第一讲 在 win7 上安装配置 HADOOP 伪分布式集群

1、 安装虚拟机和操作系统

VMware-workstation-full-10.0.0 或 VirtualBox-4.2.18-88781-Win 下载 VMware 下载 VirtualBox
ubuntu-13.04-server-amd64.iso 下载 ubuntu

2、 设置 root 用户密码

sudo passwd root

3、 上传文件

利用 WinSCP 上传 JDK 和 HADOOP 文件，利用 putty 连接虚拟机中的 ubuntu，下载 WinSCP， 下载 putty， 下载 jdk，下载 jdk

4、 配置 JDK 和 HADOOP

tar -xzvf jdk-7u40-linux-x64.tar.gz
tar -xzvf hadoop-1.2.1.tar.gz
sudo vi /etc/profile
增加：
export JAVA_HOME=/home/ysc/jdk1.7.0_40
export PATH=$PATH:$JAVA_HOME/bin: /home/ysc/hadoop-1.2.1/bin
source /etc/profile

5、配置 HADOOP

配置主机名称及网络

vi /etc/hostname
指定名称为 host001
vi /etc/hosts
替换内容为：192.168.137.128 host001
同时加入 C:\Windows\System32\drivers\etc \hosts 文件
查看是否启用 IPV6：
cat /proc/sys/net/ipv6/conf/all/disable_ipv6
显示 0 说明 ipv6 开启，1 说明关闭
关闭 ipv6 的方法：
sudo vi /etc/sysctl.conf
增加下面几行，并重启
#disable IPv6

net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1

配置 SSH

cd /home/ysc
sudo apt-get install openssh-server
ssh-keygen -t rsa（密码为空，路径默认）
cp .ssh/id_rsa.pub .ssh/authorized_keys
ssh host001
yes
cd hadoop-1.2.1

配置 HADOOP 环境变量

vi conf/hadoop-env.sh
增加：
export JAVA_HOME=/home/ysc/jdk1.7.0_40

配置 HADOOP 运行参数

vi conf/masters
改 localhost 为 host001
vi conf/slaves
改 localhost 为 host001

vi conf/core-site.xml

```
<property>
<name>fs.default.name</name>
<value>hdfs://host001:9000</value>
</property>
 <property>
      <name>hadoop.tmp.dir</name>
      <value>/home/ysc/tmp</value>
</property>
```

vi conf/hdfs-site.xml

```
<property>
   <name>dfs.name.dir</name>
   <value>/home/ysc/dfs/filesystem/name</value>
</property>
<property>
   <name>dfs.data.dir</name>
   <value>/home/ysc/dfs/filesystem/data</value>
```

```
</property>
<property>
   <name>dfs.replication</name>
   <value>1</value>
</property>
```

vi conf/mapred-site.xml

```
<property>
   <name>mapred.job.tracker</name>
   <value>host001:9001</value>
</property>
<property>
   <name>mapred.tasktracker.map.tasks.maximum</name>
   <value>4</value>
</property>
<property>
   <name>mapred.tasktracker.reduce.tasks.maximum</name>
   <value>4</value>
</property>
<property>
   <name>mapred.system.dir</name>
   <value>/home/ysc/mapreduce/system</value>
</property>
<property>
   <name>mapred.local.dir</name>
   <value>/home/ysc/mapreduce/local</value>
</property>
```

格式化名称节点并启动集群

hadoop namenode -format

启动集群并查看 WEB 管理界面

start-all.sh

访问 http://host001:50030 可以查看 JobTracker 的运行状态

访问 http://host001:50060 可以查看 TaskTracker 的运行状态

访问 http://host001:50070 可以查看 NameNode 以及整个分布式文件系统的状态，浏览分布式文件系统中的文件以及 log 等

hadoop jar hadoop-1.2.1/contrib/streaming/hadoop-streaming-1.2.1.jar -input input -output output-streaming -mapper /bin/cat -reducer /usr/bin/wc

hadoop jar hadoop-1.2.1/hadoop-examples-1.2.1.jar wordcount input output

停止集群

stop-all.sh

第二讲 建立开发环境编写 HDFS 和 MAP REDUCE 程序

1、在 eclipse 中配置 hadoop 插件

将 hadoop-eclipse-plugin-1.2.1.jar 复制到 eclipse/plugins 目录下，重启 eclipse。

2、打开 MapReduce 视图

Window -> Open Perspective -> Other 选择 Map/Reduce，图标是个蓝色的象。

3、添加一个 MapReduce 环境

在 eclipse 下端，控制台旁边会多一个 Tab，叫"Map/Reduce Locations"，在下面空白的地方点右键，选择"New Hadoop location..."，在弹出的对话框中填写如下内容：

Location name（取个名字）

Map/Reduce Master（Job Tracker 的 IP 和端口，根据 mapred-site.xml 中配置的 mapred.job.tracker 来填写）

DFS Master（Name Node 的 IP 和端口，根据 core-site.xml 中配置的 fs.default.name 来填写）

4、使用 eclipse 对 HDFS 内容进行操作

经过上一步骤，左侧"Project Explorer"中应该会出现配置好的 HDFS，点击右键，可以进行新建文件夹、删除文件夹、上传文件、下载文件、删除文件等操作。

注意：每一次操作完在 eclipse 中不能马上显示变化，必须得刷新一下。

5、创建 MapReduce 工程

5.1 配置 Hadoop 路径

Window -> Preferences 选择 "Hadoop Map/Reduce"，点击"Browse..."选择 Hadoop 文件夹的路径。这个步骤与运行环境无关，只是在新建工程的时候能将 hadoop 根目录和 lib 目录下的所有 jar 包自动导入。

5.2 创建工程

File -> New -> Project 选择"Map/Reduce Project"，然后输入项目名称，创建项目。插件会自动把 hadoop 根目录和 lib 目录下的所有 jar 包导入。

5.3 创建 Mapper 或者 Reducer

File -> New -> Mapper 创建 Mapper，自动继承 mapred 包里面的 MapReduceBase 并实现 Mapper 接口。注意：这个插件自动继承的是 mapred 包里旧版的类和接口，新版的 Mapper 得自己写。Reducer 同理。

6、实例：在 eclipse 中写一个 WordCount 程序并扩展、运行、调试、部署

7、实例：在 eclipse 中演示如果通过程序操作 HDFS

8、用 python 编写 MapReduce

vi mapper.py

输入：

```
#!/usr/bin/env python
```

```
import sys

for line in sys.stdin:
        line = line.strip()
        words = line.split()
        for word in words:
                print '%s\t%s' % (word,1)
```

chmod +x mapper.py

vi reducer.py

输入：

```
#!/usr/bin/env python

from operator import itemgetter

import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
        line = line.strip()
        word, count = line.split('\t', 1)
        try:
                count = int(count)
        except ValueError:
                continue
        if current_word == word:
                current_count += count
        else:
                if current_word:
                        print '%s\t%s' % (current_word, current_count)
                current_count = count
                current_word = word
if current_word:
    print '%s\t%s' % (current_word, current_count)
```

chmod +x reducer.py

本地操作系统测试：

echo "foo foo quux labs foo bar quux" | ./mapper.py | sort | ./reducer.py

提交 HADOOP 集群运行：

hadoop jar hadoop-1.2.1/contrib/streaming/hadoop-streaming-1.2.1.jar -input input -output output-streaming-python -mapper /home/ysc/mapper.py -reducer /home/ysc/reducer.py

第三讲 Hive – 基于 HADOOP 的数据仓库

wget http://mirror.bit.edu.cn/apache/hive/hive-0.11.0/hive-0.11.0-bin.tar.gz

tar -xzvf hive-0.11.0-bin.tar.gz

cd hive-0.11.0-bin

sudo vi /etc/profile

增加：

export HIVE_HOME=/home/ysc/hive-0.10.0-bin

export PATH=$PATH:$HIVE_HOME/bin

source /etc/profile

hadoop fs -mkdir /tmp

hadoop fs -mkdir /user/hive/warehouse

hadoop fs -chmod g+w /tmp

hadoop fs -chmod g+w /user/hive/warehouse

cp conf/hive-log4j.properties.template conf/hive-log4j.properties

如使用 local 模式：SET mapred.job.tracker=local;

使用 HADOOP 集群（默认）：SET mapred.job.tracker=host001:9001;

本地使用 hive 服务：

hive**(如 出 现 错 误 ： Missing Hive Builtins Jar: /home/ysc/hive-0.11.0-bin/lib/hive-builtins-*.jar，则需要重启 sudo reboot)**

命令行执行 HiveQL 命令：创建表、准备文本数据、导入、查询

创建 hive 表：

create table demo (key int, value string)row format delimited fields terminated by '=' stored as textfile;

加载数据到 demo 表：

load data local inpath '/home/ysc/hive-0.11.0-bin/data.txt' into table demo;

查询：

select * from demo;

select * from demo where key>=100 and key<=120;

select *,count(*) as fre from demo group by value order by fre desc;

配置 Metastore 使用 MySQL

sudo apt-get install mysql-server mysql-client

GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;

sudo vi /etc/mysql/my.cnf

注释 bind-address = 127.0.0.1

sudo service mysql restart

mysql -uroot -pysc

vi conf/hive-site.xml

内容为：

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <!-- 使用 mysql -->
```

```
    <property>
      <name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://host001:3306/hive?createDatabaseIfNotExist=true</value>
    </property>
    <property>
      <name>javax.jdo.option.ConnectionDriverName</name>
      <value>com.mysql.jdbc.Driver</value>
    </property>
    <property>
      <name>javax.jdo.option.ConnectionUserName</name>
      <value>root</value>
    </property>
    <property>
      <name>javax.jdo.option.ConnectionPassword</name>
      <value>ysc</value>
    </property>
    <!-- 使用 hwi -->
    <property>
      <name>hive.hwi.listen.host</name>
      <value>0.0.0.0</value>
    </property>
    <property>
```

```
        <name>hive.hwi.listen.port</name>
        <value>9999</value>
    </property>
    <property>
        <name>hive.hwi.war.file</name>
        <value>lib/hive-hwi-0.11.0.war</value>
    </property>
    <!-- 使用 metastore -->
    <property>
        <name>hive.metastore.uris</name>
        <value>thrift://host001:9083</value>
    </property>
</configuration>
```

将 mysql-connector-java-5.1.18.jar 放置到 hive-0.10.0-bin/lib 目录

启动独立 Metastore 服务

```
hive --service metastore    &
```

启动独立 Hive server 服务

```
hive --service hiveserver &
```

远程使用 **hive** 服务

```
hive -h host001 -p 10000
```

启动 Hive Web Interface(HWI)服务

hive --service hwi &

http://host001:9999/hwi/

Hive JDBC 编程

把 hadoop-core-1.1.2.jar 以及 HIVE_HOME/lib/*.jar 加入构建路径

```
public static void main(String[] args) throws Exception {
    Class.forName("org.apache.hadoop.hive.jdbc.HiveDriver");
    Connection con =
DriverManager.getConnection("jdbc:hive://host001:10000/default");
    String sql = "select * from person";
    PreparedStatement pst = con.prepareStatement(sql);
    ResultSet rs = pst.executeQuery();
    while(rs.next()){
        System.out.println(rs.getString(1)+" "+rs.getString(2));
    }
}
```

Hcatalog

sudo vi /etc/profile

增加：

export HADOOP_HOME=/home/ysc/hadoop-1.2.1

export HCAT_HOME=/home/ysc/hive-0.11.0-bin/hcatalog

export HCAT_PREFIX=$HCAT_HOME

export METASTORE_PORT=9083

export HCAT_LOG_DIR=/home/ysc/hive-0.11.0-bin/hcatalog/logs

export PATH=$PATH:$HCAT_HOME/bin:$HCAT_HOME/sbin

source /etc/profile

mkdir /home/ysc/hive-0.11.0-bin/hcatalog/logs

chmod +x /home/ysc/hive-0.11.0-bin/hcatalog/bin/hcat

chmod +x /home/ysc/hive-0.11.0-bin/hcatalog/sbin/*.sh

hcat -e "create table test(id int, value string)"

hcat -e "drop table test"

hcat -e "show tables"

hcat -e "desc test"

hcat_server.sh start & (注意不要启动后面的命令：hive --service metastore &)

hcat_server.sh stop

WebHCat(HCatalog REST API)

sudo vi /etc/profile

增加：

export HADOOP_CONF_DIR=$HADOOP_HOME/conf

export HADOOP_PREFIX=$HADOOP_HOME

export TEMPLETON_HOME=/home/ysc/hive-0.11.0-bin/hcatalog

source /etc/profile

hadoop fs -put /home/ysc/hadoop-1.2.1/contrib/streaming/hadoop-streaming-1.2.1.jar /apps/templeton/hadoop-streaming-1.2.1.jar
hadoop fs -put /home/ysc/pig-0.11.1.tar.gz /apps/templeton/pig-0.11.1.tar.gz
hadoop fs -put /home/ysc/hive-0.11.0-bin.tar.gz /apps/templeton/hive-0.11.0-bin.tar.gz

hadoop fs -ls /apps/templeton

vi /home/ysc/hive-0.11.0-bin/hcatalog/etc/webhcat/webhcat-site.xml

输入：

<?xml version="1.0" encoding="UTF-8"?>

```
<configuration>
    <property>
      <name>templeton.streaming.jar</name>
      <value>hdfs:///apps/templeton/hadoop-streaming-1.2.1.jar</value>
    </property>
    <property>
      <name>templeton.pig.archive</name>
      <value>hdfs:///apps/templeton/pig-0.11.1.tar.gz</value>
    </property>
    <property>
      <name>templeton.pig.path</name>
      <value>pig-0.11.1.tar.gz/pig-0.11.1/bin/pig</value>
    </property>
    <property>
      <name>templeton.hive.archive</name>
      <value>hdfs:///apps/templeton/hive-0.11.0-bin.tar.gz</value>
    </property>
    <property>
      <name>templeton.hive.path</name>
      <value>hive-0.11.0-bin.tar.gz/hive-0.11.0-bin/hive</value>
    </property>
    <property>
      <name>templeton.jar</name>
      <value>${env.TEMPLETON_HOME}/share/webhcat/svr/webhcat-0.11.0.jar</value>
    </property>
    <property>
      <name>templeton.hive.properties</name>
      <value>hive.metastore.local=false,hive.metastore.uris=thrift://host001:9083,hive.met
      astore.sasl.enabled=false</value>
    </property>
</configuration>
```

```
webhcat_server.sh start &
```

```
webhcat_server.sh stop
```

```
sudo apt-get install curl
```

```
curl -i 'http://host001:50111/templeton/v1/status'
```

```
curl                                                                    -i
'http://host001:50111/templeton/v1/ddl/database/default/table/test?user.name
```

=root'

curl -i -d user.name=root \

-d rename=test2 \

'http://localhost:50111/templeton/v1/ddl/database/default/table/test'

Hive 命令：

hive -e 'select * from demo'

hive -e 'select * from demo where key < 5'

HiveServer2：

sudo vi /etc/profile

增加：

export HIVE_SERVER2_THRIFT_BIND_HOST=host001

export HIVE_SERVER2_THRIFT_PORT=10002

source /etc/profile

hadoop fs -chmod -R 777 /tmp

启动服务：hiveserver2 & 或者 hive --service hiveserver2 &

连接服务：beeline

beeline>!connect jdbc:hive2://host001:10002 root ysc org.apache.hive.jdbc.HiveDriver

0: jdbc:hive2://host001:10002>show tables;

0: jdbc:hive2://host001:10002>select * from students;

当然也可以用 JAVA 借助 JDBC 调用

第四讲 Pig – 大数据分析平台

wget http://mirror.bit.edu.cn/apache/pig/pig-0.11.1/pig-0.11.1.tar.gz

tar -xzvf pig-0.11.1.tar.gz

sudo vi /etc/profile

增加：

export PIG_HOME=/home/ysc/pig-0.11.1

export PATH=$PATH:$PIG_HOME/bin

source /etc/profile

cp conf/log4j.properties.template conf/log4j.properties

pig --help

Local Mode：

1、pig -x local

2、java -cp /home/ysc/pig-0.11.1/pig-0.11.1.jar org.apache.pig.Main -x local

Mapreduce Mode（Default）：

1、pig

2、pig -x mapreduce

3、java -cp /home/ysc/pig-0.11.1/pig-0.11.1.jar:/home/ysc/hadoop-1.2.1/conf org.apache.pig.Main

4、java -cp /home/ysc/pig-0.11.1/pig-0.11.1.jar:/home/ysc/hadoop-1.2.1/conf org.apache.pig.Main -x mapreduce

准备数据：

hadoop fs -put /etc/passwd passwd

Interactive Mode:

进入 Pig shell（Local 或 Mapreduce Mode）:

pig（pig -x local）

```
grunt> A = load 'passwd' using PigStorage(':');
grunt> B = foreach A generate $0 as id;
grunt> dump B;
```

Batch Mode:

编写脚本：

vi id.pig

输入：

```
/* id.pig */
-- load the passwd file
A = load 'passwd' using PigStorage(':');
-- extract the user IDs
B = foreach A generate $0 as id;
-- write the results to a file name id.out
store B into 'id.out';
```

运行脚本（Local 或 Mapreduce Mode）:

pig（pig -x local） id.pig

查看结果：

hadoop fs -cat id.out/part-m-00000

Pig 使用 HCatalog 管理数据：

启动 Metastore

hcat_server.sh start & (或： hive --service metastore &)

sudo vi /etc/profile

增加：

```
export PIG_CLASSPATH=$HCAT_HOME/share/hcatalog/hcatalog-*.jar:\
$HIVE_HOME/lib/hive-metastore-*.jar:$HIVE_HOME/lib/libthrift-*.jar:\
$HIVE_HOME/lib/hive-exec-*.jar:$HIVE_HOME/lib/libfb303-*.jar:\
$HIVE_HOME/lib/jdo2-api-*-ec.jar:$HIVE_HOME/lib/slf4j-api-*.jar
export PIG_OPTS=-Dhive.metastore.uris=thrift://host001:9083
```

source /etc/profile

创建表：

```
hcat -e "CREATE TABLE students (name STRING, age INT) ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' STORED AS TEXTFILE; "
```

准备数据：

vi students.txt

输入：

```
刘德华 51
张学友 52
刘亦菲 41
杨尚川 27
成龙 55
洪金宝 52
林志玲 40
hadoop fs -put students.txt /user/ysc/students.txt
```

启动 pig：

pig -Dpig.additional.jars=$PIG_CLASSPATH

存储数据:

```
students = LOAD '/user/ysc/students.txt' AS (name:chararray, age:int);
dump students;
STORE students INTO 'students' USING org.apache.hcatalog.pig.HCatStorer();
```

加载数据：

```
A = LOAD 'students' USING org.apache.hcatalog.pig.HCatLoader();
dump A;
```

第五讲 HBase – 基于 Hadoop 的分布式数据库

wget http://mirrors.cnnic.cn/apache/zookeeper/zookeeper-3.4.5/zookeeper-3.4.5.tar.gz
tar -xzvf zookeeper-3.4.5.tar.gz
cd zookeeper-3.4.5
cp conf/zoo_sample.cfg conf/zoo.cfg
vi conf/zoo.cfg
修改：dataDir=/home/ysc/zookeeper
添加：
server.1=host001:2888:3888
maxClientCnxns=100
mkdir /home/ysc/zookeeper（注：dataDir 是 zookeeper 的数据目录，需要手动创建）
echo 1 > /home/ysc/zookeeper/myid
启动服务：
bin/zkServer.sh start
连接服务：
bin/zkCli.sh -server host001:2181
查看服务状态：
bin/zkServer.sh status

hbase 存在系统时间同步的问题，并且误差要再 30s 以内
sudo apt-get install ntp
sudo ntpdate -u 210.72.145.44
HBase 是数据库，会在同一时间使用很多的文件句柄，大多数 linux 系统使用的默认值 1024 是不能满足的，还需要修改 hbase 用户的 nproc，在压力很大的情况下，如果过低会造成 OutOfMemoryError 异常
sudo vi /etc/security/limits.conf
添加：
ysc soft nproc 32000
ysc hard nproc 32000
ysc soft nofile 32768
ysc hard nofile 32768
sudo vi /etc/pam.d/common-session
添加：
session required pam_limits.so

wget http://mirrors.hust.edu.cn/apache/hbase/hbase-0.94.12/hbase-0.94.12.tar.gz
tar -xzvf hbase-0.94.12.tar.gz
cd hbase-0.94.12
vi conf/hbase-env.sh
追加：
export JAVA_HOME=/home/ysc/jdk1.7.0_40
export HBASE_MANAGES_ZK=false

```
export HBASE_HEAPSIZE=1000
```

vi conf/hbase-site.xml

加入：

```
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://host001:9000/hbase</value>
</property>
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>host001</value>
</property>
```

vi conf/regionservers

改 localhost 为 host001

因为 HBase 建立在 Hadoop 之上，Hadoop 使用的 hadoop*.jar 和 HBase 使用的必须一致。所以要将 HBase lib 目录下的 hadoop*.jar 替换成 Hadoop 里面的那个，防止版本冲突。

```
cp  /home/ysc/hadoop-1.2.1/hadoop-core-1.2.1.jar  /home/ysc/hbase-0.94.12/lib
rm  /home/ysc/hbase-0.94.12/lib/hadoop-core-1.0.4.jar
```

启动 hadoop 并创建目录

```
hadoop fs -mkdir /hbase
```

```
sudo vi /etc/profile
```

增加：

```
export PATH=$PATH:/home/ysc/hbase-0.94.12/bin
```

```
source /etc/profile
```

启动初始 HBase 集群：

```
start-hbase.sh
```

web 界面

http://host001:60010

http://host001:60030

停止 HBase 集群：

```
stop-hbase.sh
```

启动额外备份主服务器，可以启动到 9 个备份服务器 (总数 10 个)：

```
local-master-backup.sh start 1
local-master-backup.sh start 2 3
```

启动更多 regionservers，支持到 99 个额外 regionservers (总 100 个)：

```
local-regionservers.sh start 1
local-regionservers.sh start 2 3 4 5
```

停止备份主服务器:

```
cat /tmp/hbase-ysc-1-master.pid | xargs kill -9
```

停止单独 regionserver：

```
local-regionservers.sh stop 1
```

使用 HBase 命令行模式:

hbase shell

命令演示：

创建表 Person，列族 basic 和 detail

```
create 'Person', 'basic', 'detail'
```

看有哪些表：

```
list
```

看是否有 person 表：

```
list 'person'
```

增加数据，动态增加列：

数据 1：

```
put 'Person', '533001198510125838', 'basic:idcard', '533001198510125838'
put 'Person', '533001198510125838', 'basic:name', '章子怡'
put 'Person', '533001198510125838', 'detail:age', '25'
```

数据 2：

```
put 'Person', '533001198510125837', 'basic:idcard', '533001198510125837'
put 'Person', '533001198510125837', 'basic:name', '杨尚川'
put 'Person', '533001198510125837', 'detail:age', '22'
```

查询整条数据：

```
get 'Person', '533001198510125838'
get 'Person', '533001198510125837'
```

查询单列数据：

```
get 'Person', '533001198510125837', 'basic:idcard'
```

查询多列数据：

```
get 'Person', '533001198510125837', 'basic:idcard', 'detail:age'
```

查询全部数据：

```
scan 'Person'
```

删除表：

```
disable 'Person'
drop 'Person'
```

第六讲 Storm – 流计算

wget http://download.zeromq.org/zeromq-2.1.7.tar.gz
tar -xzvf zeromq-2.1.7.tar.gz
cd zeromq-2.1.7
sudo apt-get install gcc
sudo apt-get install g++
sudo apt-get install libuuid-dev
./configure
sudo apt-get install make
make
sudo make install

sudo apt-get install git
git clone https://github.com/nathanmarz/jzmq.git
cd jzmq
sudo apt-get install pkg-config
./autogen.sh
./configure
touch src/classdist_noinst.stamp
cd src
javac -d . org/zeromq/*.java
cd ..
make
sudo make install

wget https://dl.dropboxusercontent.com/s/fl4kr7w0oc8ihdw/storm-0.8.2.zip
sudo apt-get install unzip
unzip storm-0.8.2.zip
cd storm-0.8.2
sudo vi /etc/profile
增加：
  export STORM_HOME=/home/ysc/storm-0.8.2
  export PATH=$PATH:$STORM_HOME/bin
source /etc/profile
vi conf/storm.yaml
增加：
```
storm.zookeeper.servers:
        - "host001"
nimbus.host: "host001"
storm.local.dir: "/home/ysc/storm"
supervisor.slots.ports:
      - 6700
```

- 6701
- 6702
- 6703

启动主节点

storm nimbus &

启动从节点

storm supervisor &

启动 WEB 服务

storm ui &

访问：http://host001:8080

git clone https://github.com/nathanmarz/storm-starter.git

cd storm-starter

sudo apt-get install maven2

vi m2-pom.xml

改变下面两个依赖的版本为 3.0.3

```
<dependency>
        <groupId>org.twitter4j</groupId>
        <artifactId>twitter4j-core</artifactId>
        <version>3.0.3</version>
</dependency>
<dependency>
        <groupId>org.twitter4j</groupId>
        <artifactId>twitter4j-stream</artifactId>
        <version>3.0.3</version>
</dependency>
```

mvn -f m2-pom.xml package

cp m2-pom.xml pom.xml

mvn eclipse:eclipse

导入 eclipse

Java Build Path -> Add Variable… -> Configure Variables… -> New… -> Name: M2_REPO Path:maven 存储库路径

本地运行 wordcount:

storm jar target/storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar storm.starter.WordCountTopology

集群运行 wordcount:

storm jar target/storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar storm.starter.WordCountTopology wordcount

第七讲 Sqoop – HADOOP 和 RDBMS 数据交换

Sqoop 1：

wget http://mirrors.ustc.edu.cn/apache/sqoop/1.4.4/sqoop-1.4.4.bin__hadoop-1.0.0.tar.gz

tar -xzvf sqoop-1.4.4.bin__hadoop-1.0.0.tar.gz

mv sqoop-1.4.4.bin__hadoop-1.0.0 sqoop-1.4.4

cd sqoop-1.4.4

sudo vi /etc/profile

增加：

```
export HADOOP_COMMON_HOME=/home/ysc/hadoop-1.2.1
export HADOOP_MAPRED_HOME=/home/ysc/hadoop-1.2.1
export PATH=$PATH:/home/ysc/sqoop-1.4.4/bin
export HBASE_HOME=/home/ysc/hbase-0.94.12
```

source /etc/profile

sqoop help

将 JDBC 驱动 mysql-connector-java-5.1.18.jar 拷贝到/home/ysc/sqoop-1.4.4/lib

```
sqoop list-databases --connect jdbc:mysql://host001 --username root --password ysc
sqoop list-tables --connect jdbc:mysql://host001/mysql --username root --password ysc
sqoop import --connect jdbc:mysql://host001/test --username root --password ysc --table person
sqoop import --connect jdbc:mysql://host001/test --username root --password ysc --table person -m 1
sqoop import --connect jdbc:mysql://host001/test --username root --password ysc --table person --direct -m 1
sqoop import-all-tables --connect jdbc:mysql://host001/test --username root --password ysc --direct -m 1
sqoop export --connect jdbc:mysql://host001/test --username root --password ysc --table person --export-dir person
```

sqoop export --connect jdbc:mysql://host001/test --username root --password ysc --table animal --export-dir animal

Sqoop 2 :

wget http://mirror.bit.edu.cn/apache/sqoop/1.99.2/sqoop-1.99.2-bin-hadoop100.tar.gz

tar -xzvf sqoop-1.99.2-bin-hadoop100.tar.gz

mv sqoop-1.99.2-bin-hadoop100 sqoop-1.99.2

cd sqoop-1.99.2

sudo apt-get install zip

bin/addtowar.sh -hadoop-version 1.2.1 -hadoop-path /home/ysc/hadoop-1.2.1

bin/addtowar.sh -jars /home/ysc/mysql-connector-java-5.1.18.jar

vi server/conf/sqoop.properties

修改 org.apache.sqoop.submission.engine.mapreduce.configuration.directory=/etc/hadoop/conf/ 为 org.apache.sqoop.submission.engine.mapreduce.configuration.directory=/home/ysc/hadoop-1.2.1/conf/

启动 Sqoop 2 server：

bin/sqoop.sh server start

http://host001:12000/sqoop/

停止 Sqoop 2 server：

bin/sqoop.sh server stop

客户端连接 Sqoop 2 server：

客户端直接解压即可运行

MySQL 准备数据库和表:

create database test;

create table history (userId int, command varchar(20));

insert into history values(1, 'ls');

insert into history values(1, 'dir');

insert into history values(2, 'cat');

insert into history values(5, 'vi');

交互模式：

bin/sqoop.sh client

sqoop:000> set server --host host001 --port 12000 --webapp sqoop

sqoop:000> show version --all

sqoop:000> show connector --all

sqoop:000>create connection --cid 1

Name: mysql

JDBC Driver Class: com.mysql.jdbc.Driver

JDBC Connection String:

jdbc:mysql://host001:3306/test?useUnicode=true&characterEncoding=UTF-8&createDatabaseIfNotExist=true&autoReconnect=true

Username: root

Password: ***

entry#回车

Max connections:100

sqoop:000>create job --xid 1 --type import

Name: ImportHistory

Schema name:

Table name: history

Table SQL statement:

Table column names:

Partition column name:userId

Boundary query:

Choose: 0

Choose: 0

Output directory: output-sqoop2-history

Extractors:

Loaders:

sqoop:000> submission start --jid 1

sqoop:000> submission status --jid 1

sqoop:000> submission stop --jid 1

批处理模式：

sqoop.sh client /home/ysc/script.sqoop

vi /home/ysc/script.sqoop

输入：

```
# 指定服务器信息
set server --host host001 --port 12000 --webapp sqoop
# 执行 JOB
submission start --jid 1
```

第八讲 Mahout – 机器学习

wget http://mirrors.ustc.edu.cn/apache/mahout/0.8/mahout-distribution-0.8.tar.gz

tar -xzvf mahout-distribution-0.8.tar.gz

cd mahout-distribution-0.8

sudo vi /etc/profile

增加：

export PATH=$PATH:/home/ysc/mahout-distribution-0.8/bin

source /etc/profile

数据：hadoop fs -put clustering_material.txt testdata/clustering_material.txt

聚类 1：mahout org.apache.mahout.clustering.syntheticcontrol.kmeans.Job

分析 1：mahout clusterdump --input output/clusters-10-final --pointsDir output/clusteredPoints --output output/clusteranalyze_kmeans.txt

查看：

hadoop fs -lsr output

more output/clusteranalyze_kmeans.txt

VL-19 代表这是一个 cluster，n=161 代表该 cluster 有 161 个点，c=[...] 代表该 cluster 的中心向量点，*r=[...]代表 cluster 的半径*

聚类 2：mahout org.apache.mahout.clustering.syntheticcontrol.canopy.Job

分析 2：mahout clusterdump --input output/clusters-0-final --pointsDir output/clusteredPoints --output output/clusteranalyze_canopy.txt

聚类 3：mahout org.apache.mahout.clustering.syntheticcontrol.fuzzykmeans.Job

分析 3：mahout clusterdump --input output/clusters-6-final --pointsDir output/clusteredPoints --output output/clusteranalyze_fuzzykmeans.txt

聚类 4：mahout org.apache.mahout.clustering.syntheticcontrol.dirichlet.Job

分析 4：mahout clusterdump --input output/clusters-5-final --pointsDir output/clusteredPoints --output output/clusteranalyze_dirichlet.txt

聚类 5：mahout org.apache.mahout.clustering.syntheticcontrol.meanshift.Job

分析 5：mahout clusterdump --input output/clusters-3-final --pointsDir output/clusteredPoints --output output/clusteranalyze_meanshift.txt

第九讲 Spark – 内存计算

wget

http://mirrors.cnnic.cn/apache/incubator/spark/spark-0.8.0-incubating/spark-0.8.0-incubating-bin-hadoop1.tgz

tar -zxvf spark-0.8.0-incubating-bin-hadoop1.tgz

mv spark-0.8.0-incubating-bin-hadoop1 spark-0.8.0

wget http://www.scala-lang.org/files/archive/scala-2.9.3.tgz

tar -zxvf scala-2.9.3.tgz

sudo vi /etc/profile

增加:

export SCALA_HOME=/home/ysc/scala-2.9.3

export PATH=$PATH:$SCALA_HOME/bin

source /etc/profile

cd spark-0.8.0（spark 命令和 hadoop 命令重名，不加入 path）

cp conf/spark-env.sh.template conf/spark-env.sh

vi conf/slaves

修改 localhost 为 host001

vi conf/spark-env.sh

增加:

JAVA_HOME=/home/ysc/jdk1.7.0_40

SCALA_HOME=/home/ysc/scala-2.9.3

SPARK_WORKER_INSTANCES=2

启动服务：

bin/start-all.sh

WEB 界面：

Spark Master ： http://host001:8080/

Spark Worker ： http://host001:8081/

运行例子：

集群运算：

./run-example org.apache.spark.examples.JavaSparkPi spark://host001:7077

./run-example org.apache.spark.examples.JavaWordCount spark://host001:7077 README.md

本地运算：

./run-example org.apache.spark.examples.JavaSparkPi local[4]

（4 代表线程数目）

./run-example org.apache.spark.examples.JavaWordCount local[4] README.md

停止服务：

bin/stop-all.sh

第十讲 Gora – 大数据持久化

wget http://mirrors.cnnic.cn/apache/gora/0.3/apache-gora-0.3-src.zip

unzip apache-gora-0.3-src.zip

cd apache-gora-0.3

mvn clean package

1、创建项目

mvn archetype:create -DgroupId=org.apdplat.demo.gora -DartifactId=gora-demo

2、增加依赖

vi gora-demo/pom.xml

在<dependencies>标签内增加：

```
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-core</artifactId>
    <version>1.2.1</version>
</dependency>
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase</artifactId>
    <version>0.94.12</version>
</dependency>
<dependency>
    <groupId>org.apache.gora</groupId>
    <artifactId>gora-core</artifactId>
    <version>0.3</version>
        <exclusions>
                <exclusion>
                        <groupId>org.apache.hadoop</groupId>
                        <artifactId>hadoop-core</artifactId>
                </exclusion>
                <exclusion>
                        <groupId>org.apache.cxf</groupId>
                        <artifactId>cxf-rt-frontend-jaxrs</artifactId>
                </exclusion>
```

```
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.apache.gora</groupId>
            <artifactId>gora-hbase</artifactId>
            <version>0.3</version>
                <exclusions>
                        <exclusion>
                                <groupId>org.apache.hbase</groupId>
                                <artifactId>hbase</artifactId>
                        </exclusion>
                        <exclusion>
                                <groupId>org.apache.hadoop</groupId>
                                <artifactId>hadoop-test</artifactId>
                        </exclusion>
                </exclusions>
        </dependency>
```

3、数据建模

mkdir -p gora-demo/src/main/avro

vi gora-demo/src/main/avro/person.json

输入：

```
{
  "type": "record",
  "name": "Person",
  "namespace": "org.apdplat.demo.gora.generated",
  "fields" : [
    {"name": "idcard", "type": "string"},
    {"name": "name", "type": "string"},
    {"name": "age", "type": "string"}
  ]
}
```

4、生成 JAVA 类

bin/gora goracompiler gora-demo/src/main/avro/person.json gora-demo/src/main/java/

5、模型映射

mkdir -p gora-demo/src/main/resources/

vi gora-demo/src/main/resources/gora-hbase-mapping.xml

输入：

```
<gora-orm>
  <table name="Person">
    <family name="basic"/>
    <family name="detail"/>
  </table>
  <class table="Person" name="org.apdplat.demo.gora.generated.Person" keyClass="java.lang.String">
    <field name="idcard" family="basic" qualifier="idcard"/>
    <field name="name" family="basic" qualifier="name"/>
    <field name="age" family="detail" qualifier="age"/>
  </class>
</gora-orm>
```

6、Gora 配置

vi gora-demo/src/main/resources/gora.properties

输入：

```
gora.datastore.default=org.apache.gora.hbase.store.HBaseStore
gora.datastore.autocreateschema=true
```

7、Hbase 配置

vi gora-demo/src/main/resources/hbase-site.xml

输入：

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
  <property>
    <name>hbase.zookeeper.property.clientPort</name>
    <value>2181</value>
  </property>
  <property>
    <name>hbase.zookeeper.quorum</name>
    <value>host001</value>
  </property>
</configuration>
```

8、编写 PersonManager.java 和 PersonAnalytics.java

vi gora-demo/src/main/java/org/apdplat/demo/gora/PersonManager.java

输入：

```
package org.apdplat.demo.gora;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.text.ParseException;
import org.apache.avro.util.Utf8;
import org.apache.gora.query.Query;
import org.apache.gora.query.Result;
import org.apache.gora.store.DataStore;
import org.apache.gora.store.DataStoreFactory;
import org.apache.hadoop.conf.Configuration;
import org.apdplat.demo.gora.generated.Person;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class PersonManager {
    private static final Logger log = LoggerFactory.getLogger(PersonManager.class);
    private DataStore<String, Person> dataStore;
    public PersonManager() {
      try {
        init();
      } catch (IOException ex) {
        throw new RuntimeException(ex);
      }
    }
    private void init() throws IOException {
       Configuration  conf = new Configuration();
      dataStore = DataStoreFactory.getDataStore(String.class, Person.class, conf);
    }
    private void parse(String input) throws IOException, ParseException, Exception {
      log.info("解析文件:" + input);
      BufferedReader reader = new BufferedReader(new FileReader(input));
      long lineCount = 0;
      try {
        String line = reader.readLine();
        do {
          Person person = parseLine(line);

          if(person != null) {
            //入库
```

```
                storePerson(person.getIdcard().toString(), person);
              }
              lineCount++;
              line = reader.readLine();
            } while(line != null);

          } finally {
            reader.close();
          }
          Log.info("文件解析完毕. 总人数:" + lineCount);
        }
        private Person parseLine(String line) throws ParseException {
          String[] attrs = line.split(" ");
          String idcard = attrs[0];
          String name = attrs[1];
          String age = attrs[2];

          Person person = new Person();
          person.setIdcard(new Utf8(idcard));
          person.setName(new Utf8(name));
          person.setAge(new Utf8(age));

          return person;
        }
        private void storePerson(String key, Person person) throws IOException, Exception {
           Log.info("保存人员信息: " +
person.getIdcard()+"\t"+person.getName()+"\t"+person.getAge());
          dataStore.put(key, person);
        }
        private void get(String key) throws IOException, Exception {
          Person person = dataStore.get(key);
          printPerson(person);
        }
        private void query(String key) throws IOException, Exception {
          Query<String, Person> query = dataStore.newQuery();
          query.setKey(key);

          Result<String, Person> result = query.execute();

          printResult(result);
        }
        private void query(String startKey, String endKey) throws IOException, Exception {
          Query<String, Person> query = dataStore.newQuery();
          query.setStartKey(startKey);
```

```
    query.setEndKey(endKey);

    Result<String, Person> result = query.execute();

    printResult(result);
  }
  private void delete(String key) throws Exception {
    dataStore.delete(key);
    dataStore.flush();
    log.info("身份证号码为:" + key + " 的人员信息被删除");
  }
  private void deleteByQuery(String startKey, String endKey) throws IOException, Exception {
    Query<String, Person> query = dataStore.newQuery();
    query.setStartKey(startKey);
    query.setEndKey(endKey);

    dataStore.deleteByQuery(query);
    log.info("身份证号码从 " + startKey + " 到 " + endKey + " 的人员信息被删除");
  }
  private void printResult(Result<String, Person> result) throws IOException, Exception {
    while(result.next()) {
     String resultKey = result.getKey();
     Person resultPerson = result.get();

     System.out.println(resultKey + ":");
     printPerson(resultPerson);
    }

    System.out.println("人数:" + result.getOffset());
  }
  private void printPerson(Person person) {
    if(person == null) {
      System.out.println("没有结果");
    } else {
      System.out.println(person.getIdcard()+"\t"+person.getName()+"\t"+person.getAge());
    }
  }
  private void close() throws IOException, Exception {
    if(dataStore != null)
      dataStore.close();
  }
  private static final String USAGE = "PersonManager -parse <input_person_file>\n" +
                                      "          -get <idcard>\n" +
                                      "          -query <idcard>\n" +
```

```
                                        "          -query <startIdcard> <endIdcard>\n" +
                                         "          -delete <idcard>\n" +
                                         "          -deleteByQuery <startIdcard> <endIdcard>\n";

    public static void main(String[] args) throws Exception {
      if(args.length < 2) {
        System.err.println(USAGE);
        System.exit(1);
      }

      PersonManager manager = new PersonManager();

      if("-parse".equals(args[0])) {
        manager.parse(args[1]);
      } else if("-get".equals(args[0])) {
        manager.get(args[1]);
      } else if("-query".equals(args[0])) {
        if(args.length == 2)
          manager.query(args[1]);
        else
          manager.query(args[1], args[2]);
      } else if("-delete".equals(args[0])) {
        manager.delete(args[1]);
      } else if("-deleteByQuery".equalsIgnoreCase(args[0])) {
        manager.deleteByQuery(args[1], args[2]);
      } else {
        System.err.println(USAGE);
        System.exit(1);
      }

      manager.close();
    }
    }
```

vi gora-demo/src/main/java/org/apdplat/demo/gora/PersonAnalytics.java

输入：

```
package org.apdplat.demo.gora;

import java.io.IOException;

import org.apache.avro.util.Utf8;
import org.apache.gora.mapreduce.GoraMapper;
import org.apache.gora.store.DataStore;
import org.apache.gora.store.DataStoreFactory;
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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.Reducer;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apdplat.demo.gora.generated.Person;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class PersonAnalytics extends Configured implements Tool {
    private static final Logger log = LoggerFactory
            .getLogger(PersonAnalytics.class);

    public static class PersonAnalyticsMapper extends
            GoraMapper<String, Person, Text, LongWritable> {
        private LongWritable one = new LongWritable(1L);

        @Override
        protected void map(String key, Person person, Context context)
                throws IOException, InterruptedException {
            Utf8 age = person.getAge();
            context.write(new Text(age.toString()), one);
        };
    }

    public static class PersonAnalyticsReducer extends
            Reducer<Text, LongWritable, Text, LongWritable> {
        @Override
        protected void reduce(Text key, Iterable<LongWritable> values,
                Context context) throws IOException, InterruptedException {
            long sum = 0L;
            for (LongWritable value : values) {
                sum += value.get();
            }
            context.write(key, new LongWritable(sum));
        };
    }

    public Job createJob(DataStore<String, Person> inStore, int numReducer)
```

```
            throws IOException {
        Job job = new Job(getConf());
        job.setJobName("Person Analytics");
        Log.info("Creating Hadoop Job: " + job.getJobName());
        job.setNumReduceTasks(numReducer);
        job.setJarByClass(getClass());
        GoraMapper.initMapperJob(job, inStore, Text.class, LongWritable.class,
                PersonAnalyticsMapper.class, true);
        job.setReducerClass(PersonAnalyticsReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);
        TextOutputFormat
                .setOutputPath(job, new Path("person-analytics-output"));
        return job;
    }

    @Override
    public int run(String[] args) throws Exception {
        DataStore<String, Person> inStore;
        Configuration conf = new Configuration();
        if (args.length == 1) {
            String dataStoreClass = args[0];
            inStore = DataStoreFactory.getDataStore(dataStoreClass,
                    String.class, Person.class, conf);
        } else {
            inStore = DataStoreFactory.getDataStore(String.class, Person.class,
                    conf);
        }
        Job job = createJob(inStore, 2);
        boolean success = job.waitForCompletion(true);
        inStore.close();
        Log.info("PersonAnalytics completed with "
                + (success ? "success" : "failure"));
        return success ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int ret = ToolRunner.run(new PersonAnalytics(), args);
        System.exit(ret);
    }
}
```

9、准备数据

vi gora-demo/src/main/resources/persons.txt

输入：

```
533001198510125839 杨尚川 25
533001198510125840 杨尚华 22
533001198510125841 刘德华 55
533001198510125842 刘亦菲 25
533001198510125843 蔡卓妍 25
533001198510125844 林志玲 22
533001198510125845 李连杰 55
```

10、在 Linux 命令行使用 maven2 编译运行项目

cd gora-demo

mvn clean compile

mvn exec:java -Dexec.mainClass=org.apdplat.demo.gora.PersonManager

mvn exec:java -Dexec.mainClass="org.apdplat.demo.gora.PersonManager" -Dexec.args="-parse src/main/resources/persons.txt"

mvn exec:java -Dexec.mainClass=org.apdplat.demo.gora.PersonAnalytics

cat person-analytics-output/part-r-00000

mvn exec:java -Dexec.mainClass="org.apdplat.demo.gora.PersonManager" -Dexec.args="-get 533001198510125842"

mvn exec:java -Dexec.mainClass="org.apdplat.demo.gora.PersonManager" -Dexec.args="-query 533001198510125844"

mvn exec:java -Dexec.mainClass="org.apdplat.demo.gora.PersonManager" -Dexec.args="-query 533001198510125842 533001198510125845"

mvn exec:java -Dexec.mainClass="org.apdplat.demo.gora.PersonManager" -Dexec.args="-delete 533001198510125840"

mvn exec:java -Dexec.mainClass="org.apdplat.demo.gora.PersonManager" -Dexec.args="-deleteByQuery 533001198510125841 533001198510125842"

mvn exec:java -Dexec.mainClass="org.apdplat.demo.gora.PersonManager" -Dexec.args="-deleteByQuery 533001198510125845 533001198510125846"

mvn exec:java -Dexec.mainClass="org.apdplat.demo.gora.PersonManager" -Dexec.args="-query 533001198510125838 533001198510125848"

11、在 windows 下使用 eclipse 编译运行项目

mvn clean package

rm -r target

vi .classpath

删除所有包含 path="M2_REPO 的行

删除 <classpathentry kind="src" path="target/maven-shared-archive-resources" excluding="**/*.java"/>

通过 WinSCP 把 gora-demo 传到 windows

从 http://yangshangchuan.iteye.com/blog/1839784 下载修改过的 hadoop-core-1.2.1.jar 替换文件 gora-demo\lib\hadoop-core-1.2.1.jar

将 gora-demo 导入 eclipse

将 lib 下的所有 jar 加入构建路径

12、打包项目并提交 Hadoop 运行

cd gora-demo

mvn clean package

mkdir job

cp -r lib job/lib

cp -r target/classes/* job

hadoop fs -put persons.txt persons.txt

jar -cvf gora-demo.job *

hadoop jar gora-demo.job org.apdplat.demo.gora.PersonAnalytics

第十一讲 Hadoop2 – 全新的 Hadoop

wget

http://mirrors.hust.edu.cn/apache/hadoop/common/stable2/hadoop-2.2.0.tar.gz

tar -xzvf hadoop-2.2.0.tar.gz

cd hadoop-2.2.0

vi etc/hadoop/hadoop-env.sh

修改：

export JAVA_HOME=/home/ysc/jdk1.7.0_17

vi etc/hadoop/slaves

修改 localhost 为 host001

vi etc/hadoop/core-site.xml

增加：

```
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://host001:9000</value>
</property>
```

cp etc/hadoop/mapred-site.xml.template etc/hadoop/mapred-site.xml

vi etc/hadoop/mapred-site.xml

增加：

```
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
<property>
    <name>mapreduce.cluster.local.dir</name>
    <value>/home/ysc/mapreduce/local</value>
```

```
</property>
```

vi etc/hadoop/yarn-site.xml

增加：

```
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>host001</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
```

vi etc/hadoop/hdfs-site.xml

增加：

```
<property>
    <name>dfs.name.dir</name>
    <value>/home/ysc/dfs/filesystem/name</value>
</property>
<property>
    <name>dfs.data.dir</name>
    <value>/home/ysc/dfs/filesystem/data</value>
</property>
<property>
    <name>dfs.replication</name>
    <value>3</value>
</property>
```

sudo vi /etc/profile

增加：

```
export   HADOOP_PREFIX=/home/ysc/hadoop-2.2.0
export   HADOOP_COMMON_HOME=/home/ysc/hadoop-2.2.0
export   HADOOP_MAPRED_HOME=/home/ysc/hadoop-2.2.0
export   HADOOP_CONF_DIR=/home/ysc/hadoop-2.2.0/etc/hadoop
export   HADOOP_HDFS_HOME=/home/ysc/hadoop-2.2.0
export   HADOOP_YARN_HOME=/home/ysc/hadoop-2.2.0
```

source /etc/profile

格式化：

bin/hdfs namenode -format

启动 dfs：

sbin/start-dfs.sh

启动 yarn：

sbin/start-yarn.sh

启动 historyserver：

sbin/mr-jobhistory-daemon.sh start historyserver

运行 wordcount：

```
echo "APDPlat is a java open source project, Application Product Development Platform." > text1.txt
echo "Yang Shangchuan is the founder of APDPlat which won the "2013 Outstanding Open Source Project" award." > text2.txt
bin/hadoop fs -mkdir input
bin/hadoop fs -put text1.txt input
bin/hadoop fs -put text2.txt input
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar wordcount input output
```

访问管理页面：

http://host001:8088
http://host001:50070
http://host001:19888

停止 dfs：

sbin/stop-dfs.sh

停止 yarn：

sbin/stop-yarn.sh

停止 historyserver：

sbin/mr-jobhistory-daemon.sh stop historyserver