exit，再执行：ssh slave0，然后执行exit，再执行：ssh slave1.

### **关闭防火墙设置**

在所有节点上关闭防火墙.

关闭防火墙命令：sudo systemctl stop firewalld.service

仅仅关闭防火墙，下次启动时还会打开，所以还需要disable防火墙，命令：

sudo systemctl disable firewalld.service

查看防火墙状态：sudo systemctl status firewalld.service，如果防火墙状态时inactive（dead），就说明关闭成功了.

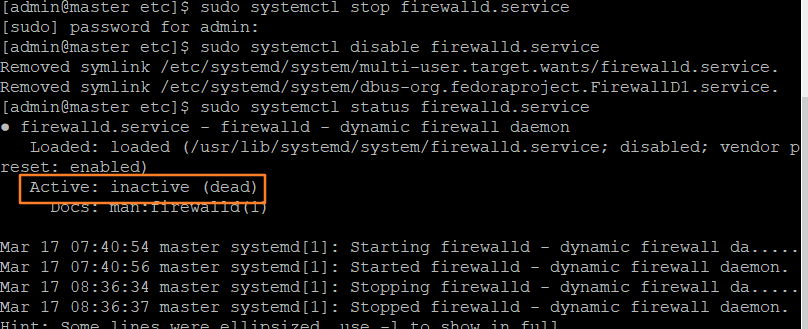


图 2‑44 查看防火墙状态

### **设置时钟同步（如果是在自己电脑上则可直接跳过此步）**

将时区设置为上海：sudo timedatectl set-timezone Asia/Shanghai

使用chrony.因为centos7已经内置了chrony，所以只要配置就可以了.

选择一个本地时钟服务器，例如master，对于时钟服务器修改/etc/chrony.conf，要允许其他节点进行连接：

allow 172.21.12.0/24.

从节点时钟服务器地址修改成master，其他的注释掉，用#.

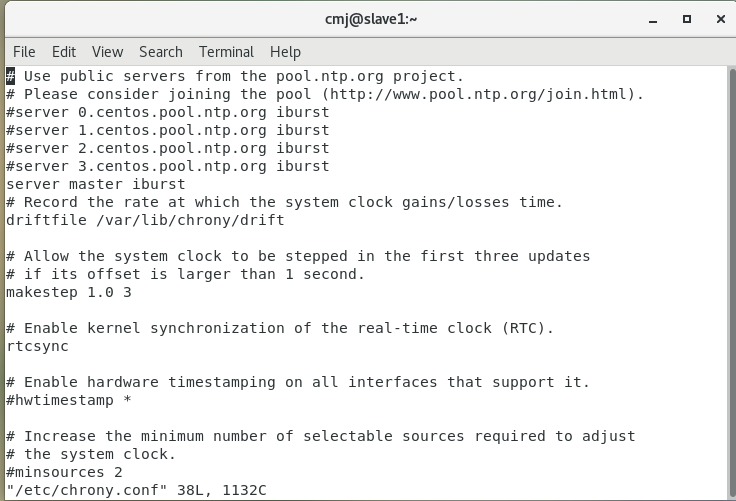


图 2‑45 配置chrony文件

所有节点，执行

sudo systemctl enable chronyd.service

sudo systemctl start chronyd.service

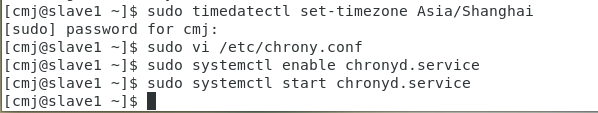


图 2‑46 设置时钟同步

重启所有节点.

在主节点上运行：chronyc sources

可以看到各个时钟服务器的延迟时间.

在从节点上运行：chronyc sources

可以看到连接到master的延迟时间，如果不是0，就说明连接正常了.

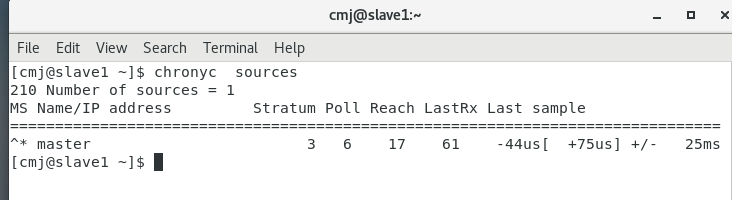


图 2‑47 查看连接到master的延迟时间

### **设置ssh免密登入**

Hadoop 是由很多台服务器所组成的， 当我们启动 Hadoop 系统时， NameNode必须与 DataNode 连接并管理这些节点 (DataNode).此时系统会要求用户输入密码.为了让系统顺利运行而不手动输入密码，需要将 SSH 设置成无密码登录.注意，无密码登录并非不需要密码，而是以事先交换的 SSH Key (密钥）来进行身份验证.

先ssh登入要登入的虚拟机，然后exit，进入目录 ： cd .ssh/，产生密码 ： ssh-keygen -t rsa传送公钥 ： ssh-copy-id要登入的虚拟机名，验证 ： ssh 要登入的虚拟机名，如果不需要密码就免密成功了.

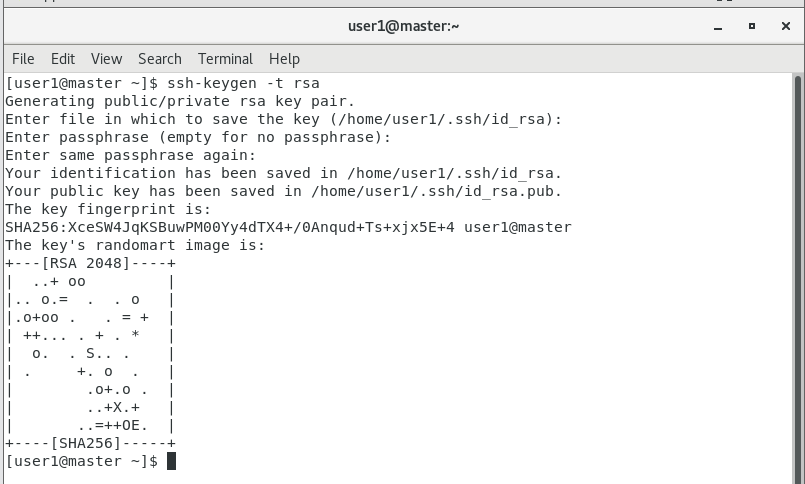


图 2‑48 ssh产生密钥

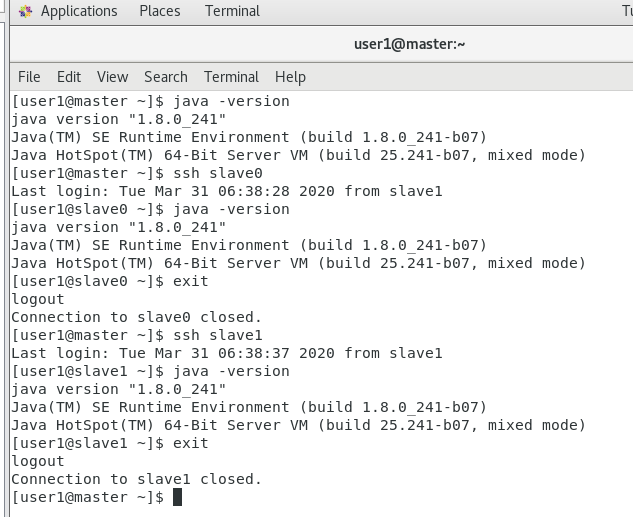


图 2‑49 验证免密操作

四个节点，每个虚机上需要设置四个免密登入（自己和其他三个节点）.

### Java安装

首先核实当前java版本.如果当前已经有java了，核实版本号java -version，如果可用，不用另外安装，如果不可用，不能直接替换此版本，因为有可能造成系统故障.

只是本账户使用使用我们的java版本.

建议不要用右键解压，使用命令解压，命令：tar –zxvf 文件名.

配置环境，编辑本用户名下的.bash\_profile文件，添加JAVA\_HOME变量，并把变量下的bin添加到系统PATH变量中.

新增加的：

export JAVA\_HOME=~/jdk1.8

export PATH=$JAVA\_HOME/bin/:$PATH

其他节点，首先将文件拷贝过去：例如：

scp -r jdk1.8 slave0:~

scp -r jdk1.8 slave1:~

然后编辑.bash\_profile文件.重启后，核实版本号java –version，我们新安装的版本是1.8：

java version "1.8.0\_241"

Java(TM) SE Runtime Environment (build 1.8.0\_241-b07)

Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07， mixed mode)

**每个节点都需要配置jdk**.

### 配置Hadoop集群

1. 使用命令解压hadoop文件
2. 配置.bash\_profile中有关hadoop的环境变量

vi不要删除原有内容，在文件尾部加入

export HADOOP\_HOME=/home/user1/hadoop-2.7

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

export PATH=$HADOOP\_HOME/sbin/:$PATH

1. 配置**hadoop-env.sh**



图 2‑50 hadoop-env.sh

找到这句，把JAVA\_HOME写进来，如下图：



图 2‑51 hadoop-env.sh（修改后）

1. 配置**yarn-env.sh**

找到如下图位置，

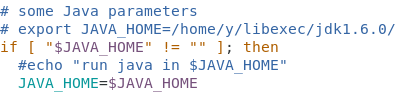


图 2‑52 yarn-env.sh

将JAVA\_HOME变量写在if前面，例如：

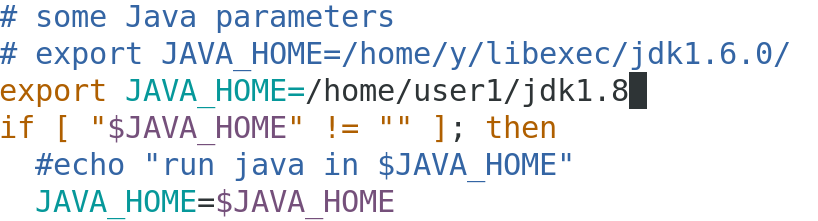


图 2‑53 yarn-env.sh（修改后）

1. 配置核心配置文件**core-site.xml**（默认文件系统和hadoop的工作文件夹）

<property>

<name>fs.defaultFS</name>

<value>hdfs://master:9000</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/home/user1/hadoopdata</value>

</property>

1. 配置**hdfs-site.xml**文件

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>master:9001</value>

</property>

说明：本文件如上的配置不可以用于实际使用.

原因：a.文件副本存储量是1，对文件而言毫无安全性可言;

b.second namenode是作为namenode失效后的后备使用的，当前的配置时把second namenode放到了和namenode相同的物理节点上，这样毫无意义，当namenode失效时，second namenode也一并失效了.

课程上这样配置是因为条件所限和仅仅是为了实验.

1. 配置**yarn-site.xml**文件

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce\_shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>master:8032</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>master:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>master:8035</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>master:8033</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>master:8088</value>

</property>2.5配置mapred-site.xml

先将模板文件转成正常文件：cp -f mapred-site.xml.template mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

1. 配置**slaves**文件

把文件原内容删除，然后添加

slave0

slave1

slave2

1. 拷贝 Hadoop到其他节点

命令示例：

scp -r hadoop-2.7/ slave0:~

scp -r hadoop-2.7 slave1:~

### 格式化HDFS

首先按照配置文件，在namenode上建立文件夹，我们配置的是/home/user1/hadoopdata.

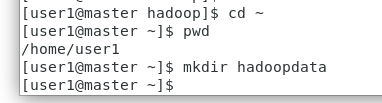


图 2‑54 建立hadoopdata文件夹

然后，在namenode上执行：

hadoop namenode -format

如果看到如下内容，说明格式化成功了：



图 2‑55 格式化成功

格式化不能多次执行，如果不通过需要清空文件夹，才能再次执行格式化.

### 启动集群

启动命令 ：start-all.sh

启动后用jps命令查看java进程，master是4个java进程，slave是3个java进程.

尝试上传一个文件，查看是否正常.

运行命令：hadoop jar hadoop-mapreduce-examples-2.7.7.jar pi 10 10

如果运行正常，说明集群工作一切正常.



图 2‑56 计算结果输出

**注意 ：关闭linux之前，务必先关闭集群（关闭集群命令：stop-all.sh），否则下次启动集群时容易出错.**

## eclipse开发环境配置

下载，解压eclipse，验证是否能够正常运行.

将hadoop-eclipse-plugin-2.6.0.jar拷贝到eclipse的dropins文件夹下.如果已经启动了eclipse，需要关闭重启eclipse.

**注意：应当使用hadoop2.7.7的插件**，但是没有下载到，只是上课时演示过程.

### 配置hadoop路径

在主菜单上选择“window”，如图，然后选择“Preferences”.



图 2‑57 window -> Perferences

在如下界面左边选择“HadoopMap/Reduce”，在右边输入Hadoop的路径（hadoop\_home路径）.输入后点击“Apply and Close”.

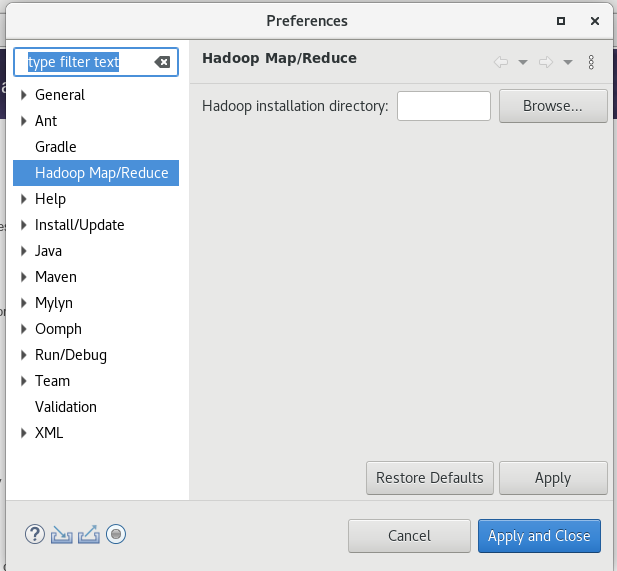


图 2‑58 传入hadoop路径

### 配置Map/Reduce Locations

从主菜单开始，然后选择“Other”.

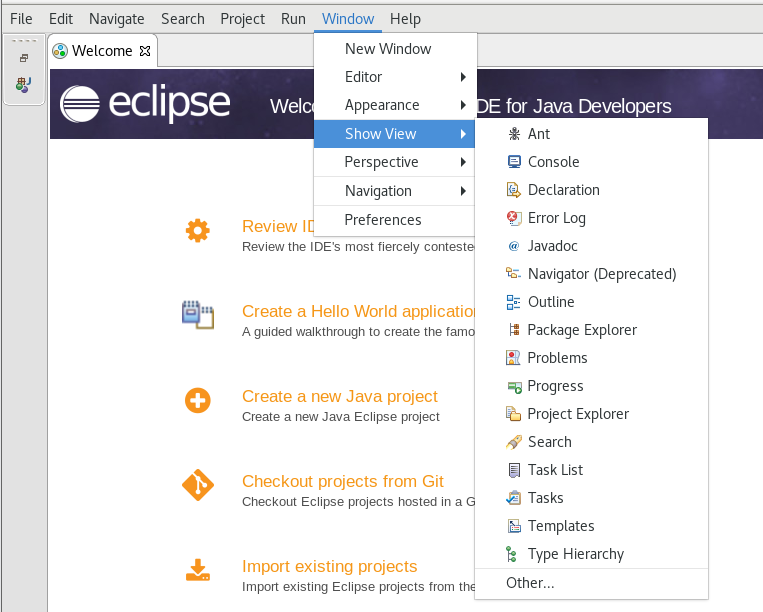


图 2‑59 选择“other”

在以下界面中选择“Map/Reduce Locations”，然后点击”Open”.

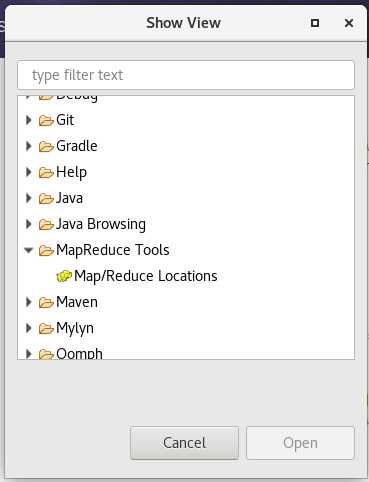


图 2‑60 选择“Map/Reduce Locations”

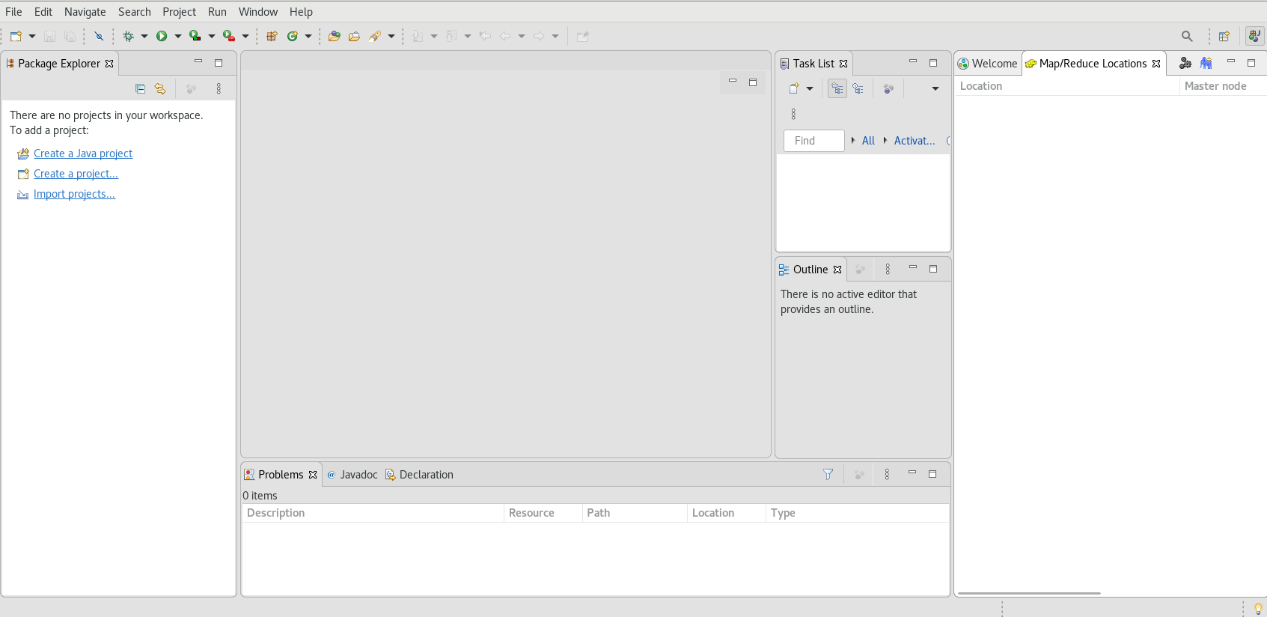


图 2‑61 “Map/Reduce Locations”打开界面

点击界面右侧上方的，新建hadoop location.

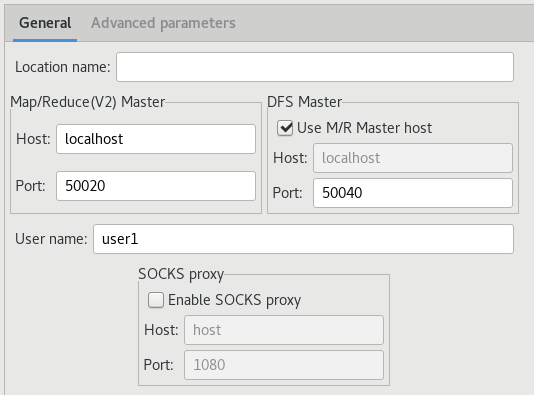


图 2‑62 新建hadoop location

在此页面上，Location name栏输入名字，随便一个字符串都可以，除了系统保留字.

Map/Reduce(V2) Master标签页中，Host要输入master节点的ip地址，我当前master节点ip地址是10.0.0.20；Port要根据配置文件输入.打开mapred-site.xml：

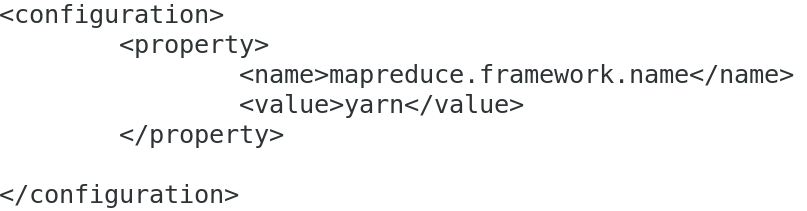


图 2‑63 mapred-site.xml

此文件中没有Map/Reduce的端口配置，只是指定了用yarn进行管理，所以需要打开yarn-site.xml:

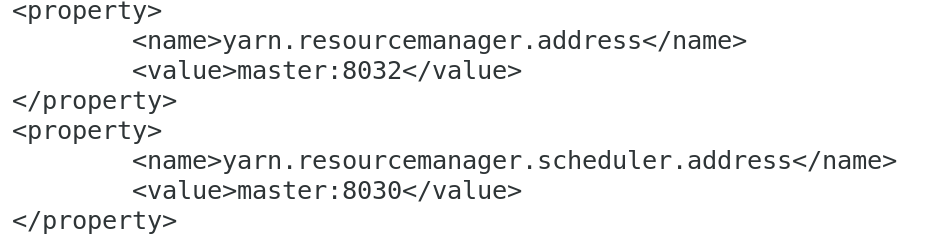


图 2‑64 yarn-site.xml

可填入如上两个端口中的一个.

DFS Master标签页中，端口号也需要根据配置文件填入，打开dfs配置文件hdfs-site.xml，发现文件内容里没有相应的端口号配置，打开core-site.xml

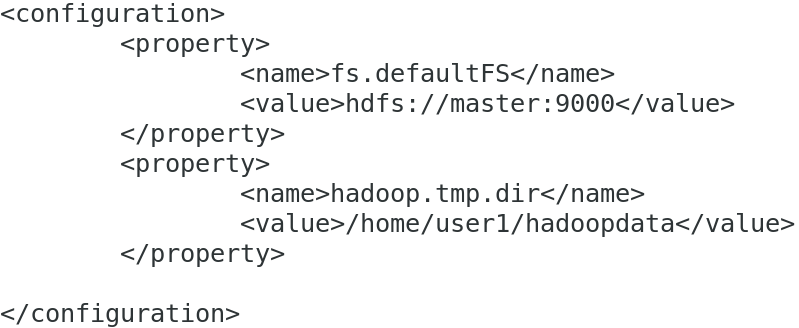


图 2‑65 core-site.xml

可以看到端口号是9000.按照我的集群配置，填好后如下图.点击右下角“Finish”.

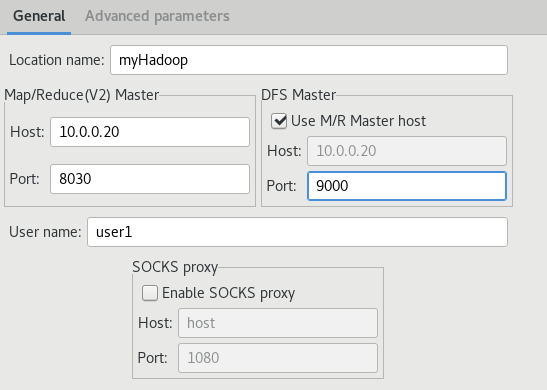


图 2‑66 按照集群配置填写信息

配置完成后，界面上会出现如下图内容：

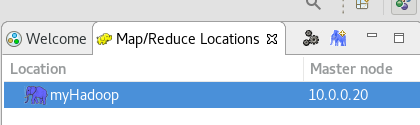


图 2‑67 新的hadoop location

### 建立java的project

从主菜单开始，如下图所示，然后选择“Other”.

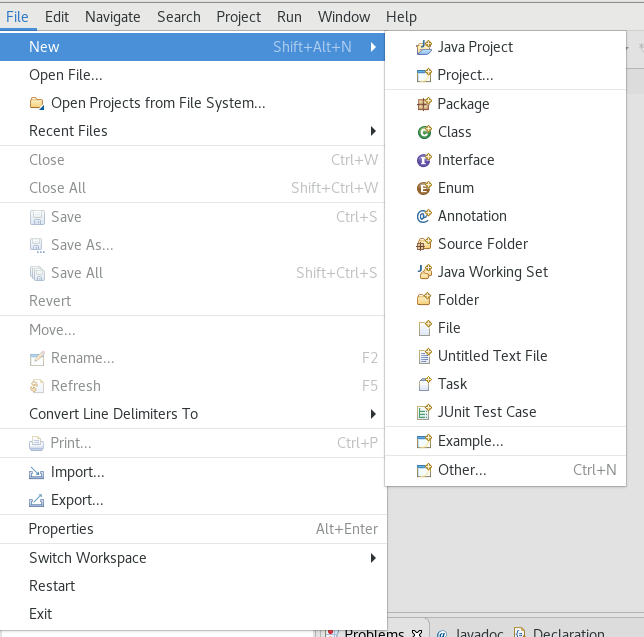


图 2‑68 新建java project

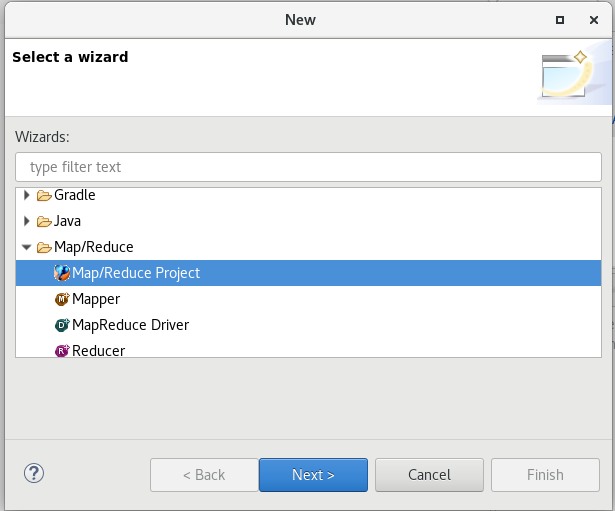


图 2‑69 选择MapReduce Project

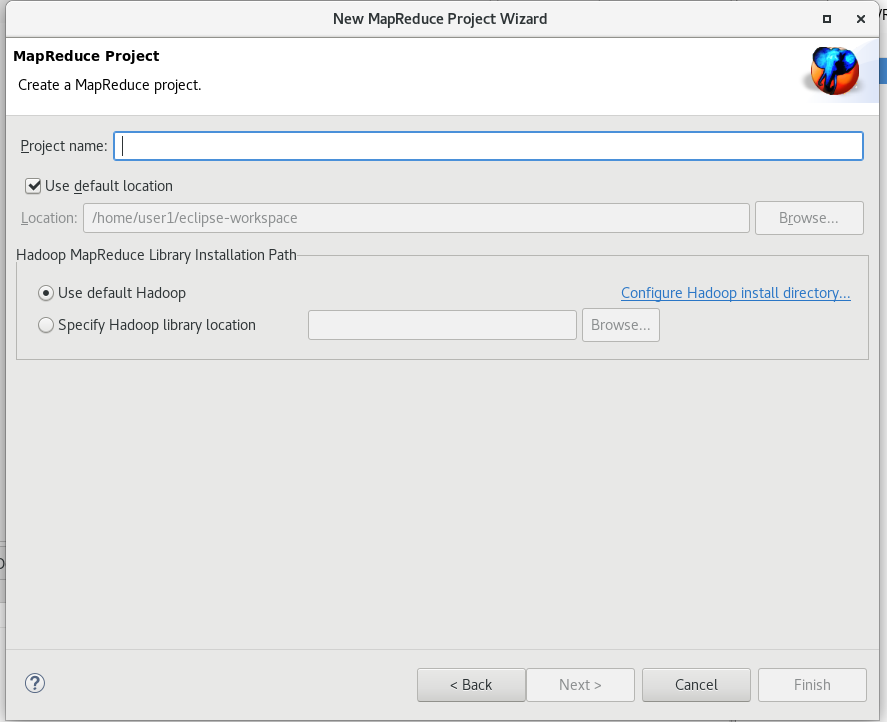


图 2‑70 New MapReduce Project

给自己的project起个名字，然后点击“Finish”.

## Spark安装

### 下载解压spark2.4.5

解压：tar -zxvf spark2.4.5.tgz

在路径：/home/user1/spark-2.4.5/bin下，运行./spark-shell.



图 2‑71启动shell程序

### Spark配置：yarn模式运行

Spark集群需要配置两个文件，spark-env.sh、slaves.

建立spark-env.sh ：cp spark-env.sh.template spark-env.sh

编辑：vi spark-env.sh，添加如下内容，红色部分替换成自己的目录.

export SPARK\_CONF\_DIR=/home/cnsb/spark-2.4.5/conf

export HADOOP\_CONF\_DIR=/home/csnb/hadoop-2.7/etc/hadoop

export YARN\_CONF\_DIR=/home/csnb/hadoop-2.7/etc/hadoop

export JAVA\_HOME=/home/csnb/jdk1.8

建立slaves文件：cp slaves.template slaves

编辑：vi slaves

把文件原内容删除，添加slave节点名：例如

slave0

slave1

slave2

文件编辑后拷贝到集群所有节点.

### 启动spark集群

由于是用yarn模式运行，首先要启动yarn，启动yarn的方式是用hadoop下的start-all.sh.启动顺序是：hadoop的start-all.sh，spark的start-all.sh.

由于两个文件同名，所以必须区分是哪一个.区分的方式可以是，spark路径不配置环境变量，而是用全路径执行.Spark要在master节点上启动.

启动后，用jps查看，**master**节点：

NameNode

Jps

ResourceManager

Master

SecondaryNameNode

**Slave**节点：

DataNode

NodeManager

Worker

Jps

说明启动成功.

## Scala配置

### 下载解压Scala2.11.12

解压：tar -zxvf scala-2.11.12.tgz

进入目录：/home/user1/scala-2.11.12/bin，运行./scala

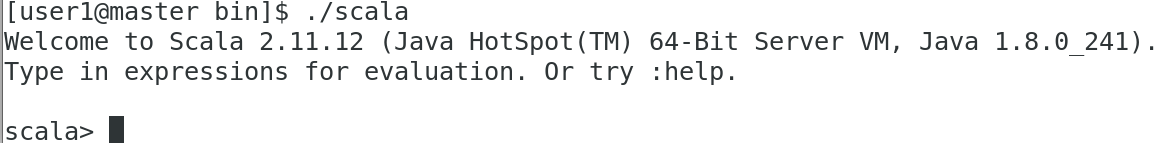


图 2‑72 全路径运行scala

出现scala>说明scala运行正常.

### 配置Scala路径

在.bash\_profile里添加如下内容：

export SCALA\_HOME=/home/user1/scala-2.11.12

export PATH=$SCALA\_HOME/bin/:$PATH

然后运行.bash\_profile：source .bash\_profile

### scala环境退出

输入命令行：“:q”

## Scala集成开发环境idea搭建

### 下载解压idea

输入命令行:tar-zxvf

### 初始化idea

进入bin目录，执行/idea.sh

一路默认，到达：



图 2‑73 启动idea

选择“Create New Project”.按下图选择，然后点击“Next”

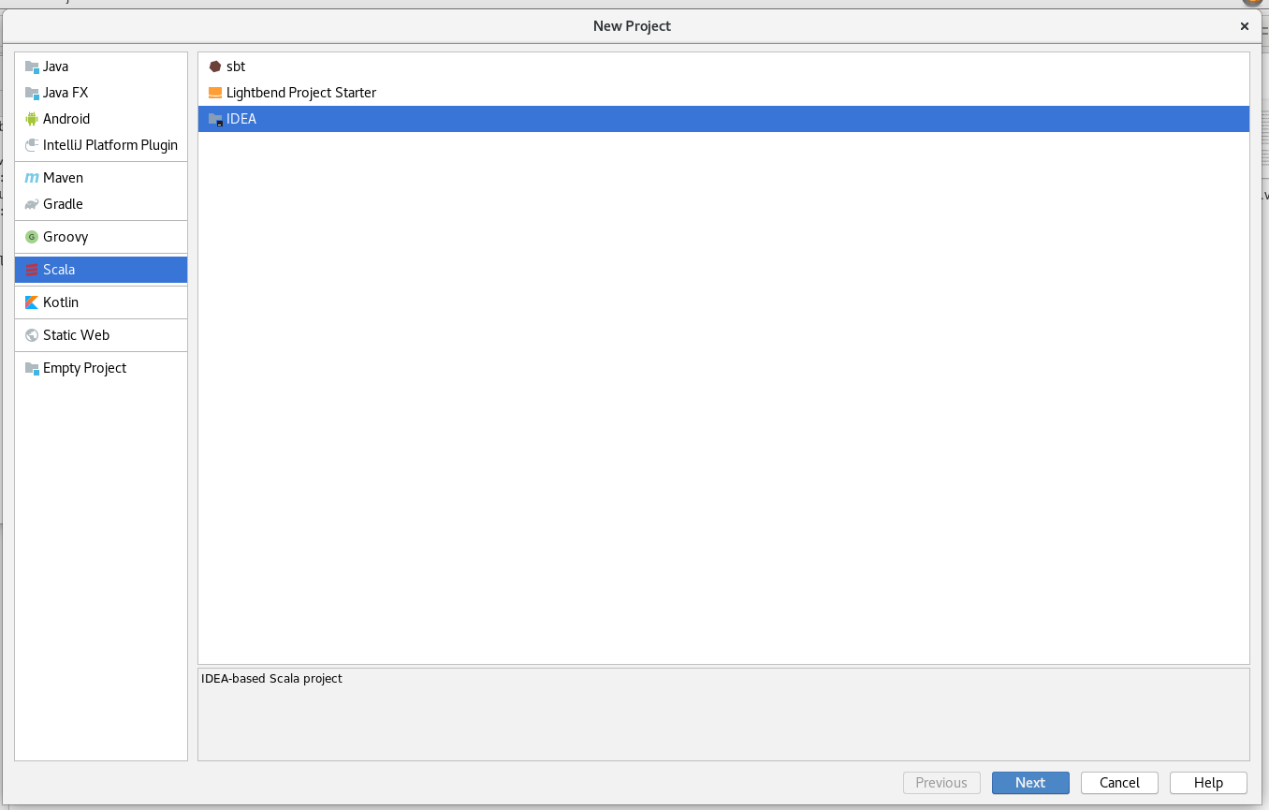


图 2‑74 Create New Project

在“Project name”栏里输入名字，JDK栏要选择jdk1.8，因为scala2.11.12是基于jdk18的.由于我们要在Hadoop框架下运行spark程序，所以需要尽量选择同一个jdk1.8的包.

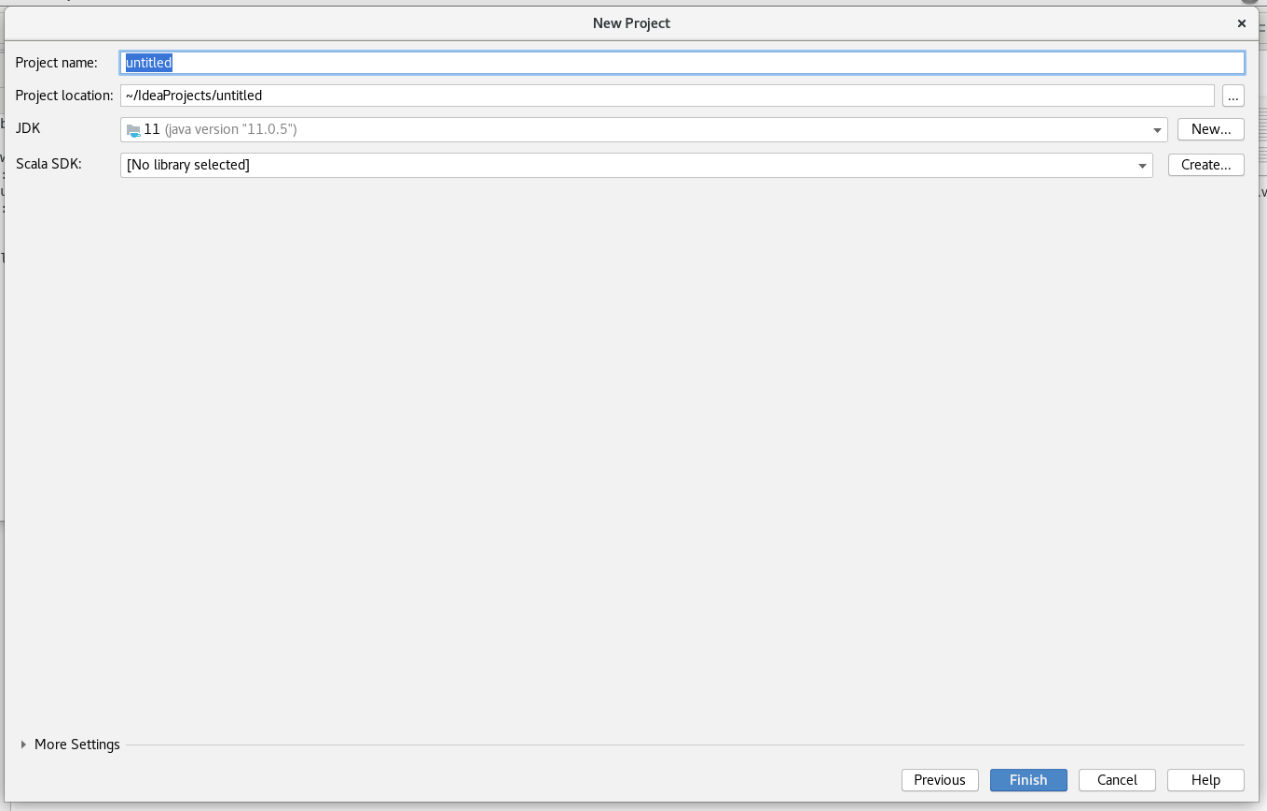


图 2‑75 New Project

点击JDK栏右侧的“New”，选择我们Hadoop集群用的jdk1.8包，点击下方“OK”按钮.

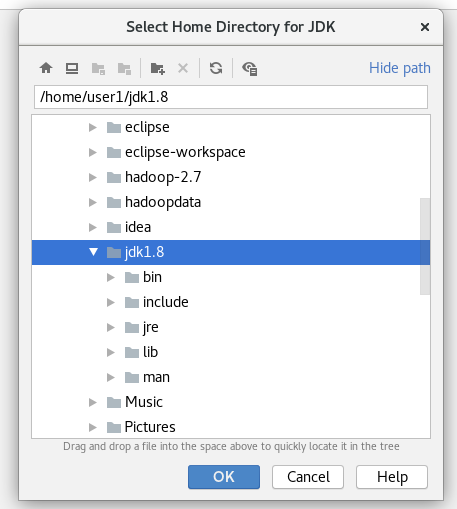


图 2‑76 选择同一个jdk1.8包

点击Scala SDK右侧的“Create”.

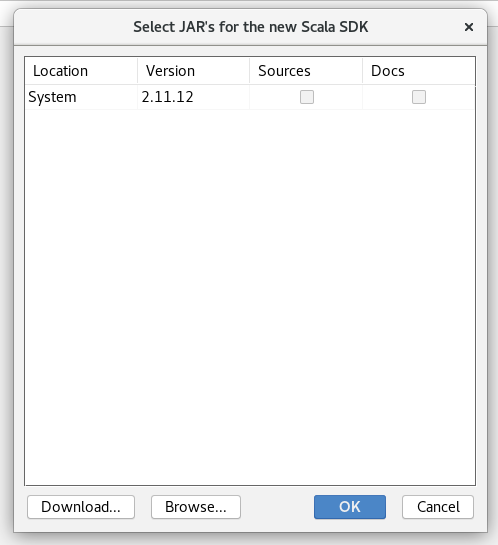


图 2‑77 Select JAR's for the new Scala SDK

如果没有可选项，点击下方的“Browse...”，

选中我们使用的scala文件夹，然后点击下方“OK”按钮.

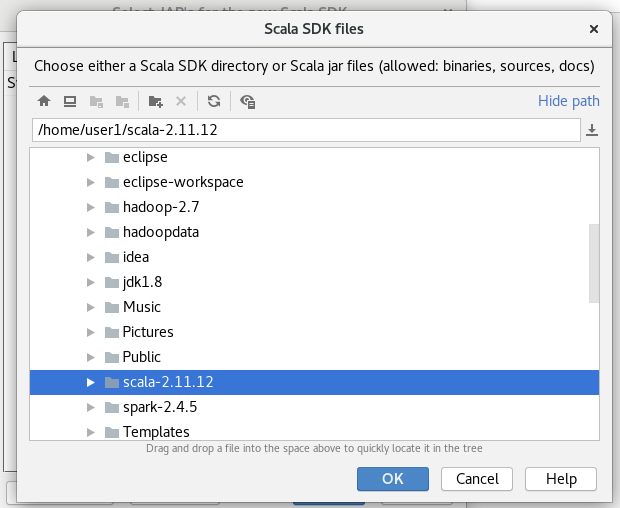


图 2‑78 选中使用的scala文件

可以看的如下正确配置：

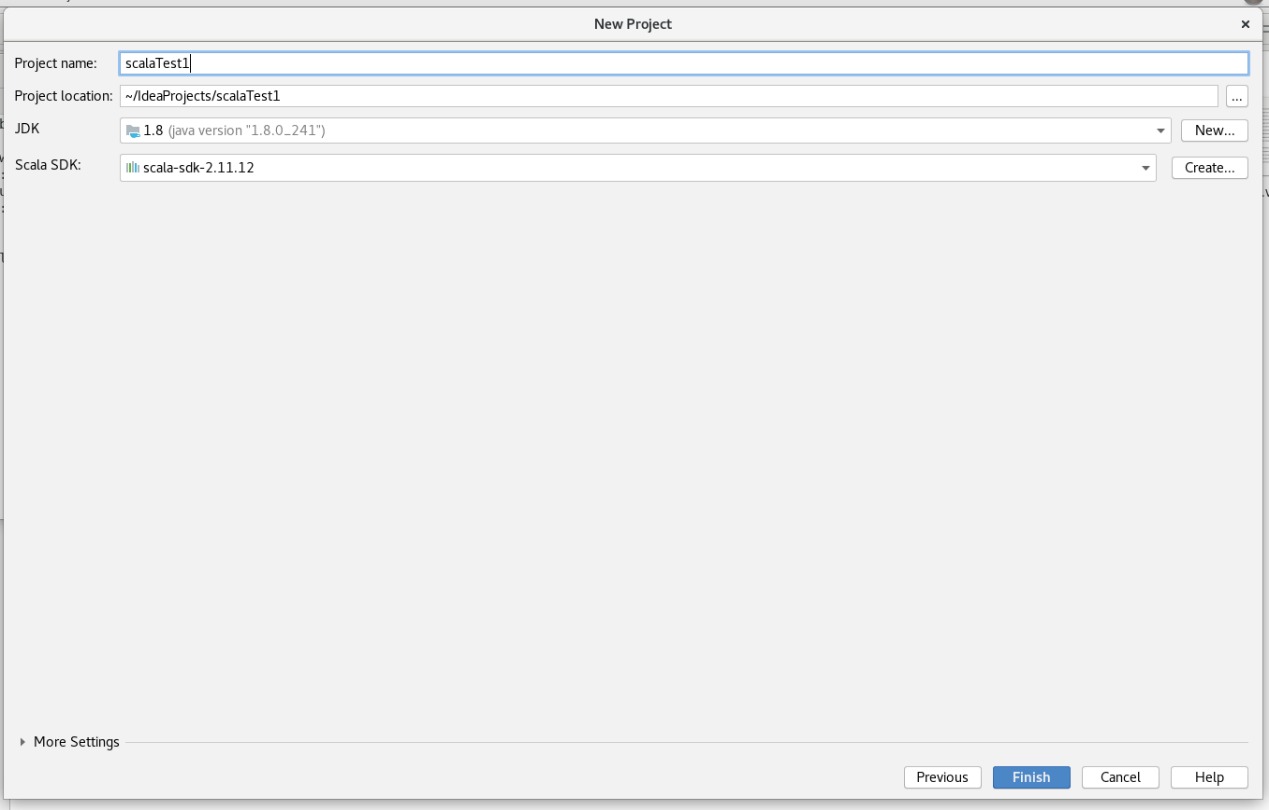


图 2‑79 配置结果

点击下方的“Finish”，就建立了第一个scala project.

初始化后的界面如下：

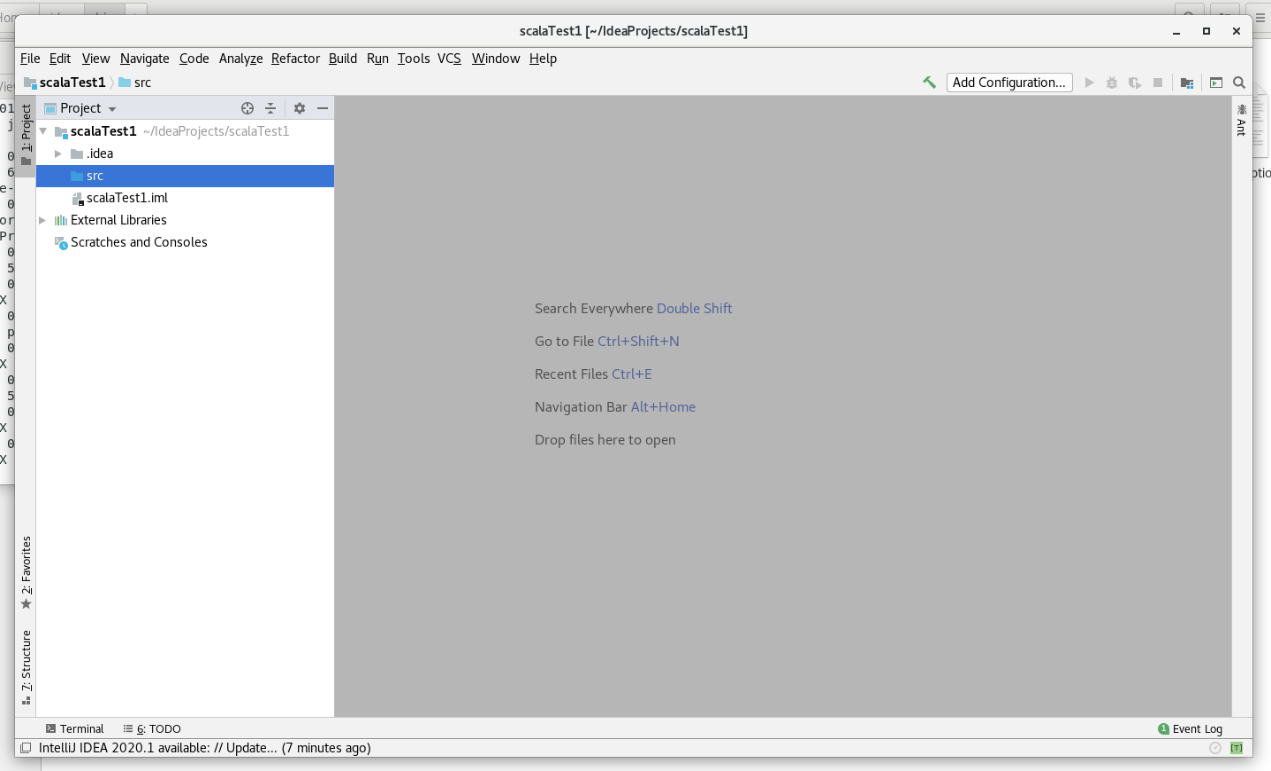


图 2‑80 初始化界面

点击页面左侧的src文件夹

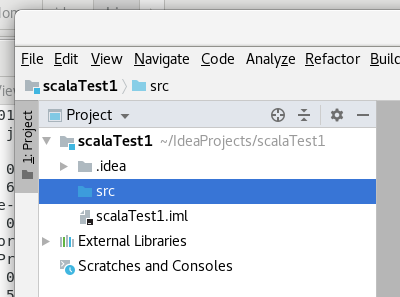


图 2‑81 选中src文件夹

主菜单选择“File”->“New”

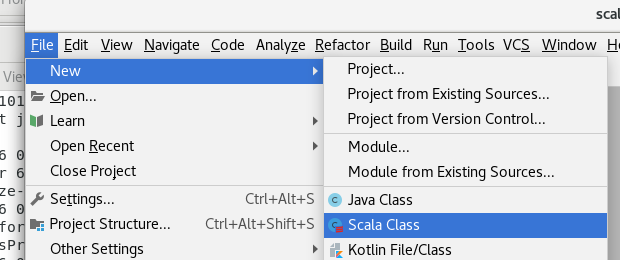


图 2‑82 新建Scala Class

选择**Object**输入名字后回车

输入一段程序，

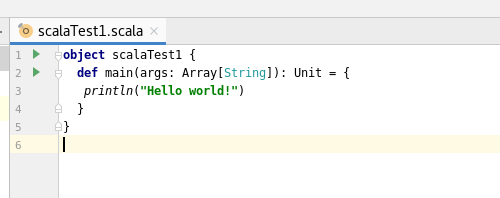


图 2‑83 程序代码

在程序上右键鼠标选择run，可以看到执行结果.



图 2‑84 执行结果

## 修改Hadoop有关配置

首先执行命令：hadoop-daemon.sh start datanode

利用yarn-utils.py计算，将计算机过复制到yarn.site.xml文件中，如图2‑85所示：

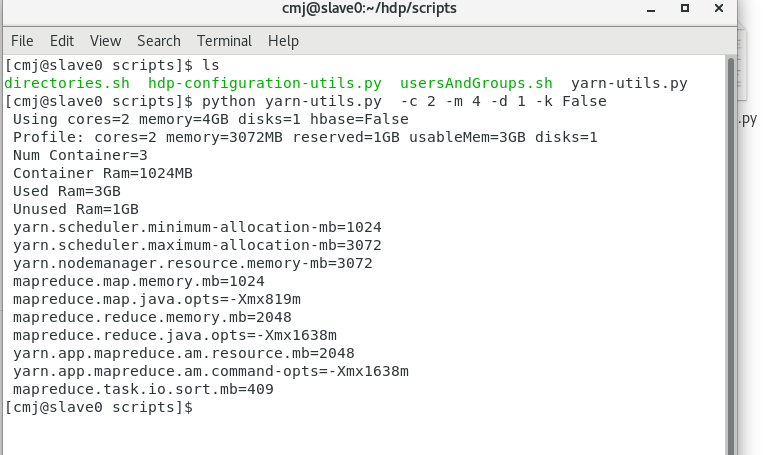


图 2‑86 yarn-utils.py计算结果

将yarn-utils.py的计算结果写到文件中：

python yarn-utils.py -c 2 -m 4 -d l -k False >yarnset

将这些yarnset文件中的内容复制到hadoop中的yarn.site-xml中，并添加标签信息.



图 2‑87 添加标签信息之后的yarn.site-xml

再在最后两行添加俩行内容.

yarn.nodemanager.vmem-pmem-ratio ：任务每使用1MB物理内存，最多可使用虚拟内存量，默认是2.1

yarn.nodemanager.resource.cpu-vcores ：表示该节点上YARN可使用的虚拟CPU个数，默认是8，注意，目前推荐将该值设值为与物理CPU核数数目相同.如果节点的CPU核数不够8个，则需要调减小这个值，因为YARN不会智能地探测节点的物理CPU总数.

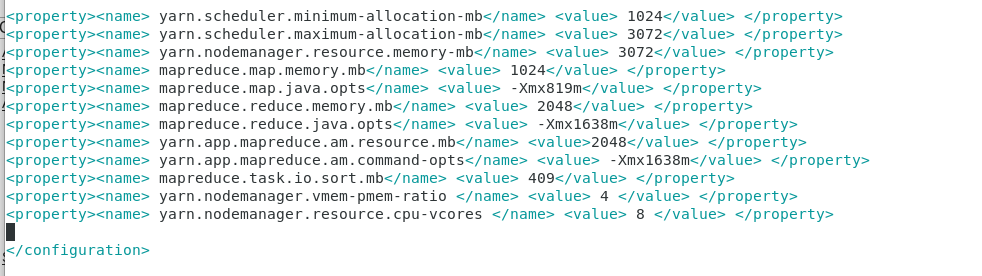


图 2‑88 修改之后的yarn.site-xml

Sparkpi运行指令：

~/spark-2.4.5/bin/spark-submit --class org.apache.spark.examples.SparkPi --master yarn-cluster ~/spark-2.4.5/examples/jars/spark-examples\_2.11-2.4.5.jar

运行结果和hadoop计算结果分别如下图 2‑89和图2‑90所示.

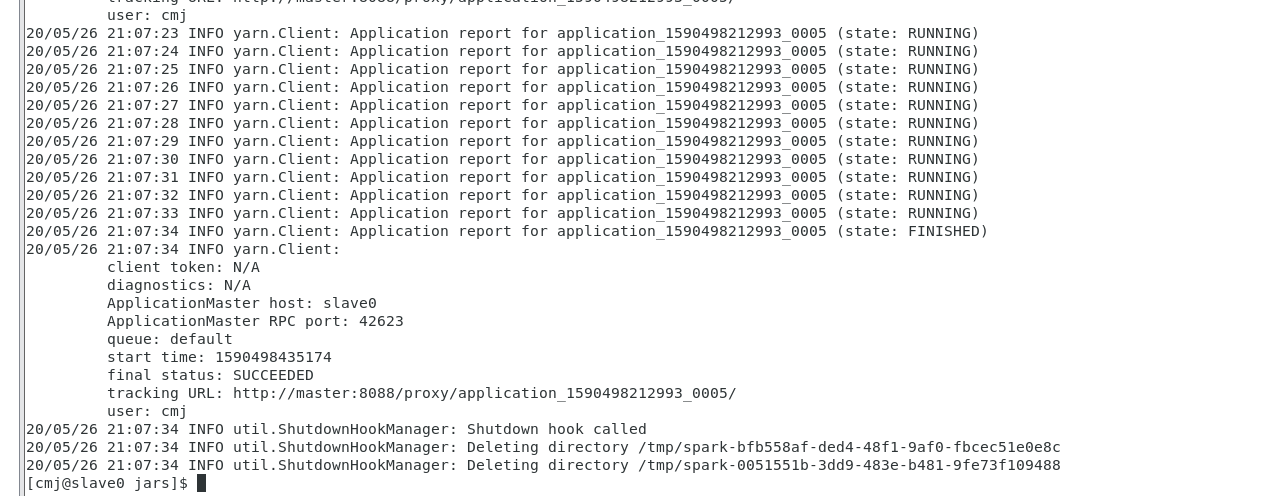


图 2‑91 运行结果



图 2‑92 hadoop计算结果

# 大数据开发重要概念

## GPL 自由软件协议

GPL自由软件协议有效地保护了自由软件不受商业软件的非法侵犯.计算机软件作为人类的知识财富，为人类社会的发展起到了巨大的作用，但长期以来软件源码作为个人或公司的私有财产受到严格的保密，很难做到像文学艺术作品一样地进行公开的交流，很大程度上造成软件的低水平，重复劳动严重，在一定意义上制约了软件的发展.在 GPL 下人们就可以自由交流、修改软件源码，极大地推动了整个计算机软件行业的发展.

GPL 协议的核心就是要对源码进行公开，并且允许任何人修改源码，但是只要使用了 GPL 协议的软件源码，其衍生软件也必须公开源码，准许其他人阅读和修改源码，即 GPL 协议具有继承性.

另外，适应GPL 协议的软件并非就是免费软件，这里所说的自由软件是指对软件源码的自由获得与自由使用、修改，软件开发者不但可以通过服务来收费，而且还可以通过出售 GPL 软件来获利.

## Vi编辑器

Vi编辑器是Unix/Linux操作系统中最经典的文本编辑器，只能是编辑字符，不能对字体、段落进行排版;既可以新建文件，也可以编辑文件；没有菜单，只有命令，且命令繁多，运行于字符界面，经常被使用.

Vi编辑器主要有三种命令模式：Command（命令）模式、Insert（插入）模式和Visual（可视）模式.

* Command（命令）模式，用于输入命令.
* Insert（插入）模式，用于插入文本.
* Visual（可视）模式，用于视化的的高亮并选定正文.

Vi编辑器有三种工作模式：一般模式、文本编辑模式和命令行模式.

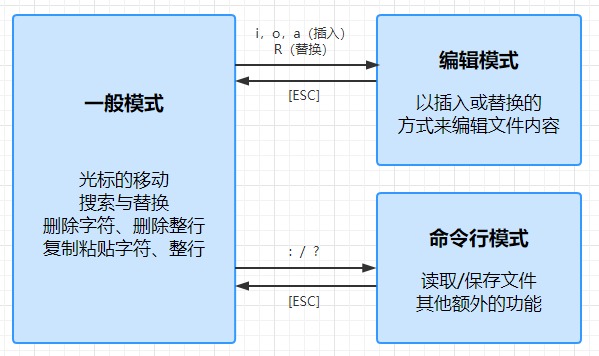


图 3‑1 Vi编辑器的三种工作模式

* 命令模式：命令模式是启动vi后进入的工作模式，并可转换为文本编辑模式和最后行模式.在命令模式下，从键盘上输入的任何字符都被当作编辑命令来解释，而不会在屏幕上显示.如果输入的字符是合法的vi命令，则vi就会完成相应的动作;否则vi会响铃警告.
* 文本编辑模式：文本编辑模式用于字符编辑.在命令模式下输入i（插入命令）、a（附加命令）等命令后进入文本编辑模式，此时输入的任何字符都被vi当作文件内容显示在屏幕上.按Esc键可从文本编辑模式返回到命令模式.
* 一般模式：在命令模式下，按“：”键进入最后行模式，此时vi会在屏幕的底部显示“：”符号年作为一般模式的提示符，等待用户输入相关命令.命令执行完毕后，vi自动回到命令模式.

## 大数据

大数据又称为巨量资源、 巨量数据或海量数据. 一般来说，大数据的特性可归类为 3V: Volume（大量数据）、 Variety（多样性） 和 Velocity（时效性）.

* Volume：大数据的数据量等级极高，因特网、 企业 IT、物联网、 社区、 短信、 电话、 网络搜索、 在线交易等，随时都在快速累积庞大的数据，导致数据量很容易达到 TB(Terabyte， 1024 GB) 甚至 PB(Petabyte， 1024 TB) 或 EB (Exabyte ， l 024 PB) 的等级.
* Variety：大数据的数据类型非常多样化，可分为非结构化信息和结构化信息.非结构化信息包括文字、 图片、 图像、 视频、 音乐、 地理位置信息、 个人化信息——­如社区、 交友数据等.结构化信息指的是数据库、 数据仓库等.
* Velocity：对于大数据来说，时间太久就会失去数据的价值，所以数据必须要能够在最短时间内得出分析结果.

## Hadoop平台

Hadoop是存储与处理大量数据的平台，是Apache软件基金会的开放源码、免费且广泛使用的软件.

Hadoop的发展历史：

* 2002年Doug Cutting与Mike Cafarelia开始进行Nutch项目.
* 2003年Google发表GFS (Google File System)与MapReduce论文.
* 2004年Doug Cutting开始将DFS与MapReduce加入Nutch项目.
* 2006年Doug Cutting加入Yahoo团队，并将Nutch改名为Hadoop.
* 2008年Yahoo使用Hadoop包含了910个集群，对1TB的数据排序花了297秒.

## Hadoop HDFS分布式文件系统

HDFS采用分布式文件系统(Hadoop Distributed File System) ， 可以由单台服务器扩充到数于台服务器.

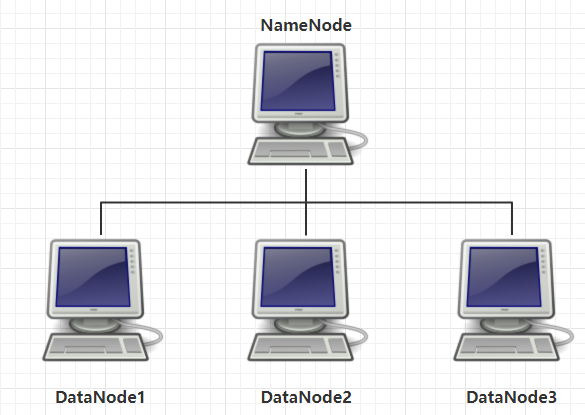


图 3‑2 HDFS分布式文件系统

**NameNode** 服务器负责管理与维护 HDFS 目录系统并控制文件的读写操作；多个 **DataNode** 服务器主要负责存储数据.

HDFS设计的前提与目标：

* 硬件故障是常态而不是异常.HDFS 是设计运行在低成本的普通服务器上的，所以 HDFS 被设计成具有高度容错能力、能够实时检测错误并且自动恢复，这是 HDFS 核心的设计目标.
* Streaming 流式数据存取.运行在 HDFS 上的应用程序会通过 Streaming 存取数据集. HDFS 的主要设计是批处理，而不是实时互动处理，优点是可以提高存取大量数据的能力，但是牺牲了响应时间.
* 存储大数据集. HDFS 提供了 cluster 集群架构，用于存储大数据文件，集群可扩充至数百个节点.
* 简单一致性模型.HDFS 的存取模式是一次写入多次读取 ， 一个文件被创建后就不会再修改. 这样设计的优点是： 可以提高存储大量数据的能力，并简化数据一致性的问题.
* 移动“计算”比移动“数据”成本更低.cluster 集群存储了大量的数据，要搬移数据必须耗费大量的时间成本. 因此如果有“计算”数据的需求时，就会将 “计算功能” 在接近数据的服务器中运行，而不是搬移数据.
* 跨硬件与软件平台.利于Hadoop的推广.

## Hadoop MapReduce计算方法

利用大数据进行数据分析处理时数据量庞大，所需的运算量也巨大.Hadoop MapReduce 的做法是采用分布式计算的技术：

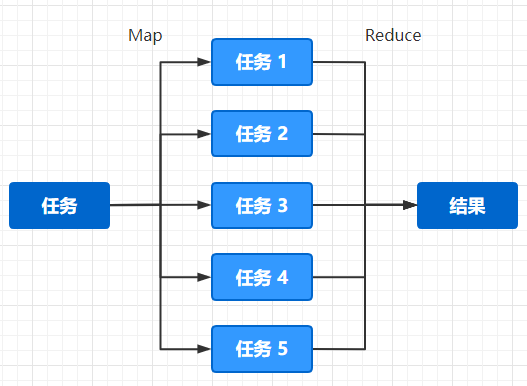


图 3‑3 MapReduce 分布式计算方法

**Map** 方法将任务分割成更小任务，由每台 服务器分别运行；**Reduce** 方法将所有服务器的运算结果汇总整理，返回最后的结果.通过 MapReduce 的计算方法，Hadoop可以在上千台机器上并行处理巨量的数据，大大减少数据处理的时间.

Hadoop 的MapReduce 架构称为**YARN** (Yet Another Resource Negotiator），是效率更高的资源管理核心.

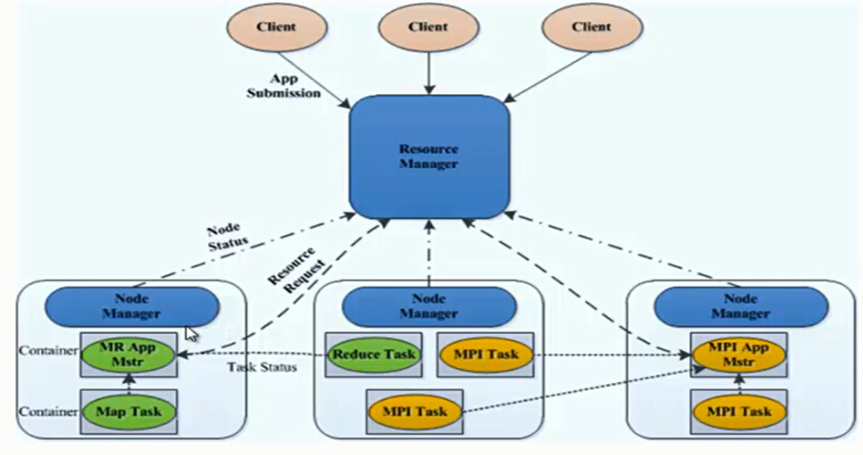


图 3‑4 YARN架构图

在**Client 客户端**，用户会向Resource Manager 请求执行运算（或执行任务）；在**NameNode** 会有Resource Manager 统筹管理运算的请求；在其他的**DataNode**会有NodeManager负责运行，以及监督每一个任务(task)， 并且向ResourceManager汇报状态.

## Spark数据处理引擎

Spark 是开放源码的集群运算框架.Spark基于内存计算，提高了在大数据环境下数据处理的的实时性，同时保证了高容错性和高可伸缩性.

Spark框架的特点：

* 命令周期短.Spark 是基于内存计算的开放源码集群运算系统， 比原先的Hadoop MapReduce 快100 倍.
* 易于开发程序.Spark 支持多种语言： Scala、Python、Java， 也就是说开发者可以根据应用的环境来决定使用哪一种语言开发Spark 程序， 更具弹性， 更符合开发时的需求.
* 与Hadoop 兼容.Spark 支持Hadoop 的HDFS 存储系统， 并且支持Hadoop YARN， 可共享存储资源与运算.

## Spark RDD计算框架

RDD（Resilient Distributed Dataset），是一个不可变的对象集合，在集群的不同节点上进行计算，可以理解为是一个数据分布在各个不同节点上的数据集合，当对它进行操作计算时，实际上是对各个节点上的数据进行计算.

### RDD计算类型

在 RDD 之上， 可以施加 3 种类型的运算：Transformation、Action和Persistence.RDD通过“Transformation”运算可以得出新的RDD，但Spark会延迟这个“Transformation”动作的发生时间点.它并不会马上执行，而是等到执行了“Action”动作之后才会基于所有的RDD关系来执行转换.

|  |  |
| --- | --- |
| RDD运算类型 | 说明 |
| **Transformation** | * RDD执行 “Transformation” 运算的结果，会产生另外一个RDD * RDD具有惰性机制，所以 “Transformation” 运算并不会立刻实际执行， 等到执行 “ Action” 运算才会实际执行 |
| **Action** | * RDD执行 “Action” 运算后不会产生另外一个RDD， 而是会产生数值、数组或写入文件系统 * RDD执行 “Action” 运算时会立刻实际执行， 并且连同之前的 “Transformation” 运算一起执行 |
| **Persistence** | * 对于那些会重复使用的RDD， 可以将RDD"待久化 ” 在内存中作为后续使用， 以提高执行性能 |

### RDD计算框架的特点

* 内存计算：它将中间计算结果存储在分布式内存(RAM)中，而不是磁盘中.
* 延迟计算： Spark中的所有transformation都是惰性的，因为它们不会立即计算结果，它们会记住应用于数据集的那些transformation.直到action出现时，才会真正开始计算.
* 容错性：Spark RDDs能够容错，因为它们跟踪数据沿袭信息，以便在故障时自动重建丢失的数据.
* 不可变性：你可以通过对RDD计算得到新的RDD，但是无法改变现有RDD内的数据.跨进程共享数据是安全的.它也可以在任何时候创建或检索，这使得缓存、共享和复制变得容易.因此，它是一种在计算中达到一致性的方法.
* 分区性：partition是Spark RDD中并行性的基本单元，每个分区都是数据的逻辑分区.Partition—task一一对应
* 持久化：用户可以声明他们将重用哪些RDDs，并为它们选择存储策略.
* 数据本地性：RDDs能够定义计算分区的位置首选项.位置首选项是关于RDD位置的信息.

### RDD分区

RDD是弹性分布式数据集，通常RDD很大，会被分成很多个分区，分别保存在不同的节点上.RDD分区的作用：

* 增加并行度
* 减少通信开销

RDD分区的一个原则是使得分区的个数尽量等于集群中的CPU核心（core）数目  
对于不同的Spark部署模式而言（本地模式、Standalone模式、YARN模式、Mesos模式），都可以通过设置spark.default.parallelism这个参数的值，来配置默认的分区数目

创建RDD时手动指定分区个数.在调用textFile()和parallelize()方法的时候手动指定分区个数即可，语法格式如下：sc.textFile(path， partitionNum)其中，path参数用于指定要加载的文件的地址，partitionNum参数用于指定分区个数.

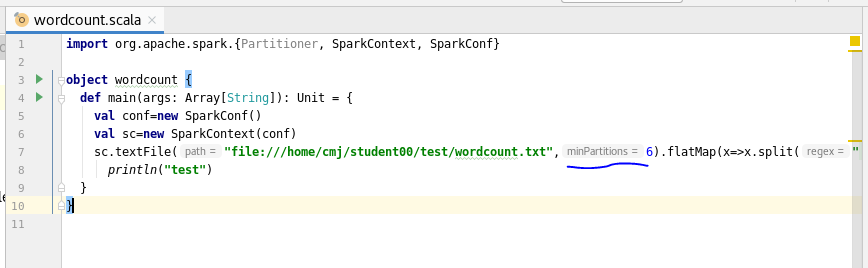


图 3‑5 RDD手动指定分区个数

## Scala语言

Scala 是一门多范式（multi-paradigm）的编程语言，设计初衷是要集成面向对象编程和函数式编程的各种特性.Scala 语言可以直接运行在Java虚拟机上，并兼容现有的Java程序.Scala 语言具有如下特点：

* 面向对象特性.Scala是一种纯面向对象的语言，每个值都是对象，对象的数据类型以及行为由类和特质描述.
* 函数式编程.Scala也是一种函数式语言，其函数也能当成值来使用.Scala提供了轻量级的语法用以定义匿名函数，支持高阶函数，允许嵌套多层函数.

### 值和变量

在Scala语言中，**val表示值，var表示变量**.值在声明后不能修改，变量可以.



图 3‑6 核心数值类型

在计算过程，低级数值类型会自动向高级数值类型转换，但不会自动从高级数据类型向低级数据类型转换.

类型转换可以强制执行，使用toType方法.



图 3‑7 数值字面量

字面量中的字母不区分大小写.

### 字符串String

在Scala语言中，String类型需要用双引号.String类型比数值类型级别高，所以计算过程中数值类型会转换为String类型，进行String类型的计算（拼接）.

### Scala类型概述

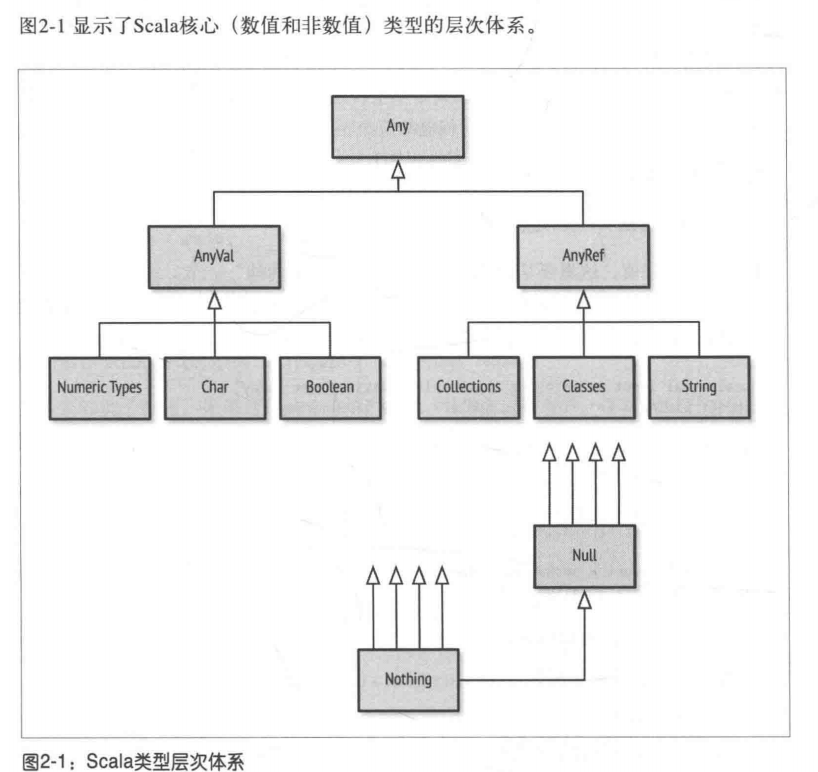


图 3‑8 Scala类型层次体系

### Scala if...else语句

scala只有if else形式，没有if、else if、else形式.

同时scala语言没有问号表达式，而要使用if else来代替.

### Scala 循环

* 匹配模式

类似其他语言的case语句.

1. 通配模式匹配：使用“other”，或者“\_”，当所有选项都没有选中时执行这个语句.

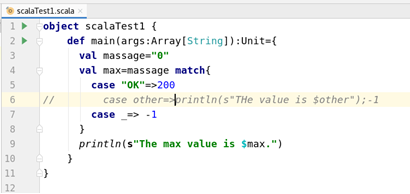


图 3‑9通配模式匹配

b.模式哨位匹配：模表达式中使用if表达式.

* for 循环

a.没有yield的循环

object scalaTest1 {

def main(args: Array[String]): Unit = {

val res=for(x<-1 to 7)println(s"Day $x.")

println(s"The max value is $res.")}

}

b.有yield的for循环

object scalaTest1 {

def main(args: Array[String]): Unit = {

val res=for(x<-1 to 7) yield{ println(s"Day $x.");x}

println(s"The max value is $res.")}

}

返回值是一个Vector

c.迭代器哨位

就是在循环中出现if语句

object scalaTest1 {

def main(args: Array[String]): Unit = {

val res=for(i<-1 to 20 if i%3==0) yield{ println(s"Day $i.");i}

println(s"The max value is $res.")

}

}

d.for循环中的值绑定

第一段程序

object scalaTest1 {

def main(args: Array[String]): Unit = {

val res= for (i<- 0 to 8) yield {var pow=1<<i;pow}

println(res)

}

}

第二段程序

object scalaTest1 {

def main(args: Array[String]): Unit = {

val res= for (i<- 0 to 8; pow=1<<i;) yield pow

println(res)

}

}

俩段程序运行结果相同.

* while循环

a. while

object scalaTest1 {

def main(args: Array[String]): Unit = {

var x=10

val res= while(x>0) {println(s"$x.");x-=2;x}

println(res)

}

}

b. do-while

object scalaTest1 {

def main(args: Array[String]): Unit = {

var x=10

val res=do{x-=2;println(s"$x.")} while(x>0)

println(res)

}

}

### Scala 方法与函数

Scala 有方法与函数，二者在语义上的区别很小.Scala 方法是类的一部分，而函数是一个对象可以赋值给一个变量.Scala 中的方法跟 Java 的类似，方法是组成类的一部分；Scala 中的函数则是一个完整的对象，Scala 中的函数其实就是继承了 Trait 的类的对象.

Scala 中使用 **val** 语句可以定义函数，**def** 语句定义方法.

* 无参数函数

a.object scalaTest1 {

def main(args: Array[String]): Unit = {

def fHi="hi"

var res=fHi

println(res)

}

}

b.object scalaTest1 {

def main(args: Array[String]): Unit = {

def fHi()="hi"

var res=fHi()

println(res)

}

}

* 指定返回类型的函数

用冒号指定返回类型.例如def fHi():String="hi"

* 指定参数类型和返回类型的函数

a.乘法

object scalaTest1 {

def main(args: Array[String]): Unit = {

def mul(x:Int，y:Int):Int=x\*y

var res=mul(3，5)

println(res)

}

}

b. 除法

object scalaTest1 {

def main(args: Array[String]): Unit = {

def div(x:Int，y:Int):Float={var tmp:Float=x;tmp/y}

var res=div(3，5)

println(res)

}

}

* 递归函数

递归函数可以提高代码的利用率，但有栈溢出危险，所以需要仔细处理终点条件

* 调用函数使用命名参数

调用函数时，参数的传递采用“形参=实参”的形式，好处是参数传递不用按照形参顺序

* 有默认参数值的函数

函数定义时可以给形参一个值，作为默认参数，有默认值的要放到形参表的尾部.

特殊情况：如果有默认值的形参不在形参表的尾部，函数调用时如果使用默认值，就需要使用命名参数调用形式.

* vararg参数

就是可以输入个数可变的参数.定义方法是在形参类型后加“\*”

# 大数据开发常用方法介绍和分析

## Linux系统远程使用

一般服务器系统都不安装GUI，因为没有必要，而且GUI非常占用系统资源.一般情况下，Server系统都是远程使用.如果桌面系统是linux或mac，可以直接ssh远程登陆，如果是windows，由于windows一般版本没有相应组件，需要使用第三方软件，例如：putty，xshell.

以putty为例介绍.修改字体大小，是在window下的Appearance里.

连接远程主机时需要在Session页面上的Host Name(or IP address)栏目里输入主机名或ip地址，如果输入主机名，必须先解决域名解析问题.

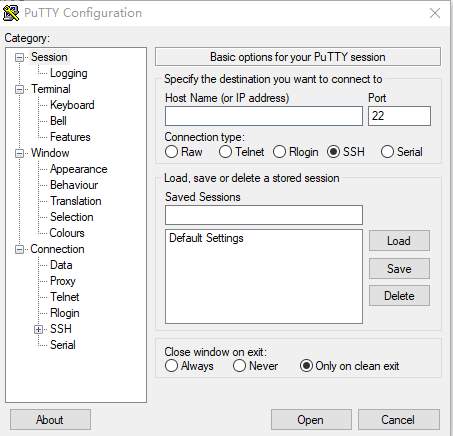


图 4‑1 putty主页面

## 从非集群主机使用集群

从非集群主机访问集群，首先需要网络畅通.还需要有命令解释器，其次要知道集群配置情况，最简单的办法就是从集群中拷贝一个hadoop软件包，还需要一个同版本的java.

一般情况下，java路径与集群中的不同，需要修改，修改

vi hadoop-env.sh

vi yarn-env.sh

注意用户名问题，如果与集群用户不同，需要改成相同的.

## 分析WordCount.java

WordCount.java程序源代码：

1. /\*\*
2. \* Licensed to the Apache Software Foundation (ASF) under one
3. \* or more contributor license agreements.  See the NOTICE file
4. \* distributed with this work for additional information
5. \* regarding copyright ownership.  The ASF licenses this file
6. \* to you under the Apache License， Version 2.0 (the
7. \* "License"); you may not use this file except in compliance
8. \* with the License.  You may obtain a copy of the License at
9. \*
10. \*     http://www.apache.org/licenses/LICENSE-2.0
11. \*
12. \* Unless required by applicable law or agreed to in writing， software
13. \* distributed under the License is distributed on an "AS IS" BASIS，
14. \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND， either express or implied.
15. \* See the License for the specific language governing permissions and
16. \* limitations under the License.
17. \*/
18. **package** com;
19. **import** org.apache.hadoop.fs.FileSystem;
20. **import** org.apache.hadoop.conf.Configuration;
21. **import** org.apache.hadoop.fs.Path;
22. **import** org.apache.hadoop.io.IntWritable;
23. **import** org.apache.hadoop.io.Text;
24. **import** org.apache.hadoop.mapreduce.Job;
25. **import** org.apache.hadoop.mapreduce.Mapper;
26. **import** org.apache.hadoop.mapreduce.Reducer;
27. **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
28. **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
29. **import** org.apache.hadoop.util.GenericOptionsParser;
31. **import** java.io.IOException;
32. **import** java.util.StringTokenizer;
34. **public** **class** WordCount {
36. **public** **static** **class** TokenizerMapper
37. **extends** Mapper<Object， Text， Text， IntWritable>{
39. **private** **final** **static** IntWritable one = **new** IntWritable(1);
40. **private** Text word = **new** Text();
42. **public** **void** map(Object key， Text value， Context context
43. ) **throws** IOException， InterruptedException {
44. StringTokenizer itr = **new** StringTokenizer(value.toString());
45. **while** (itr.hasMoreTokens()) {
46. word.set(itr.nextToken());
47. context.write(word， one);
48. }
49. }
50. }
52. **public** **static** **class** IntSumReducer
53. **extends** Reducer<Text，IntWritable，Text，IntWritable> {
54. **private** IntWritable result = **new** IntWritable();
56. **public** **void** reduce(Text key， Iterable<IntWritable> values，
57. Context context
58. ) **throws** IOException， InterruptedException {
59. **int** sum = 0;
60. **for** (IntWritable val : values) {
61. sum += val.get();
62. }
63. result.set(sum);
64. context.write(key， result);
65. }
66. }
68. **public** **static** **void** main(String[] args) **throws** Exception {
69. Configuration conf = **new** Configuration();
70. String[] otherArgs = **new** GenericOptionsParser(conf， args).getRemainingArgs();
71. **if** (otherArgs.length < 2) {
72. System.err.println("Usage: wordcount <in> [<in>...] <out>");
73. System.exit(2);
74. }
75. Job job = Job.getInstance(conf， "word count");
76. job.setJarByClass(WordCount.**class**);
77. job.setMapperClass(TokenizerMapper.**class**);
78. job.setCombinerClass(IntSumReducer.**class**);
79. job.setReducerClass(IntSumReducer.**class**);
80. job.setOutputKeyClass(Text.**class**);
81. job.setOutputValueClass(IntWritable.**class**);
82. **for** (**int** i = 0; i < otherArgs.length - 1; ++i) {
83. FileInputFormat.addInputPath(job， **new** Path(otherArgs[i]));
84. }
85. FileOutputFormat.setOutputPath(job，
86. **new** Path(otherArgs[otherArgs.length - 1]));
87. System.exit(job.waitForCompletion(**true**) ? 0 : 1);
88. }
89. }

WordCount执行流程: input 🡪 map 🡪 reduce 🡪 output.

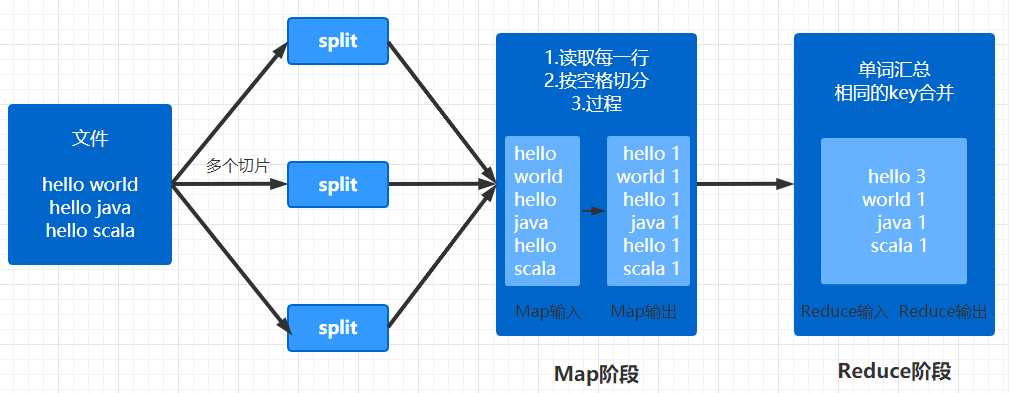


图 4‑2 WordCount执行流程图

* Input: 将文件按行分割形成<key，value>对，如图 4‑3所示:

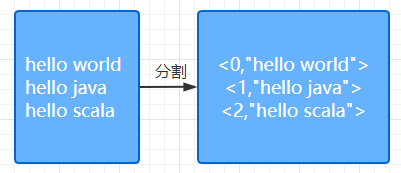


图 4‑4 将文件按行分割

* Map：将分割好的<key，value>对交给用户定义的map方法处理，形成新的<key，value>对，得到map方法输出的<key，value>对后，Mapper会将他们按照key值进行排序，得到Mapper的最终输出结果，如图4‑5所示.

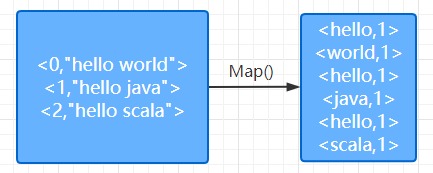


图 4‑6 得到map方法输出

* Reduce: Reducer先对从Mapper接受的<key，value>对进行shuffle处理，然后再交给用户自定义的reduce方法进行合并处理，再得到新的<key，value>对，最后将这个<key，value>对作为wordcount的结果输出，如图4‑7所示.

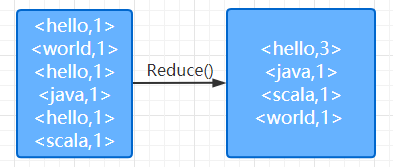


图 4‑8 用reduce方法进行合并处理并输出

## Spark惰性计算及简单实践

开启虚拟机，用start-all.sh命令打开hadoop集群 ，然后先全路径打开Spark.

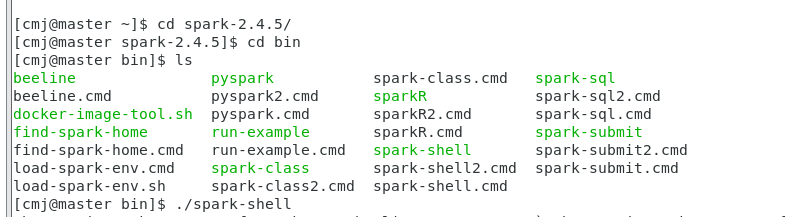


图 4‑9 全路径打开Spark

### 从文件系统中加载数据创建RDD

1. 从本地文件系统中加载数据创建RDD

先新建目录和word.txt文档.

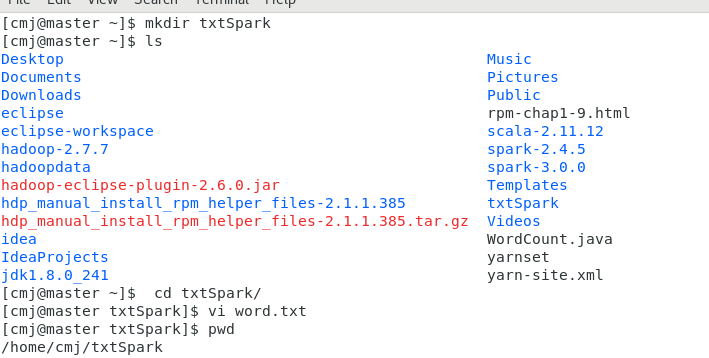


图 4‑10 新建目录和文档

测试获取文档中的内容.

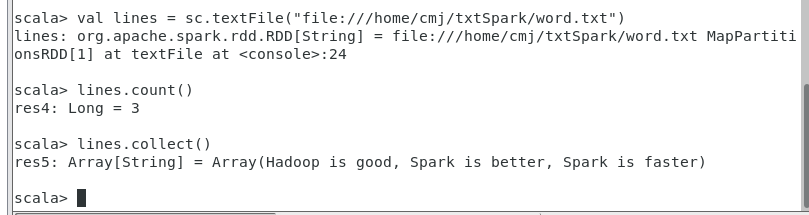


图 4‑11 获取文档中的内容

1. 从分布式文件系统HDFS中加载数据

创建删除文件夹（需要加-r，删除文件不需要加-r）



图 4‑12 创建删除文件夹操作

输入 “hadoop fs -ls /”命令查看hadoop分布式文件系统HDFS中的文件夹

或者进入<http://master:50070/explorer.html#/> 网址直接查看文件夹

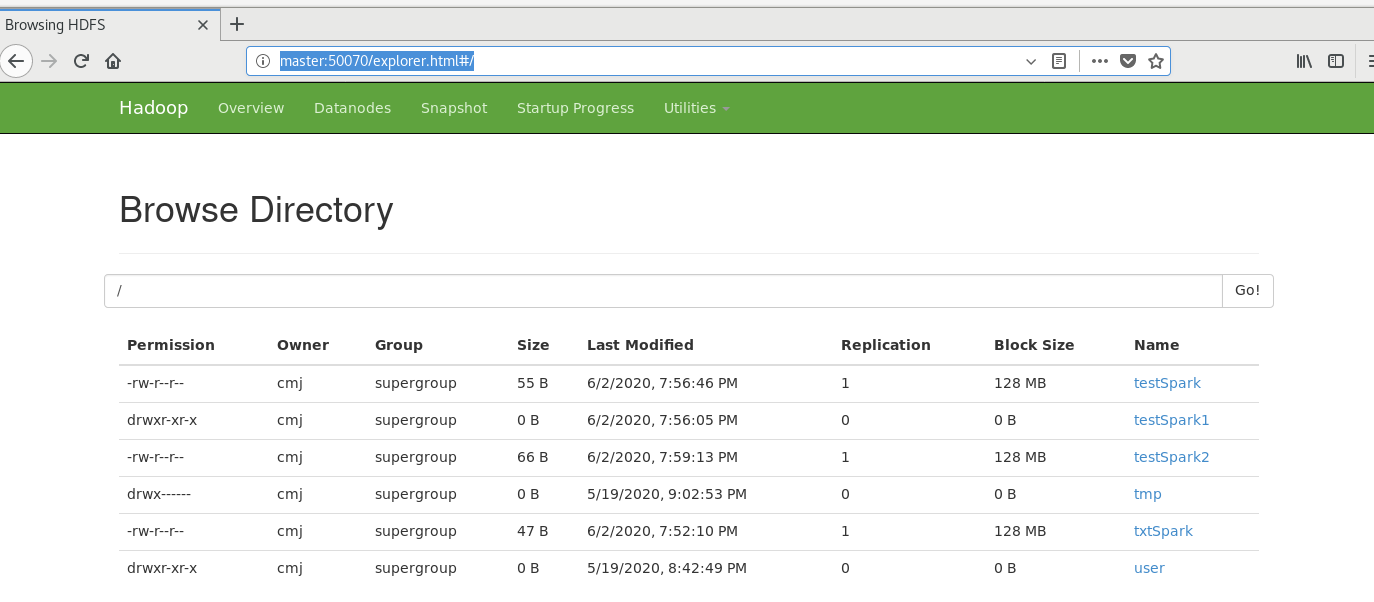


图 4‑13 查看HDFS文件夹

输入命令：

val hdfsfile = sc.textFile("hdfs://master:9000/testSpark1/hdfsword.txt")

因为惰性计算，所以还要输入 hdfsfile.count() 开始执行，截图如下：

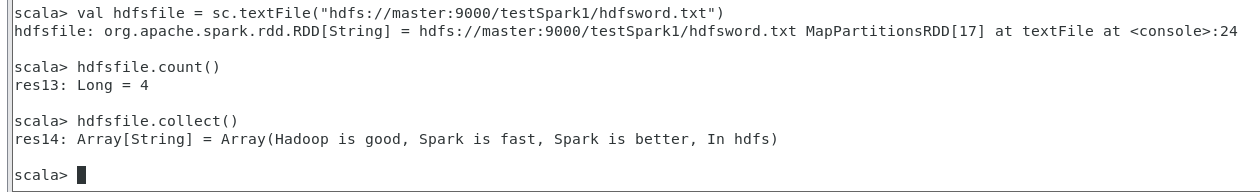


图 4‑14 Spark惰性运算

### 通过并行集合（数组）创建RDD

可以调用SparkContext的parallelize方法，在Driver中一个已经存在的集合（数组）上创建.

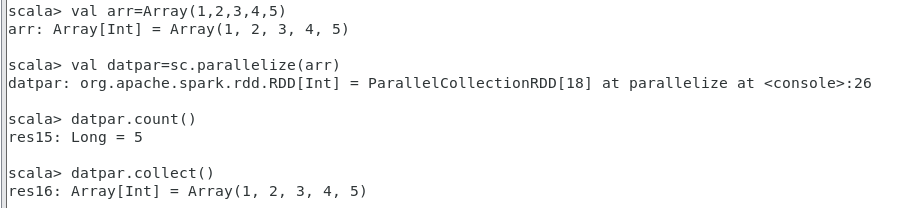


图 4‑15 parallelize（）创建RDD

或者，也可以从列表中创建：



图 4‑16从列表中创建RDD

### 常用的RDD转换操作API

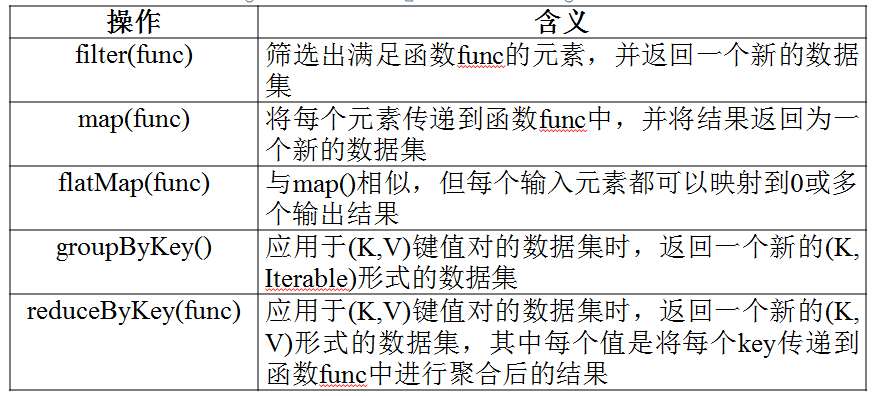


图 4‑17 常用的RDD转换操作

1. filter(func)

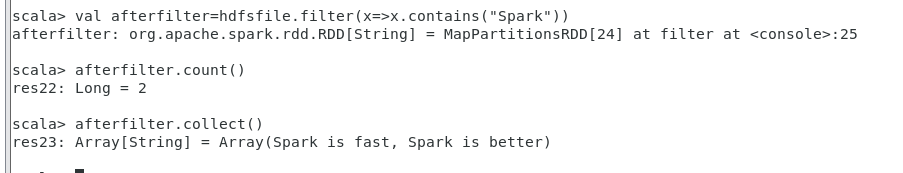


图 4‑18 filter(func)

1. map(func)

map(func)操作将每个元素传递到函数func中，并将结果返回为一个新的数据集

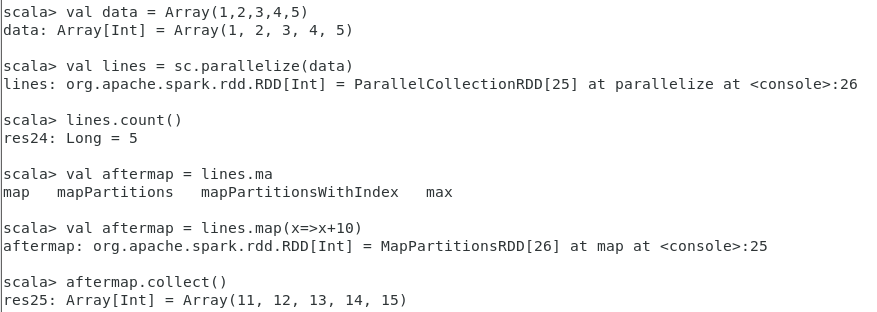


图 4‑19 map(func) - 1

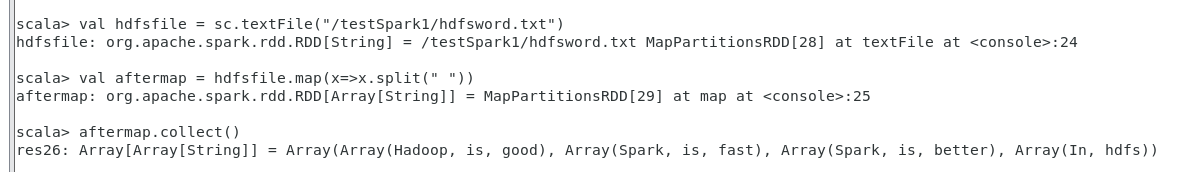


图 4‑20 map(func) - 2

1. flatMap(func)

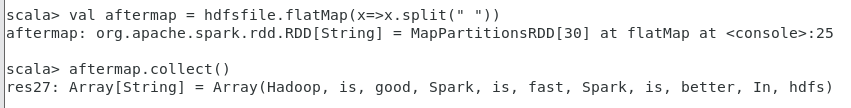


图 4‑21 flatMap(func)

1. groupByKey()

groupByKey()应用于(K，V)键值对的数据集时，返回一个新的(K， Iterable)形式的数据集

1. reduceByKey(func)

reduceByKey(func)应用于(K，V)键值对的数据集时，返回一个新的(K， V)形式的数据集，其中的每个值是将每个key传递到函数func中进行聚合后得到的结果

### 从本地文件系统中加载数据创建RDD

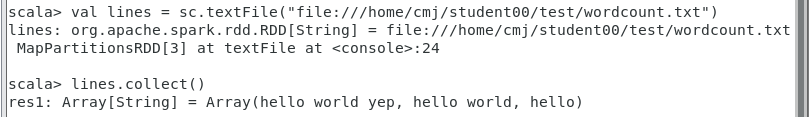


图 4‑22 读取文件并显示内容

读取文件：

val lines = sc.textFile("file:///home/cmj/student00/test/wordcount.txt")

分词：val aftermap = lines.map(x=>x.split(" "))

结果: Array[String] = Array(hello world yep， hello world， hello)

将分词结果转换为键值对，例（hello，3）val mkKV=aftermap.map((\_，1))

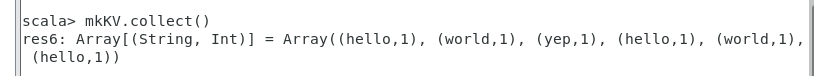


图 4‑23 分词结果

聚合：val afterreduce=mkKV.reduceByKey((x，y)=>(x+y))



图 4‑24 聚合结果

保存：afterreduce.saveAsTextFile("file:///home/cmj/student00/test/wordout1")



图 4‑25 保存结果

sc.textFile("file:///home/cmj/student00/test/wordcount.txt").flatMap(x=>x.split(" ")).map((\_，1)).reduceByKey((x，y)=>(x+y)).saveAsTextFile("file:///home/cmj/student00/test/wordout2")

### idea实现WordCount程序

1. 全路径打开idea

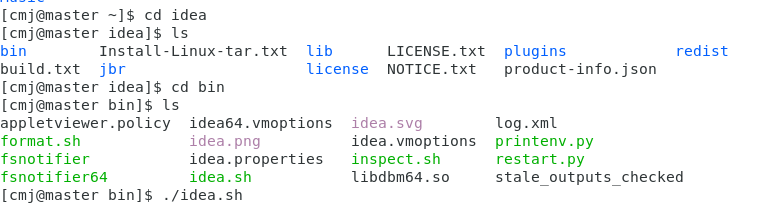


图 4‑26 全路径打开idea

1. 新建project

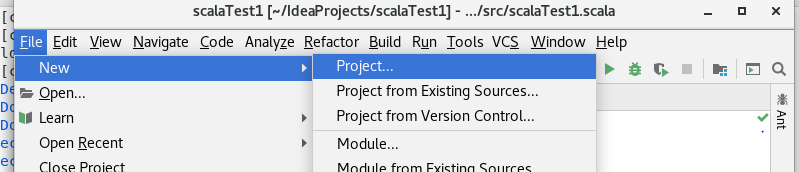


图 4‑27 新建project -1

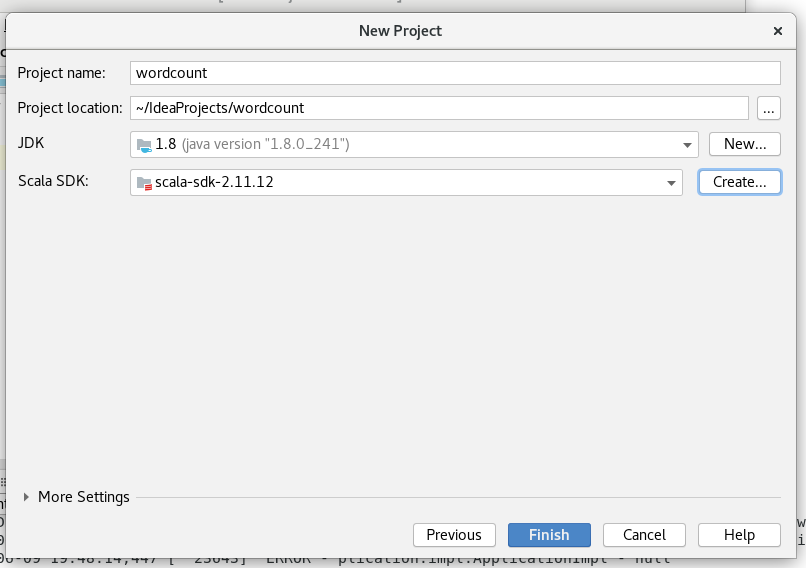


图 4‑28 新建project -2

1. 新建包



图 4‑29 新建包

1. 新建类

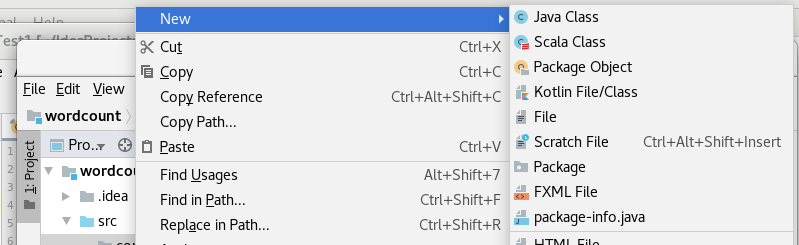


图 4‑30 新建类

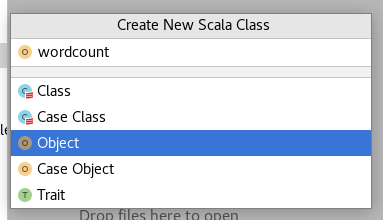


图 4‑31 新建Object

1. 实验主程序

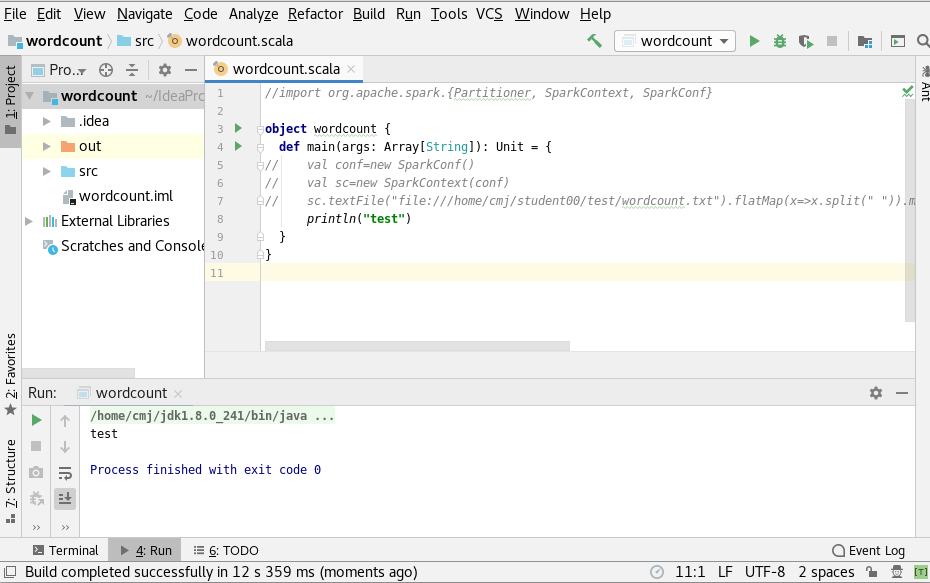


图 4‑32 实验主程序（报错）

选择“File”菜单栏下的“Project Structure…”.

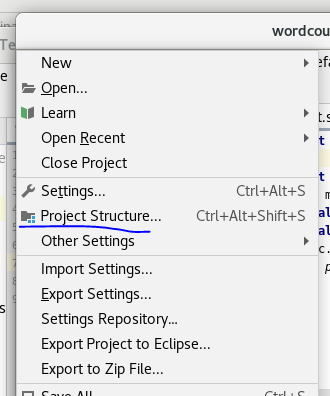


图 4‑33 选择“File”菜单栏下的“Project Structure…”

添加java包.

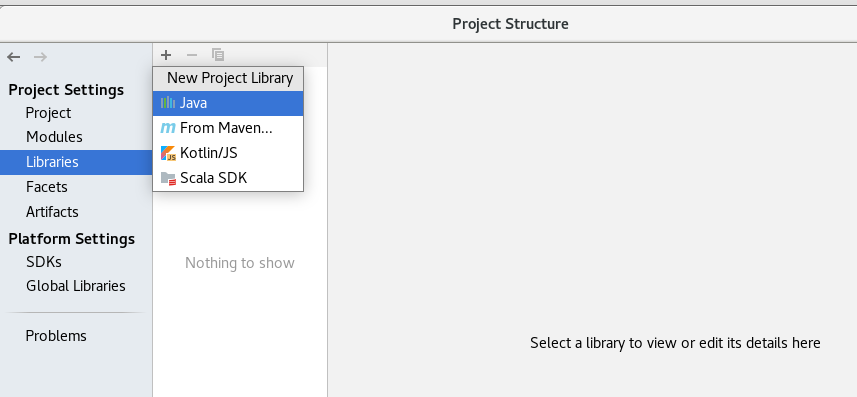


图 4‑34 添加java包

编辑Configurations，添加信息：

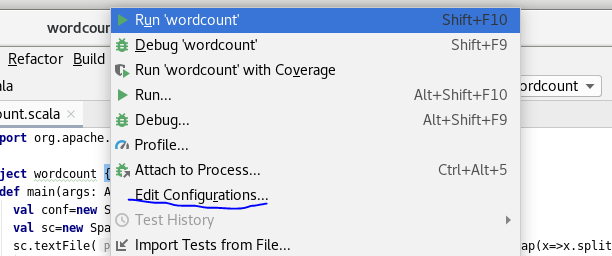


图 4‑35 编辑Configurations - 1

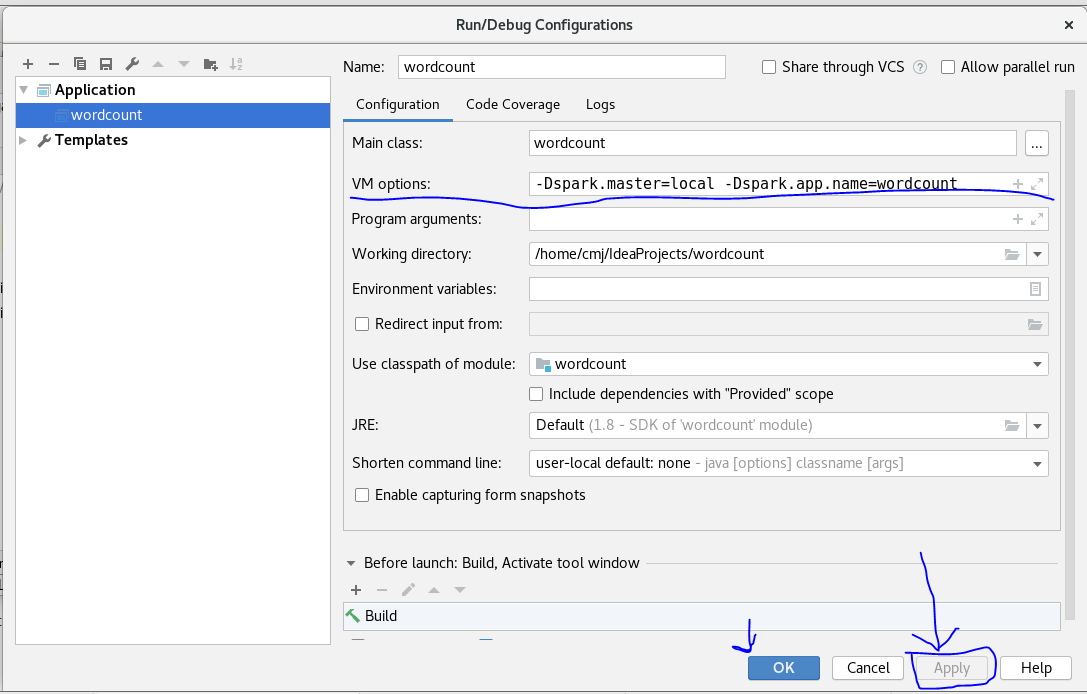


图 4‑36 编辑Configurations - 2

运行退出码是0说明运行成功

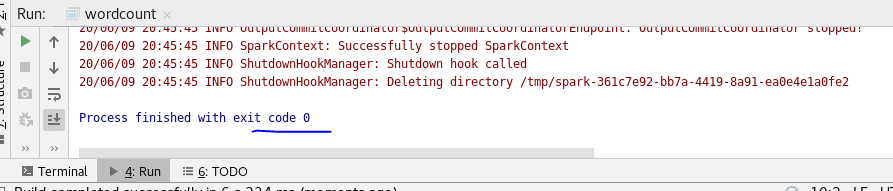


图 4‑37 运行成功

### WordCount生成jar包上传HDFS

1. 查看HDFS目录下的文件夹

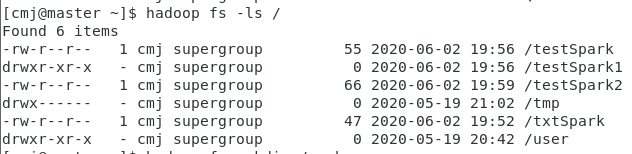


图 4‑38 查看HDFS目录

1. 新建文件夹

输入命令：hadoop fs -mkdir /csnb

上传文件到HDFS目录.



图 4‑39 上传文件到HDFS目录

短路径：val lines=sc.textFile("/csnb/wordcount.txt")

完整url（“hdfs：//master:9000/csnb/wordcount.txt”）



图 4‑40 短路径读取文件

全路径打开idea，修改4.4.5的程序路径.

1. 生成jar包

选择菜单栏中的“File”，点击“Project Structure”.

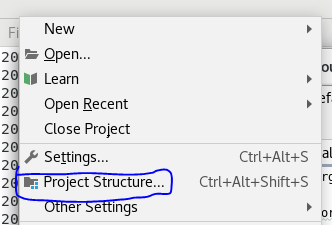


图 4‑41 点击“Project Structure”

选择“Artifacts”选项卡.

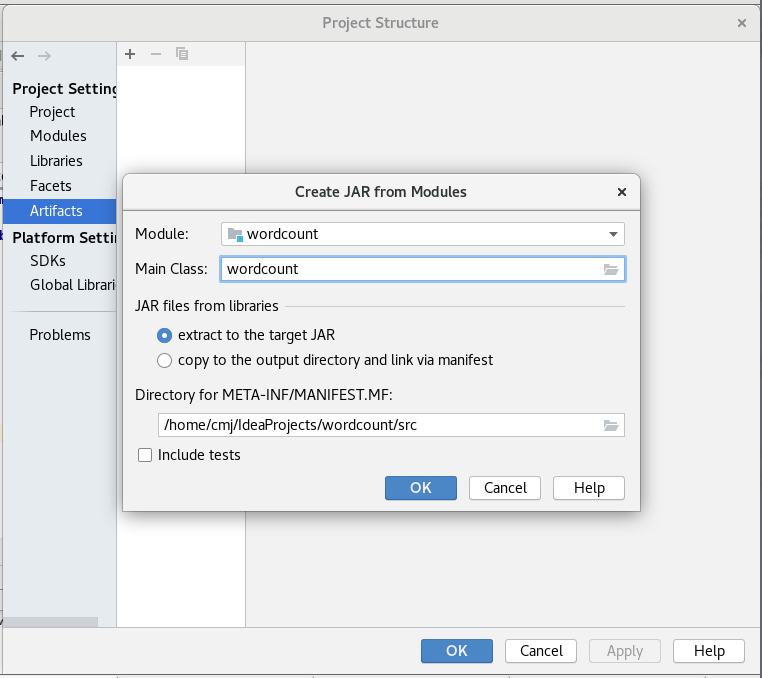


图 4‑42 选择“Artifacts”选项卡

在Main Class标签，点击右边的文件夹图标，browse搜索输入之后点击“OK”

删除掉多余的jar包，剩下最后一个wordcount.jar，变成“瘦包”

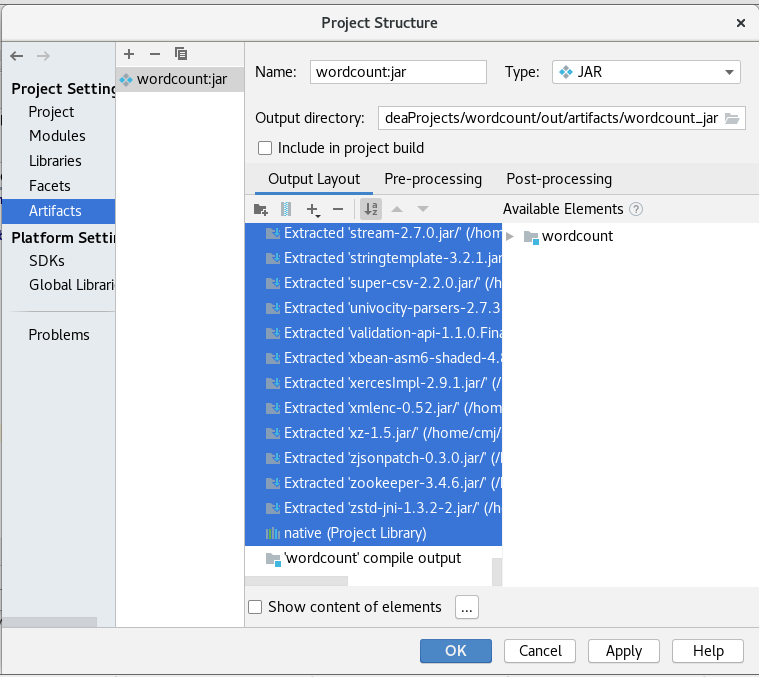


图 4‑43 删除掉多余的jar包

在“Build”菜单下选择“Build Artifacts..”选项卡.

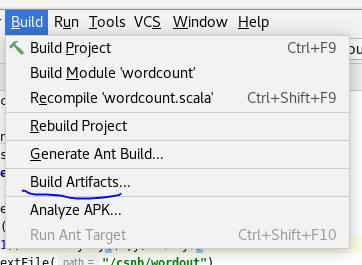


图 4‑44 选择“Build Artifacts..”选项卡

因为是第一次，选择“Build”.

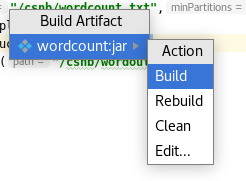


图 4‑45 选择“Build”

打包完成后，scp远程拷贝.

首先进入目录，如图4‑46所示.



图 4‑47 目录

点击jar文件右键打开Terminal编辑器，将jar文件拷贝到master中.

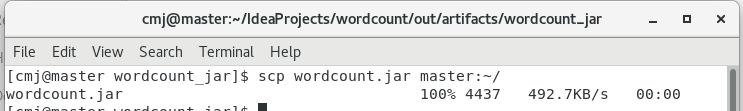


图 4‑48 将jar文件拷贝到master中

在家目录下执行：/home/cmj/spark-2.4.5/bin/spark-submit --master yarn-cluster wordcount.jar

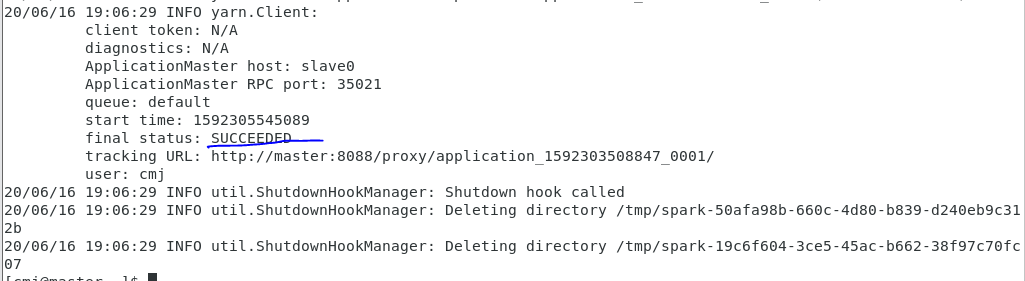


图 4‑49 执行结果

出现即表示成功.

1. 利用cat命令查看分区内容



图 4‑50 利用cat命令查看分区内容

1. WordCount程序统计多个文件夹内容

首先删除csnb文件夹下的所有内容.

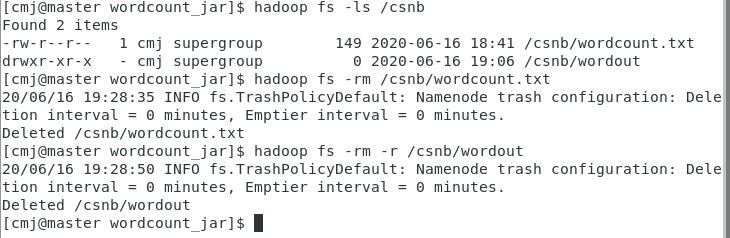


图 4‑51 删除文件夹下的所有内容

建立一个wordin文件夹，用来存放所有的txt文件.

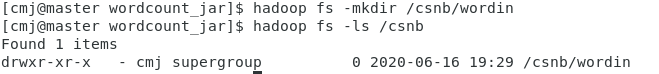


图 4‑52 建立wordin文件夹

上传俩个txt到hdfs上去，



图 4‑53 上传俩个txt文件

在idea项目中修改路径为wordin文件夹，选择Build下的“Build Artifacts”（因为之前已经build过了）所以这里选择“rebuild”

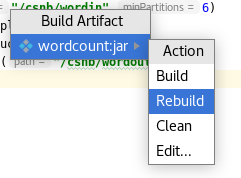


图 4‑54 选择“rebuild”选项

等待build完成之后重复之前的操作，上传jar包到hdfs.

# 课程过程中遇到的问题以及解决方法

## Hadoop 警告: WARN hdfs.DFSClient: DataStreamer Exception org.apache.hadoop.ipc.RemoteException

这是在配置hadoop集群的时候遇到的错误.解决方法：

首先要先检查一下防火墙是否已经关闭了，**防火墙没关**可能导致报错.还有一种可能就是在**core-site.xml**中没有配置hadoop运行时产生的文件的存储目录.

如果上述都没问题的话，有可能是使用hadoop namenode -format格式化时格式化了多次造成那么spaceID不一致造成的，这时候就需要停止集群 stop-all.sh，然后删除在hdfs中配置的data目录（就是在core-site.xml中配置的hadoop.tmp.dir对应的文件）下面的所有数据rm -rf /home/user1/hadoop2.7/hadoopdata/\*（注意这里的目录就是你core-site.xml中配置的hadoop.tmp.dir对应的目录）.

接着重新格式化namenode（需要切换到bin目录）./hadoop namenode -format

最后重新启动hadoop集群再操作就行了.

## Hadoop报错 : INFOmapred.ClientServiceDelegate:Application state is completed. FinalApplicationStatus=FAILED

这是在配置hadoop集群的时候遇到的错误.

jps发现job history service 没有启动，解决方法：

解决方法就是启动historyserver进程:输入命令 mr-jobhistory-daemon.sh start historyserver 解决.

# 课程思考和个人总结

通过从开学到现在的大数据课程学习，我收获了很多知识，得到了很大的提升，并把大数据课程的内容记录在了个人博客的专栏：

<https://blog.csdn.net/qq_44702847/category_9785663.html>.

因为疫情的特殊原因，这学期的大数据课大多时间运用在系统搭建而非数据处理和程序编写上.在家的线上教学，让我打开了新视野，了解到了大数据这个新领域，同时也让我的动手实践能力得到了很大进步.在系统搭建的过程中我遇到了超级多的问题，但很幸运的是，老师的耐心解答和同学的热心帮助都让我有惊无险地解决了很多大大小小的问题.另外我经历过的问题大多数通过百度的方式都可以解决.另外，返校之后的物理集群搭建是我们对课堂上所学知识的巩固和实践最佳的方式.在那节课上，我感到有些手足无措，因为我的理论知识不能与实践相结合，从何遇到很多细节上的问题，导致Hadoop集群无法正常启动.后来，在小组成员的相互帮助和自己不懈的努力下，我们终于确保了Hadoop集群的而正常启动.在系统搭建的过程中，我们小组也遇到过很多问题，但大家都没有退怯或者逃避责任，而是一起讨论问题，积极地靠自己的努力解决问题.随着课程的紧密进行，我对大数据也有了越发完善的了解，包括Hadoop集群的搭建和启动，明白了全路径启动和系统配置路径的方法，理解了WordCount程序的运行原理，让我的实践能力大大提升，也让我对大数据运算的概念更加理解透彻.在这个过程中，我曾多次延续上课没跟上的内容直到深夜，但是我也常常为问题得以解决而欢呼雀跃，这一切都是值得的.

# 附录

1. Vi编辑器采用操作汇总博客：

<https://blog.csdn.net/xcg132566/article/details/78797282>

1. GPLv2自由软件协议拓展资料: <http://c.biancheng.net/view/674.html>
2. YARN的架构及原理介绍学习博客：

<https://www.cnblogs.com/zimo-jing/p/8846569.html>

1. Spark RDD计算框架学习博客：

<https://blog.csdn.net/Mai_NO/article/details/87184740>