一、搭建 Hadoop 分布式集群环境

一. 安装 jdk

打开一个虚拟机，右键单击桌面选择 Open in Terminal，进入编辑界面：

1.假设用户名是 user(获取 root 权限)

(1)使 user 成为 sudoer

```
su
cd  /etc
vi  sudoers
```

i 进入编辑状态

在 root ALL=(ALL) ALL 的下一行编辑

```
user  ALL=(ALL) ALL
```

按 ESC 键

按 Shift + :

输入 wq!

(2)创建 hadoop 文件夹

```
cd
mkdir  hadoop
```

(3)解压 jdk-7u79-linux-x64.gz 文件

```
cd
cd  hadoop
tar  -zxvf  /home/user/Desktop/jdk-7u79-linux-x64.gz
```

```
文件(F)  编辑(E)  查看(V)  搜索(S)  终端(T)  帮助(H)
jdk1.8.0_101/jre/lib/ext/nashorn.jar
jdk1.8.0_101/jre/lib/fontconfig.SuSE.10.properties.src
jdk1.8.0_101/jre/lib/net.properties
jdk1.8.0_101/jre/lib/fonts/
jdk1.8.0_101/jre/lib/fonts/LucidaSansDemiBold.ttf
jdk1.8.0_101/jre/lib/fonts/LucidaBrightRegular.ttf
jdk1.8.0_101/jre/lib/fonts/LucidaTypewriterRegular.ttf
jdk1.8.0_101/jre/lib/fonts/LucidaBrightDemiItalic.ttf
jdk1.8.0_101/jre/lib/fonts/LucidaBrightItalic.ttf
jdk1.8.0_101/jre/lib/fonts/LucidaSansRegular.ttf
jdk1.8.0_101/jre/lib/fonts/LucidaTypewriterBold.ttf
jdk1.8.0_101/jre/lib/fonts/LucidaBrightDemiBold.ttf
jdk1.8.0_101/jre/lib/fonts/fonts.dir
jdk1.8.0_101/jre/lib/fontconfig.SuSE.10.bfc
jdk1.8.0_101/jre/lib/fontconfig.RedHat.5.properties.src
jdk1.8.0_101/jre/lib/content-types.properties
jdk1.8.0_101/jre/lib/resources.jar
jdk1.8.0_101/jre/lib/meta-index
jdk1.8.0_101/jre/lib/tzdb.dat
jdk1.8.0_101/jre/lib/calendars.properties
jdk1.8.0_101/jre/lib/flavormap.properties
jdk1.8.0_101/jre/lib/fontconfig.properties.src
jdk1.8.0_101/jre/lib/rt.jar
```

(4) 设置 jdk 环境变量

cd
cd hadoop
su
vi /etc/profile

进入后在最后一行添加以下指令：

export JAVA_HOME=/home/user/hadoop/jdk1.7.0_79
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar

点击保存后关闭，输入以下指令使 jdk 生效：

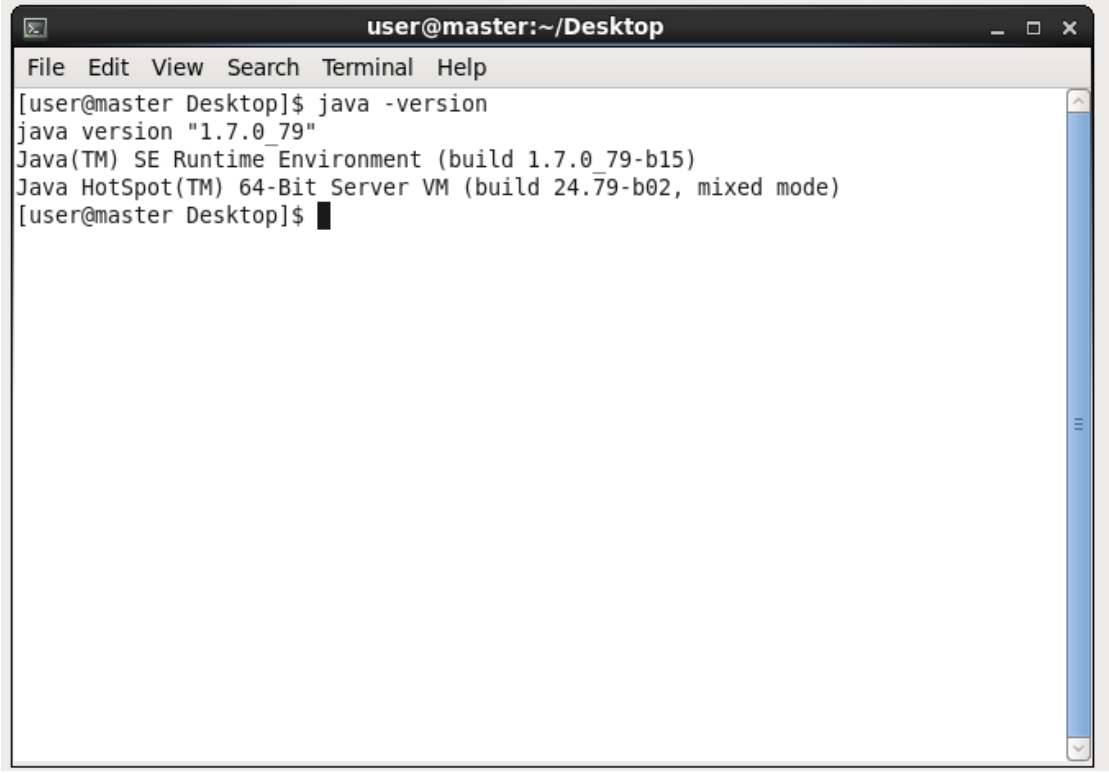
source /etc/profile

(5) 检查 jdk 是否安装成功

java -version

成功后显示如下信息：

java version "1.7.0_79"
Java(TM) SE Runtime Environment (build 1.7.0_79-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.79-b02, mixed mode)



二、创建集群

1. 克隆虚拟机

将已经安装好 jdk 的虚拟机克隆两个，创建三个虚拟机的集群。

2. 修改后 hostname

su
vi /etc/sysconfig/network

将三个虚拟机分别命名 master、slave1、slave2

如图：（完成后重启虚拟机 reboot）

3. 将三个虚拟机的 ip 地址相互连接

首先必须确保虚拟机联网，如果 NET 模式连不上网，则选中桥接模式。

网络通畅后执行以下操作：

(1) 分别对三个虚拟机执行指令 **ifconfig**，查看各虚拟机 ip 地址

192.168.88.131

192.168.88.132

192.168.88.133

(2)在 master 中执行以下指令

```
su
cd   /etc
gedit   /etc/hosts
```

进入编辑界面后按“IP 地址　　hostname”填写信息，如图：

填写完后按 Save 按钮，关闭编辑页。

(3)将配置好的文件复制到 slave1、slave2 中

在 master 中执行以下指令：

```
scp   /etc/hosts   root@slave1:/etc/
scp   /etc/hosts   root@slave2:/etc/
```

(4)检查各虚拟机是否互联

在 master 中执行以下指令：

```
ping   slave1
ping   slave2
```

```
user@master:~/Desktop
File  Edit  View  Search  Terminal  Help
[user@master Desktop]$ ping slave1
PING slave1 (192.168.88.132) 56(84) bytes of data.
64 bytes from slave1 (192.168.88.132): icmp_seq=1 ttl=64 time=0.219 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=2 ttl=64 time=0.212 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=3 ttl=64 time=0.192 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=4 ttl=64 time=0.313 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=5 ttl=64 time=0.202 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=6 ttl=64 time=0.340 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=7 ttl=64 time=0.214 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=8 ttl=64 time=0.395 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=9 ttl=64 time=0.184 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=10 ttl=64 time=0.276 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=11 ttl=64 time=0.235 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=12 ttl=64 time=0.378 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=13 ttl=64 time=0.205 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=14 ttl=64 time=0.186 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=15 ttl=64 time=0.207 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=16 ttl=64 time=0.205 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=17 ttl=64 time=0.474 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=18 ttl=64 time=0.264 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=19 ttl=64 time=0.153 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=20 ttl=64 time=0.227 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=21 ttl=64 time=0.198 ms
64 bytes from slave1 (192.168.88.132): icmp_seq=22 ttl=64 time=0.186 ms
```

4. 配置 SSH 无密钥登录

(1) 关闭防火墙

对每个虚拟机进行如下操作：

su
chkconfig iptables off

执行后重启虚拟机： **reboot**

(2) 关闭防火墙后在 master 下执行以下指令：一路回车

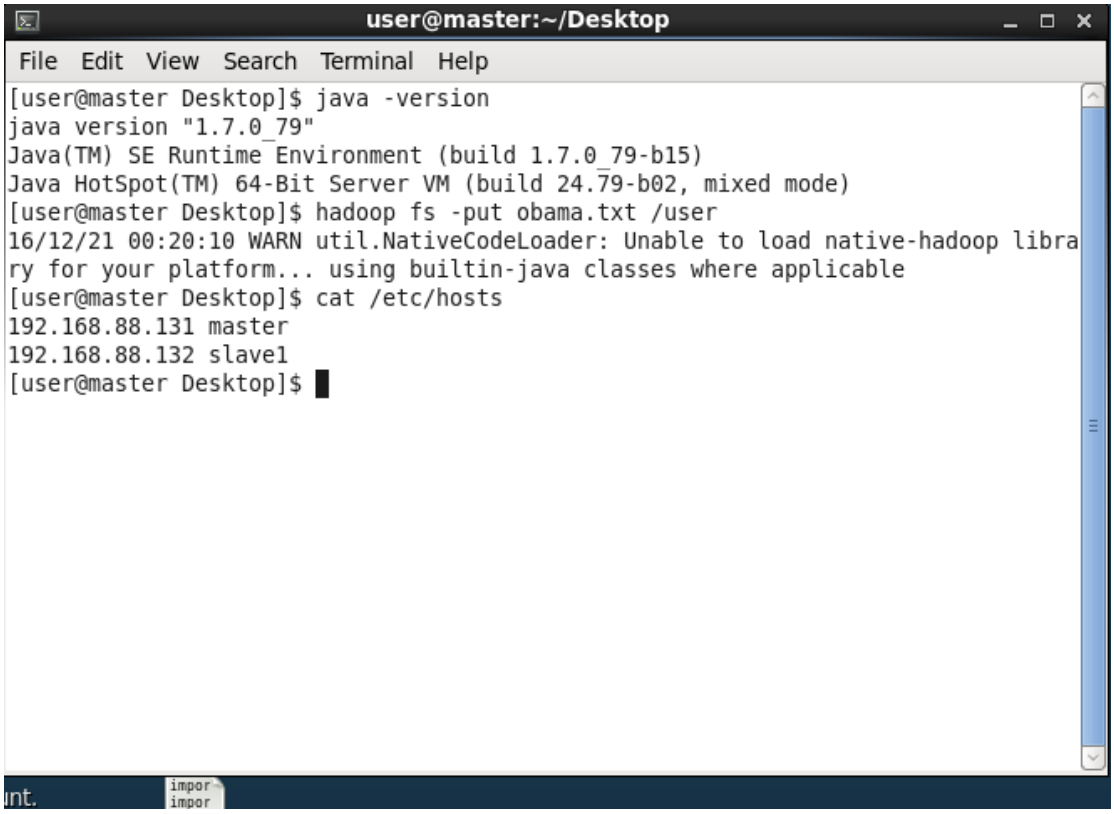
cd
ssh-keygen –t rsa
cd .ssh
cat id_rsa.pub >> authorized_keys
chmod 600 authorized_keys
sudo chmod 700 ~/.ssh
scp authorized_keys user@slave1:~/.ssh/
scp authorized_keys user@slave2:~/.ssh/

(3) 检查无密钥登录是否成功

ssh slave1
ssh slave2

ssh　master

成功后显示如下：



5. 安装并配置 hadoop-2.6.4(在 master 中)

(1)将 hadoop-2.6.4.tar.gz 安装包复制到 hadoop 文件目录下（与 windows 环境下类似)。

(2)解压 hadoop-2.6.4.tar.gz

cd
cd　hadoop
tar　-zxvf　/home/user/Desktop/hadoop-2.6.4.tar.gz

(3)配置 hadoop-2.6.4 的各项文件

cd
cd　hadoop/hadoop-2.6.4
cd　etc/hadoop
gedit　hadoop-env.sh

在最后一行添加:**export JAVA_HOME=/home/user/hadoop/jdk1.7.0_79**

gedit　core-site.xml

添加代码：

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://master:9000</value>
    <final>true</final>
</property>
<property>
    <name>hadoop.tmp.dir</name>
    <value>/home/user/hadoop/tmp</value>
</property>
<property>
    <name>ds.default.name</name>
    <value>hdfs://master:54310</value>
    <final>true</final>
</property>
```

gedit　hdfs-site.xml

添加代码：

```
<property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/home/wxx/hadoop/dfs/name</value>
    <final>true</final>
</property>
<property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/home/user/hadoop/dfs/data</value>
    <final>true</final>
</property>
<property>
    <name>dfs.replication</name>
    <value>2</value>
</property>
```

gedit　mapred-site.xml

（注意：必须先复制 mapred-site.xml.template 文件更名为 mapred-site.xml）

添加代码：

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

```
        <name>mapreduce.jobhistory.address</name>
        <value>master:10020</value>
    </property>
    <property>
        <name>mapreduce.jobhistory.webapp.address</name>
        <value>master:19888</value>
    </property>
```

gedit　yarn-site.xml

添加代码：

```
    <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.hostname</name>
        <value>master</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:8031</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>
```

gedit　master

添加代码：

```
master

gedit   slaves
```

添加代码：

```
master
slave1
slave2
```

(4)将配置好的文件复制到 slave1、slave2 中

```
cd
cd   hadoop
scp   -r   hadoop-2.6.4   slave1:~/hadoop
scp   -r   hadoop-2.6.4   slave2:~/hadoop
```

(5)启动集群

```
cd
cd   hadoop/hadoop-2.6.4
bin/hdfs   namenode   -format          // 格式化 namenode
sbin/start-dfs.sh
sbin/start-yarn.sh
sbin/hadoop-daemon.sh   start   secondarynamenode
```

(6)检查集群情况

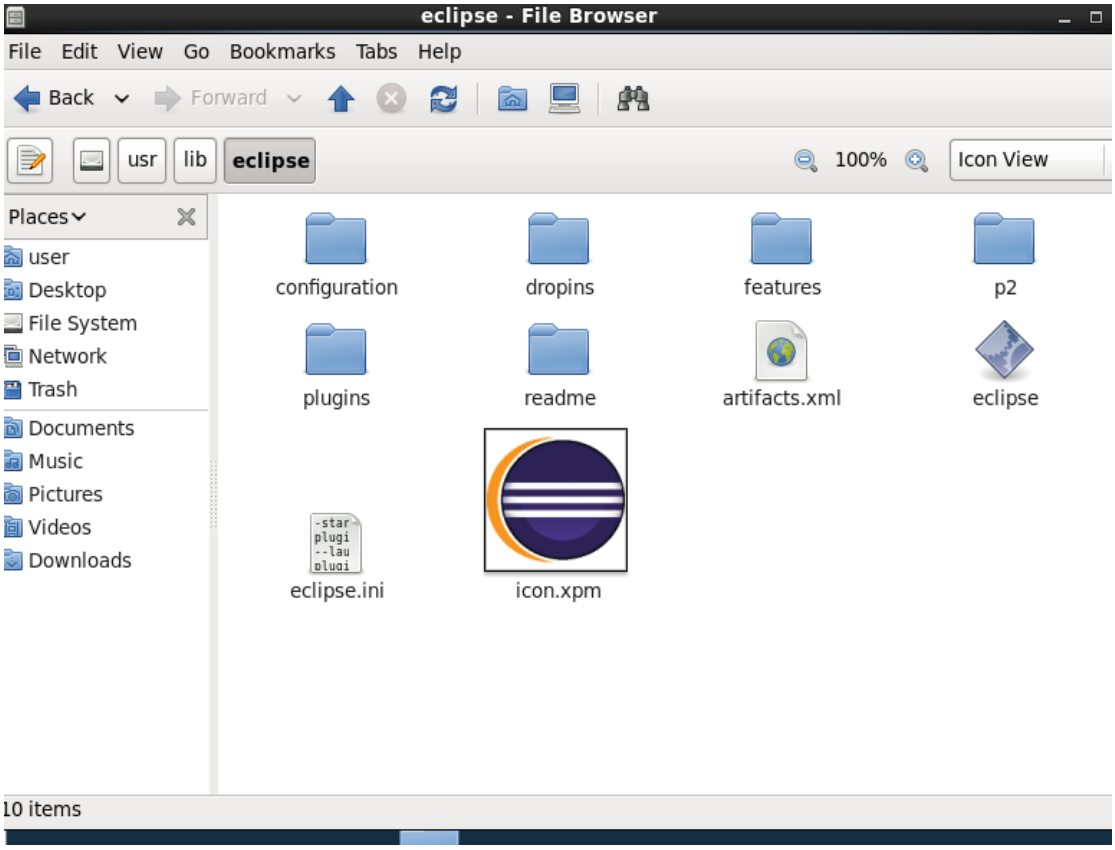
```
jps
```

```
user@master:~/Desktop
File  Edit  View  Search  Terminal  Help
[user@master Desktop]$ hadoop fs -put obama.txt /user
16/12/20 22:28:50 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
put: `obama.txt': No such file or directory
[user@master Desktop]$ hadoop fs -put obama.txt /user
16/12/20 22:29:31 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
[user@master Desktop]$ hadoop fs -put shu /user
16/12/20 22:55:44 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
[user@master Desktop]$ jps

2898 SecondaryNameNode
10058 Jps
3040 ResourceManager
2620 NameNode
3701 DataNode
7310 org.eclipse.equinox.launcher_1.3.100.v20150511-1540.jar
3139 NodeManager
[user@master Desktop]$
[user@master Desktop]$
```

三. 安装 eclipse

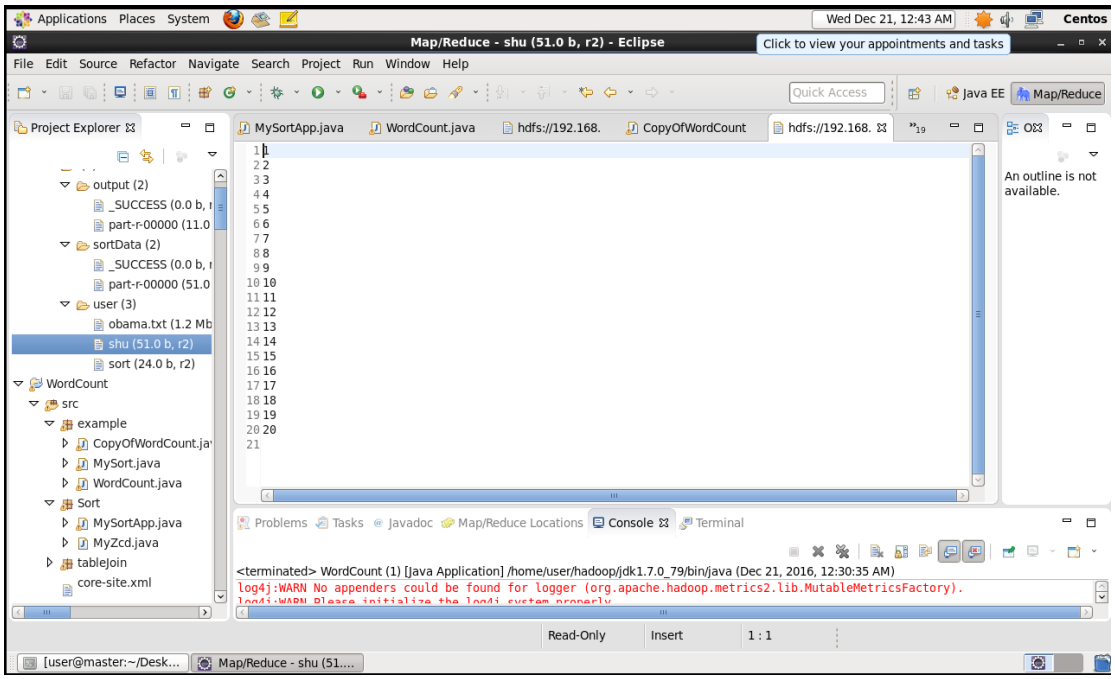
sudo tar –zxf /home/user/Desktop/eclipse-jee-mars-2-linux-gtk-x86_64.tar.gz -C /usr/lib



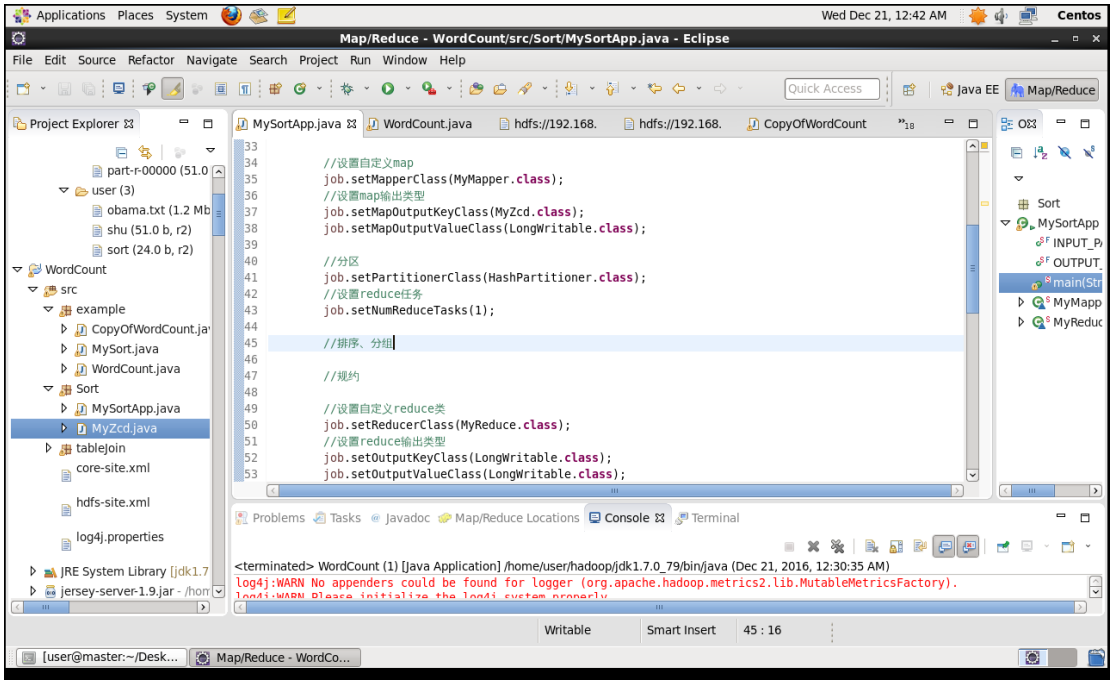
安装 **Hadoop-Eclipse-Plugin**

四.给定一些同时包含奇数和偶数的数字，对这些数字进行排序，要求：所有的奇数在所有的偶数前，且奇数按照升序排列，偶数按照降序排列。

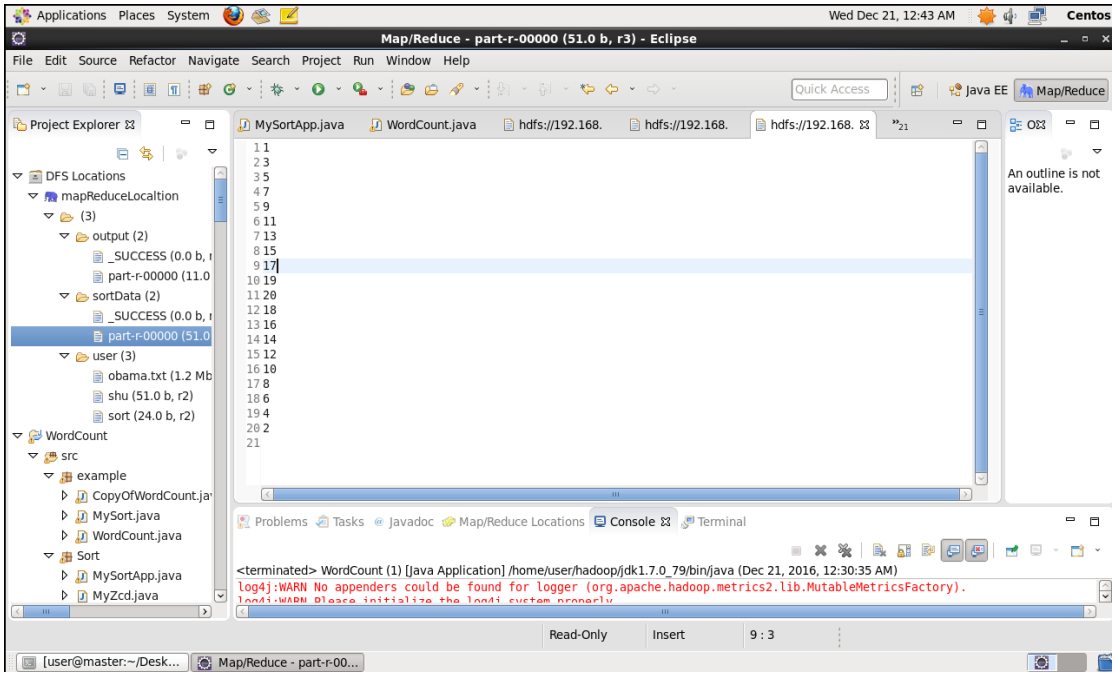
输入文件：数据包括 **1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20** 升序排列的数



实验过程截图



输出文件:输出数据 **1 3 5 7 9 11 13 15 17 19 20 18 16 14 12 10 8 6 4 2** ，按照奇数在前偶数在后，奇数升序排列偶数降序排序排列。



MySortApp.java

package Sort;

```
import java.io.IOException;
import java.net.URI;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.partition.HashPartitioner;

public class MySortApp {

    public static final String INPUT_PATH="hdfs://192.168.88.131:9000/user/shu";
    public static final String OUTPUT_PATH="hdfs://192.168.88.131:9000/sortData";

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf,MySortApp.class.getSimpleName());

        //设置输入路径
        FileInputFormat.addInputPath(job, new Path(INPUT_PATH));
        //设置输入格式化
        job.setInputFormatClass(TextInputFormat.class);

        //设置自定义 map
        job.setMapperClass(MyMapper.class);
        //设置 map 输出类型
        job.setMapOutputKeyClass(MyZcd.class);
        job.setMapOutputValueClass(LongWritable.class);

        //分区
        job.setPartitionerClass(HashPartitioner.class);
        //设置 reduce 任务
        job.setNumReduceTasks(1);

        //排序、分组
```

```
        //规约

        //设置自定义 reduce 类
        job.setReducerClass(MyReduce.class);
        //设置 reduce 输出类型
        job.setOutputKeyClass(LongWritable.class);
        job.setOutputValueClass(LongWritable.class);

        //删除已存在的路径
        FileSystem fileSystem = FileSystem.get(new URI(INPUT_PATH), conf);
        Path path=new Path(OUTPUT_PATH);
        if(fileSystem.exists(path)){
            fileSystem.delete(path,true);
        }

        //设置输出路径
        FileOutputFormat.setOutputPath(job, new Path(OUTPUT_PATH));
        //设置输出格式化类
        job.setOutputFormatClass(TextOutputFormat.class);

        //提交任务
        job.waitForCompletion(true);
    }

    static class MyMapper extends Mapper<LongWritable, Text, MyZcd, LongWritable>{
        @Override
        protected void map(LongWritable key, Text value,Context context)
                throws IOException, InterruptedException {
            MyZcd myK2 = new MyZcd(Long.parseLong(value.toString()));
            context.write(myK2, new LongWritable(Long.parseLong(value.toString())));
        }
    }

    static class MyReduce extends Reducer<MyZcd, LongWritable,LongWritable, LongWritable>{
        @Override
        protected void reduce(MyZcd myk2, Iterable<LongWritable> v2s,Context context)
                throws IOException, InterruptedException {
            context.write(new LongWritable(myk2.myk2),null);
        }
    }
}
```

MyZcd.java

```
package Sort;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

import org.apache.hadoop.io.WritableComparable;

public class MyZcd implements WritableComparable<MyZcd>{

	public long myk2;


	public MyZcd() {
		// TODO Auto-generated constructor stub
	}

	public MyZcd(long myk2) {
		this.myk2 = myk2;

	}

	@Override
	public void write(DataOutput out) throws IOException {
		out.writeLong(myk2);

	}

	@Override
	public void readFields(DataInput in) throws IOException {
		this.myk2=in.readLong();

	}

	/**
	核心代码
	 * 首先按照奇数大于偶数排序，然后同是奇数按照升序排列
	 ，偶数按照降序排列。
	 */

	@Override
	public int compareTo(MyZcd my) {
		long temp1=this.myk2%2-my.myk2%2;
		if(temp1>0){
```

```
                temp1=-1;
                return (int) temp1;
            }else if(temp1<0){
                temp1=1;
                return (int) temp1;
            }
            long temp=this.myk2-my.myk2;
            if(this.myk2%2==0){
                return -(int) (temp);
            }else{
                return (int)temp;
            }

        }

        /*
         *
         *
         * @param obj
         * @return
         * @see java.lang.Object#equals(java.lang.Object)
         */
        @Override
        public boolean equals(Object obj) {
            // TODO Auto-generated method stub
            return super.equals(obj);
        }

        /*
         *
         *
         * @return
         * @see java.lang.Object#hashCode()
         */
        @Override
        public int hashCode() {
            // TODO Auto-generated method stub
            return super.hashCode();
        }



    }
```