[1. 网络设置 2](#_Toc21980)

[2. MyEclipse、Forecast、Flex配置 3](#_Toc23354)

[2.1. Windows上配置 hadoop环境变量 3](#_Toc1873)

[2.2. \*.hbm.xml文件与表 3](#_Toc30502)

[2.3. Bean及其属性在applicationContext-\*.xml中的定义 3](#_Toc4427)

[2.4.工程配置环境和配置文件 4](#_Toc29724)

[2.5.Flex基础 5](#_Toc11614)

[3. SVN 5](#_Toc15403)

[4. Linux常用命令 5](#_Toc250)

[4.1 CentOS下gcc/g++/gfortran编译环境的安装 6](#_Toc15388)

[4.1.1 linux上安装与卸载VMware 7](#_Toc5751)

[4.1.2 linux上安装VMware,kernel-headers与kernel-devel版本不一致问题 7](#_Toc10960)

[4.1.3 sudo 与su 7](#_Toc28534)

[4.1.4 Linux系统命令 7](#_Toc6305)

[4.1.5 Linux配置本地yum源 7](#_Toc826)

[4.2 Linux上跑Spark算法包缓存清除 7](#_Toc28365)

[4.3 zip/unzip命令 8](#_Toc20718)

[4.4 tar解压，编译文件 8](#_Toc1838)

[4.5 Linux编辑网卡配置 8](#_Toc11505)

[4.6 Shell脚本 8](#_Toc1748)

[4.7 Linux挂载 9](#_Toc27032)

[4.8 将windows中的文件夹共享到linux中 9](#_Toc14900)

[4.9 Windows远程连接Linux桌面 9](#_Toc16148)

[4.10 Centos6.5 安装桌面环境 10](#_Toc26635)

[4.11 Linux硬件环境查看 10](#_Toc13137)

[5. Hbase 10](#_Toc30763)

[5.1 矩阵相乘 11](#_Toc2575)

[6. MySql 12](#_Toc30144)

[6.1 navicat中null与字符串 12](#_Toc18052)

[6.2 Sql语句执行顺序及例子 12](#_Toc28896)

[6.3 Mysql修改最大连接数 12](#_Toc21316)

[6.4 mysql卸载 13](#_Toc8537)

[6.5 Mysql连接/远程连接权限 13](#_Toc19020)

[7. Windows 13](#_Toc32017)

[docs常用命令 13](#_Toc26196)

[修改默认浏览器 14](#_Toc18057)

[8. Tomcat 14](#_Toc12253)

[8.1 关闭tomcat进程 14](#_Toc31799)

[8.2 Linux启动Tomcat报错 14](#_Toc19354)

[8.3 Tomcat VM与java VM优化 14](#_Toc6090)

[9. Hadoop 15](#_Toc15648)

[9.1 集群中的四个配置文件 15](#_Toc17571)

[9.1.1. core-site.xml: 15](#_Toc18425)

[9.1.2. hdfs-site.xml: 15](#_Toc12039)

[9.1.3. mapred-site.xml: 16](#_Toc10909)

[9.1.4. yarn-site.xml: 16](#_Toc22988)

[9.1.5. hadoop-env.sh: 18](#_Toc2520)

[9.2. eclipse 中运行mapreduce 程序，java程序配置 18](#_Toc5784)

[9.3. hadoop jar/java -jar执行jar包，已指定主类 19](#_Toc22686)

[9.4. HDFS 19](#_Toc23784)

[9.5. Hadoop程序运行，ToolRunner 19](#_Toc10090)

[9.6. MapReduce程序测试，MRTest.java 20](#_Toc24180)

[9.7 Sqoop安装及hadoop-yarn-common-2.2.0.jar修改 20](#_Toc24465)

[10. Java 21](#_Toc27577)

[10.1. GB2312/GBK 21](#_Toc21043)

[10.2. 输入流/输出流 21](#_Toc4514)

[10.3. 加密类：com.caic.pub.util.DESUtils 22](#_Toc32687)

[10.4. Java环境变量 22](#_Toc25002)

[10.5. 转义 22](#_Toc25544)

[11. 常见英文单词 23](#_Toc8177)

[12. R语言基础 23](#_Toc13275)

[13. 算法Service\_页面\_参数 25](#_Toc12058)

[14. Spark运行命令样例 29](#_Toc10198)

[15. Hive 30](#_Toc4073)

[15.1 Hive中日期时间，时间戳转换 32](#_Toc27597)

# 网络设置

ThinkPad SL510进入BIOS与BOOT启动盘

都是F12，同一个界面，但进入BIOS，是选择最下面的STEUP,回车。

花生壳：

用户名：1234uuehff

密码：12345qwert

虚拟机192.168.222.80:root/admin

openstack:

192.168.63.200 root/123123 openstack\_yum

192.168.63.201 root/root controller

192.168.63.202 root/ root compute001

192.168.63.203 root/ root neutron

http://192.168.63.201/dashboard amdin/ADMIN\_PASS

tipdm/tipdmcom

wifi密码：tipdm.cn

wifi密码:tipdm998

# 2. MyEclipse、Forecast、Flex配置

## 2.1. Windows上配置 hadoop环境变量

org.apache.hadoop.util.Shell:336 - Failed to locate the winutils binary in the hadoop binary path

解决：

需要在windows上 配置 hadoop环境变量；HADOOP\_HOME F:\hadoop\hadoop-2.5.1

将项目中 win32需要的文件 解压到 F:\hadoop\hadoop-2.5.1\bin目录下。

## 2.2. \*.hbm.xml文件与表

===========================================================================================

<class name="com.caic.forecast.cloud.model.DmWebSMJob" table="dm\_web\_sm\_job">

<id name="jobId">

<generator class="assigned"></generator> //主键生成策略，assigned由程序在save之前指定，native系统指定。

</id>

<property name="algorithmName" type="string" column="ALGORITHM\_NAME"/>

</class>

类中要定义属性的get和set方法。

====================================================

<class name="类名" table="表名">

<id name="主键在java类中的字段名" column="对应表中字段" type="类型 ">

<generator class="主键生成策略"/>

</id>

</class>

## 2.3. Bean及其属性在applicationContext-\*.xml中的定义

<bean id="cloudAlgorithmManager" class ="com.caic.cloud.service.CloudAlgorithmManager" autowire="byName" >

<property name="dmHdfsFileService" ref="dmHdfsFileService"/>

<property name="pool" ref="rsessionPool"/> 其中name="pool"中的pool属性名称与CloudAlgorithmManager中属性pool名称一致，用beanID为rsessionPool的类填充。

</bean>

CloudAlgorithmManager类中要定义dmHdfsFileService，pool的get和set方法。

===============================================================================

## 2.4.工程配置环境和配置文件

1、api-conf.properties ——》Webservice 配置文件，127.0.0.1和localhost是有区别的，与浏览器保持一致！

2、Rserv-conf.properties ——》R服务配置文件，用户名和密码是登录 linux时的登录用户名和密码。

3、测试机上，WebService乱码，修改forecast工程中web.xml,com.caic.webservice.filter.ApiCharsetFilter，encoding为UTF-8,默认windows为GBK。

extMain.jsp中修改height: 48,logoCSG.png或修改conf下的ApplicationResources.properties文件。

login.jsp中修改loginBg.jpg，背景登录。这里的图片路径在frame工程WebRoot\images下。

frame工程web.xml中设置http://127.0.0.1:8080/login.jsp

myeclipse 搜索所有资源：ctrl + shift + r

java build path >> Default output folder >> caic-forecast-gdt/WebRoot/WEB-INF/classes

在此目录下：F:\Workspaces\caic-forecast-gdt\WebRoot\WEB-INF\classes为编译后的.class文件。

编译环境：java build path >>>> Libraries下的jar包，是编译时用。

运行环境：lib 下的jar包，项目运行时用。

tomcat运行项目，会在webroot/lib下找jar包，因此，需要将第三方jar包拷贝到lib下面(配置文件部署，不能用 Add Library方式添加第三方jar包)

，支持运行；add to build path 来支持编译。

如果是用Myeclipse 手动部署，可以用Add Library方式添加第三方jar包，因为这种方式，会把第三方jar包，自动拷贝到lib下面，而配置文件方式，不会自动拷贝。

**===============================================================================**

flex Server

root folder:${DOCUMENT}\caic-forecast\WebRoot

root URL:http://localhost:8080

context root:/forecast

-compiler.debug=false -services "F:\Workspaces\caic-forecast-gdt\WebRoot\WEB-INF\flex\services-config.xml" -locale zh\_CN en\_US -source-path=../locale/{locale}

source folder:src

output URL: http://localhost:8080/forecast

output folder: F:\Workspaces\caic-forecast-gdt\WebRoot\flex

## 2.5.Flex基础

Flex sdk3.2 不支持方法的重载(远程调用java方法同名时，参数个数不能一样，个数一样，类型不一样，也无法区别。)

MXML是一种用于创建用户界面的功能强大的标记性语言。在大多数情况下，使用ActionScript布局不如使用MXML布局。

ActionScript更适合描述逻辑和数据模型，然而MXML和ActionScript并不是有巨大的不同。实际上，在编译的时候，

MXML会转化成AS，MXML结构能理解成一种动态的AS类。

由于在大多数情况下，使用ActionScript布局不如使用MXML布局，所以用IDE环境拖拽的方式创建控件的UI界面是最方便不过的了。

UI确定后，再用AS描述逻辑和数据模型，如事件监听和相应的处理等。

ActionScript中，空心：变量,绿色：public ,红色：private。

[Bindable] 用来同步变量。

vertical 垂直的

horizontal 水平的

absolute 无条件的;

flex对象可以由 构造方法 和 mxml两种方式创建。

## 2.6 Flex 中HDFS数据传递

setData()对应点击页面 “导入数据”，方法会this.splitter = this.splitterInput1.text;调用hDFSExtractFiles.execFilesExtract(this.schemeId,-1,arrayPatterns,true); arrayPatterns是选择的文件数组，

从 /user/root/DATA/方案/\* 中抽取数据到 /user/root/ALGORITHM/0067/input/part-m-00000 ,第一次的话获取jobId返回。

接着完触发afterExtractFiles，它接受jobId,调用hDFSFileUtil.getFileData(this.jobId,this.splitter);读取/user/root/ALGORITHM/0067/input/part-m-00000数据，返回数据和列数。

接着触发onGetFileData，它接受数据和列，绑定显示数据this.fileDataText.text=file\_data; 及更新列，它再调用this.dmSparkSVDService.loadData(this.jobId,this.currentScheme);保存jobId等数据到

dm\_cloud\_job表,没有返回值。(此方法完成训练后就不在调用，有判断)。

接着完触发 onLoadData ,他在commonCloud.as中，由于this.dmSparkSVDService.loadData没有返回值，则直接返回。

hdfsUploadClick()，对应点击 “配置输入”，他在commonCloud.as中，点击时弹出HDFSUploadMenu.xml框。

hdfsUploadClick()内容：{

hdfsUploadMenu = HDFSUploadMenu(PopUpManager.createPopUp(this, HDFSUploadMenu, true));

// 注册目录配置窗口监听事件

hdfsUploadMenu.addEventListener(HDFSUploadEvent.UPLOAD,handleUpload);

hdfsUploadMenu.addEventListener(HDFSUploadEvent.CHECKPERMISSION,handleCheck);

}

HDFSUploadMenu.xml中，有两个输入框，目录和分隔符，目录对应change="checkPermission()",分隔符对应upload()，

checkPermission()：

var hdfsCheckEvent :HDFSUploadEvent=new HDFSUploadEvent(

HDFSUploadEvent.CHECKPERMISSION,this.inputHdfsDir.text,this.splitterInput2.text,false,false);

dispatchEvent(hdfsCheckEvent);

解释：

用输入框this.inputHdfsDir.text，this.splitterInput2.text的值构造HDFSUploadEvent对象，分别赋值给

HDFSUploadEvent.as中pathIn，splitter两个变量，函数最后dispatchEvent(hdfsCheckEvent)抛出hdfsCheckEvent对象，

执行完毕触发监听函数handleCheck，监听函数接受到hdfsCheckEvent事件对象，可用event.pathIn和this.splitter = event.splitter更新值;

upload():(与checkPermission同理)

private function upoad(): void {

// 发送更新schemedata 事件,用于更新方案

var hdfsUploadEvent :HDFSUploadEvent=new HDFSUploadEvent(HDFSUploadEvent.UPLOAD,

this.inputHdfsDir.text,this.splitterInput2.text,false,false);

dispatchEvent(hdfsUploadEvent);

}

HDFSUploadEvent.as中有pathIn，splitter两个变量，及构造方法。

内容：public var pathIn : String ;

public var splitter :String;

public function HDFSUploadEvent(type:String,pathIn:String,splitter:String,

bubbles:Boolean=false, cancelable:Boolean=false)

{ super(type, bubbles, cancelable);

this.pathIn=pathIn;

this.splitter=splitter;

}

输入路径和分隔符后，点击 “配置”，this.splitter = event.splitter;HDFS导入方式，第一次更新字符串，

之后调用setHDFSDir()，此方法在每个页面对应的.as中，setHDFSDir()调用this.dmSparkSVDService.loadData(this.currentScheme,this.hdfsUploadMenu.inputHdfsDir.text,-1);

每个service有两个loadData()方法，属于重载，页面导入和HDFS导入用了不同的loadData()，但是用了同一个onLoadData(),页面调用onLoadData(),直接返回，因为他用的loadData()无返回值，

onLoadData()中的event就为空，且其中还对数据库路径有判断。页面先从HDFS读取数据，页面显示，再loadData()保存数据库，

而HDFS导入先 loadData()，保存jobId,路径等 到数据库，返回jobId，接着触发onLoadData()。

// 添加列分隔符

[Bindable] private var splitter:String =",";

setData()-->this.splitter = this.splitterInput1.text;// 添加分隔符

afterExtractFiles下的

// 添加分隔符

hDFSFileUtil.getFileData(this.jobId,this.splitter);

共

handleUpload// 添加分隔符，抛出事件。

this.dmHdfsFileService.getHDFSFileData(this.jobId,this.splitter);// 获取文件数据// 添加分隔符

private function setHDFSDir():void{

// this.splitter = this.hdfsUploadMenu.splitterInput2.text;// 更新分隔符；

this.mahoutFuzzyKmeansService.loadData(this.currentScheme,this.hdfsUploadMenu.inputHdfsDir.text,-1);

:void{

//--\*\*execFilesExtract(String strPathIn, String strPathOut, String[] arrayPatterns, int[] arrayIndexes)

//--\*\* 测试获得数组

// 更新 splitter的值

// Alert.show("splitter:"+this.splitter);

<mx:HBox>

<mx:Label text="列分隔符" />

<mx:TextInput id="splitterInput1" text="{this.splitter}" width="90" />

</mx:HBox>

HDFSUploadMenu.xml

<mx:HBox width="100%" horizontalAlign="left" >

<mx:Label text="列分隔符" />

<mx:TextInput id="splitterInput2" text="," width="175" />

<!--<mx:Spacer width="38" />-->

<mx:Button label="配置" click="upoad()"/>

</mx:HBox>

<mx:VBox width="100%" >

<mx:HBox>

<mx:Label text="@Resource(key='scheme.inputdir', bundle='scheme')" width="60" textAlign="right"/>

<mx:Label id="permissionRet" text="请输入HDFS目录" width="100" textAlign="right"/>

</mx:HBox>

<mx:TextInput id="inputHdfsDir" text="/" width="280" change="checkPermission()" />

</mx:VBox>

<mx:HBox width="100%" horizontalAlign="left" >

<mx:Label text="列分隔符" />

<mx:TextInput id="splitterInput2" text="," width="175" />

<!--<mx:Spacer width="38" />-->

<mx:Button label="配置" click="upoad()"/>

</mx:HBox>

</mx:VBox>

commonCloud.as

private function handleUpload(event:HDFSUploadEvent):void{

if(!this.finishedCheck){

this.hdfsUploadMenu.permissionRet.text="校验未通过!";

return ;

}

// 输入目录设置,每个页面有自己的实现

showLoading();

// showLoading\_commonCloud();

PopUpManager.removePopUp(this.hdfsUploadMenu);

// setHDFSDir(event.pathIn);

this.splitter = event.splitter;

// Alert.show("handleUpload:"+this.splitter);

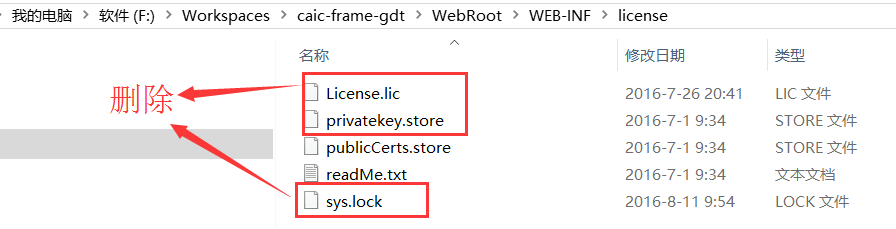
setHDFSDir();

}

// Alert.show("onLoadData:"+this.splitter);

## 2.7 License设置

1、如果是从其他地方拷贝过来的项目，删除以下红色部分，登录会生成机器码，然后授权。



# 3. SVN

svn用户名/密码:weiwenchao/tipdm

spark算法源码已建立svn，地址svn://112.74.76.115:9999/gdt/caic-spark-gdt

svn://112.74.76.115:9999/gdt/caic-forecast-gdt

svn://112.74.76.115:9999/gdt/caic-frame-gdt

svn://112.74.76.115:9999/gdt/caic-ria-gdt

team 中cleanup/refresh，之后再更新，去掉锁。

project 中clean，自动重新编译。

svn 本地多于服务器上的文件夹不能删除，刷新后才可以删除。

# Linux常用命令

Vi后无法保存没有权限的文件：

:w !sudo tee % > /dev/null

``和$()都是命令替换。

ls $(which vmware) == ls `which vmware`

1.gcc是编译一个文件，make是编译多个源文件的工程文件的工具。MAKE是运用GCC去编译文件的。

cat /etc/issue Linux显示系统发行版信息

Linux查看系统32位、64位：file /bin/bash

查看home下面各目录大小(只是home下面的一级目录)：du -h --max-depth=1 /home

/opt/hadoop-2.2.0/sbin/mr-jobhistory-daemon.sh start historyserver

history |grep CMD 查看包含CMD命令的历史记录。

Vi 替换命令：/替换为 -

加g替换每行全部，不加g替换每行第一个。

:1,$s/\//\-/g

shell通配符与正则表达式：

通配符是shell在做PathnameExpansion时用到的。说白了一般只用于文件名匹配，它是由shell解析的，比如find，ls，ll，cp，scp，mv等。其中\*?在通配符中代表的意思和正则中的意思也不一样。

|  |  |  |
| --- | --- | --- |
| ****通配符**** | ****含义**** | ****实例**** |
| \* | 匹配 0 或多个字符 | a\*b a与b之间可以有任意长度的任意字符, 也可以一个也没有, 如aabcb, axyzb, a012b, ab。 |
| ? | 匹配任意一个字符 | a?b a与b之间必须也只能有一个字符, 可以是任意字符, 如aab, abb, acb, a0b。 |
| [list] | 匹配 list 中的任意单一字符 | a[xyz]b  a与b之间必须也只能有一个字符, 但只能是 x 或 y 或 z, 如: axb, ayb, azb。 |
| [!list]或[^list] | 匹配 除list 中的任意单一字符 | a[!0-9]b a与b之间必须也只能有一个字符, 但不能是阿拉伯数字, 如axb, aab, a-b。 |
| [c1-c2] | 匹配 c1-c2 中的任意单一字符 如：[0-9] [a-z] | a[0-9]b 0与9之间必须也只能有一个字符 如a0b, a1b... a9b。 |
| [!c1-c2]或[^c1-c2] | 匹配不在c1-c2的任意字符 | a[!0-9]b 如acb adb |
| {string1,string2,...} | 匹配 sring1 或 string2 (或更多)其一字符串 | a{abc,xyz,123}b 列出aabcb,axyzb,a123b |

正则表达式是用来匹配字符串的，针对文件内容的文本过滤工具里，大都用到正则表达式，如vi，grep，awk，sed等。

top 命令，查看资源。

1、tail -f filename

说明：监视filename文件的尾部内容（默认10行，相当于增加参数 -n 10），刷新显示在屏幕上。退出，按下CTRL+C。

2、tail -n 20 filename

说明：显示filename最后20行。

3、head -n 20 filename

说明：显示filename前面20行。

tomcat下logs下文件全部删除，会自动创建。

tail -f /usr/local/apache-tomcat-7.0.65/logs/catalina.out

netstat -ano

top 命令，查看资源。

系统编码：locale

nohup hive --service hiveserver2 & 后台运行程序

==========================================================================

注意：一定要进入到目标文件夹，在当前文件夹下，才能使用此命令！

ntpdate -u s1a.time.edu.cn 北京邮电大学

ln -s 源文件 目标文件名

============================================

统计当前文件个数：ls -l |grep "^-"|wc -l

帮助：chkconfig --help或者 man chkconfig

==========================================================

chkconfig --list 列出所有的系统服务

chkconfig --add httpd 增加httpd服务

chkconfig --del httpd 删除httpd服务

chkconfig --level 2345 httpd on/off/reset

mysql linux下自启动。

cp /usr/local/mysql/share/mysql/mysql.server /etc/rc.d/init.d/mysqld

chkconfig --add mysqld

chkconfig --level 2345 mysqld on/off/reset

Linux/Unix 的档案存取权限分为三级 : 档案拥有者、群组、其他。

chmod [-cfvR] [ugoa...][[+-=][rwxX]...][,...] file

chmod ugo+r file1.txt

chmod ug+w,o-w file1.txt file2.txt

chmod -R a+r \*

chown [-cfhvR] user[:group]

一般来说，这个指令只有是由系统管理者(root)所使用，一般使用者没有权限可以

改变别人的档案拥有者，也没有权限可以自己的档案拥有者改设为别人。

将档案 file1.txt 的拥有者设为 users 群体的使用者 jessie :

chown jessie:users file1.txt

useradd haha （添加用户）

passwd haha （修改密码）

chgrp [-cfhvR] 组 文件

chgrp users file1.txt

=================================

1）把 textfile1 的档案内容加上行号后输入 textfile2 这个档案里

cat -n textfile1 > textfile2

2）将几个文件加行号合并为一个文件：

cat -n file1 file2 > file

cat -n file1 file2 >> file

3）把test.txt文件扔进垃圾箱，赋空值test.txt

cat /dev/null > /etc/test.txt

============================================

more命令功能：让画面在显示满一页时暂停，此时可按空格健继续显示下一个画面，或按Q键停止显示。

less命令功能：less命令的用法与more命令类似，也可以用来浏览超过一页的文件。所不同的是less命令除了可以按空格键向下显示文件外，还可以利用上下键来卷动文件。当要结束浏览时，只要在less命令的提示符“: ”下按Q键即可。

## 4.1 CentOS下gcc/g++/gfortran编译环境的安装

centos6.5 编译工具 安装：yum -y install gcc gcc-c++ kernel-devel

1.安装

在ubuntu下直接apt-get install gcc g++就可以了。按照类似的逻辑，再CentOS下yum install gcc g++ 报告无法找到g++包。 查了一下，原来这个包的名字叫做gcc-c++。完整的应该是yum install gcc gcc-c++

(1)安装gcc，命令：yum install gcc

(2)安装g++/gfortran，命令：yum install gcc-c++

执行完该命令后，g++/gfortran都被安装上了。

===============================================================================

### 4.1.1 linux上安装与卸载VMware

卸载vmware:

1.先查看安装的虚拟机

vmware-installer -l

2.卸载虚拟机

sudo vmware-installer --uninstall-product vmware-workstation

安装VMware:

./VMware-Workstation-Full-11.1.0-2496824.x86\_64.bundle

### 4.1.2 linux上安装VMware,kernel-headers与kernel-devel版本不一致问题

uname -r ,uname -a 查看系统内核版本。

### 4.1.3 sudo 与su

root用户，系统密码修改：

sudo passwd root 即可修改 root 密码。

### 4.1.4 yum常用命令

参考：http://blog.chinaunix.net/uid-346158-id-2131252.html

yum check-update 检查可更新的所有软件包

yum update 下载更新系统已安装的所有软件包

yum upgrade 大规模的版本升级,与yum update不同的是,连旧的淘汰的包也升级

yum install 安装新软件包

yum update 更新指定的软件包

yum remove 卸载指定的软件包

yum groupinstall 安装指定软件组中的软件包

yum groupupdate 更新指定软件组中的软件包

yum groupremove 卸载指定软件组中的软件包

yum grouplist 查看系统中已经安装的和可用的软件组

yum list 列出资源库中所有可以安装或更新以及已经安装的rpm包

yum list 列出资源库中与正则表达式匹配的可以安装或更新以及已经安装的rpm包

yum list available 列出资源库中所有可以安装的rpm包

yum list available 列出资源库中与正则表达式匹配的所有可以安装的rpm包

yum list updates 列出资源库中所有可以更新的rpm包

yum list updates 列出资源库中与正则表达式匹配的所有可以更新的rpm包

yum list installed 列出资源库中所有已经安装的rpm包

yum list installed 列出资源库中与正则表达式匹配的所有已经安装的rpm包

yum list extras 列出已经安装的但是不包含在资源库中的rpm包

yum list extras 列出与正则表达式匹配的已经安装的但是不包含在资源库中的rpm包

yum list recent 列出最近被添加到资源库中的软件包

yum search 检测所有可用的软件的名称、描述、概述和已列出的维护者，查找与正则表达式匹配的值

yum provides 检测软件包中包含的文件以及软件提供的功能，查找与正则表达式匹配的值

yum clean headers 清除缓存中的rpm头文件

yum clean packages 清除缓存中rpm包文件

yum clean all 清除缓存中的rpm头文件和包文件

yum deplist 显示软件包的依赖信息

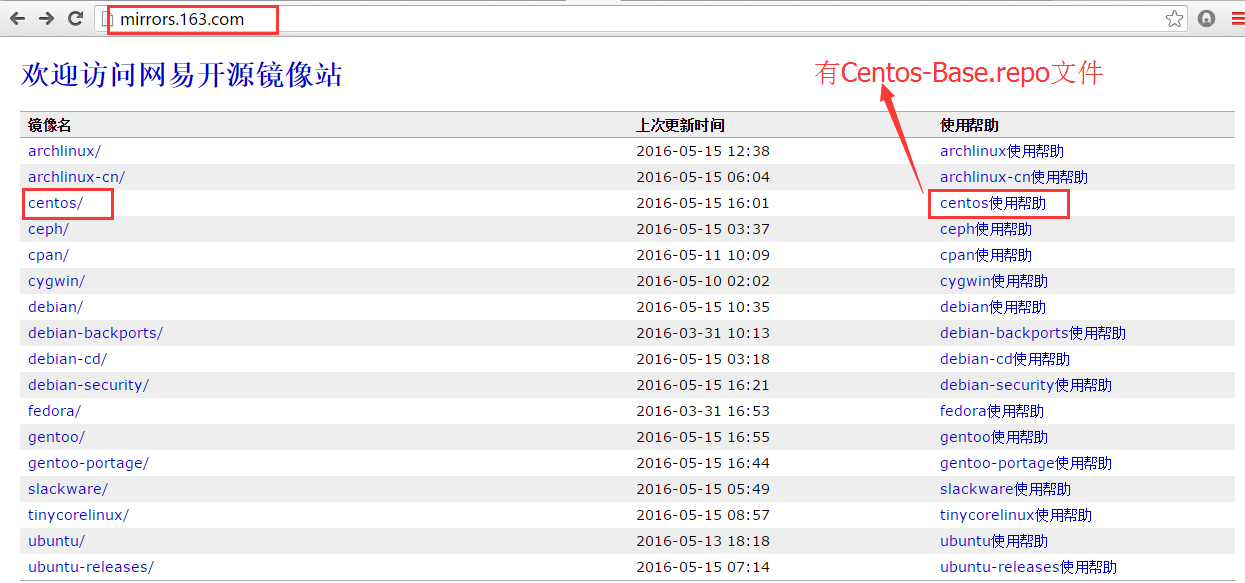
### 4.1.5 Linux配置本地yum源、ftp(局域网)yum源

ftp(局域网)yum源：<http://www.linuxidc.com/Linux/2017-03/141391.htm>

**\*.repo内容格式：file:/// http:// ftp://**

**局域网可以使用Ftp方式，配置yum源。**

**网易开源镜像站：<http://mirrors.163.com/>**

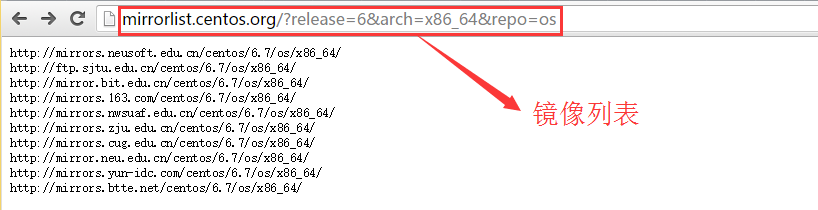


/etc/yum.repos.d/目录下CentOS-Base.repo内容如下：

[base]

name=CentOS-$releasever - Base

mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os&infra=$infra



#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/

http://mirror.centos.org/centos/7/os/x86\_64/



$releasever：代表发行版的版本，6,7，指的是centos6,centos7。

$arch：cpu体系，如 x86\_64,i386等

$basearch：cpu的基本体系组，如 x86\_64,i386等。

**本地源配置3中方法：**

1. **使用挂载光盘，作为yum源**

虚拟机启动时，挂载光盘(添加iso路径)，勾选连接。

挂载：mount /dev/dvd /media/dvd

配置yum源：

清除其他的yum源，否则会先联网安装！

mkdir /etc/yum.repos.d/bak

mv /etc/yum.repos.d/\*.repo /etc/yum.repos.d/bak

cp /etc/yum.repos.d/bak/Centos-Media.repo /etc/yum.repos.d/localyum.repo

localyum.repo内容如下：

[c6-media-localyum]

name=CentOS-$releasever - localyum

baseurl=file:///media/dvd

http://

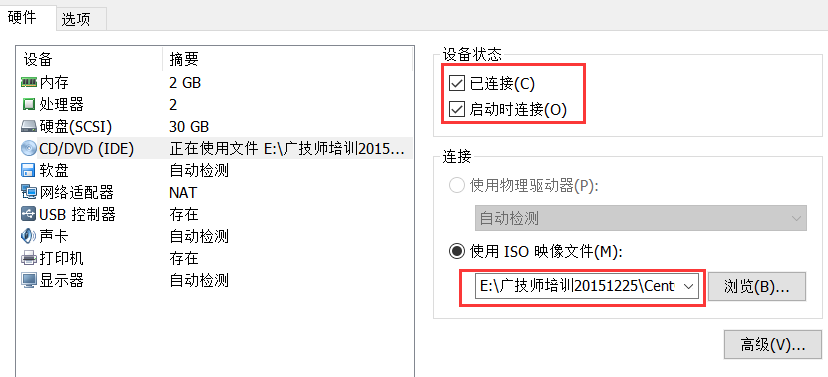
gpgcheck=0

enabled=0

**编辑之后：**

yum clean all

yum makecache



**UltraISO工具:**

**用目录文件，可以直接制作ISO镜像，可以在Linux中挂载。**

**2.使用本地ISO文件**

挂载本地ISO文件，ISO文件可以从网站下载，例如：



**mount -o loop CentOS-6.5-x86\_64-minimal.iso /media/dvd**

**余下同1的步骤。**

1. **使用本地目录，createrepo创建本地yum源：**

**1）将ISO中的Packages中的rpm包，拷贝到 /opt/yumserver**

**2）先联网安装createrepo工具包**

**3）创建索引信息：createrepo /etc/yumserver**

**Yum clean all**

**Yum makecache**

<http://www.cnblogs.com/shenliang123/p/3829044.html>

http://www.cnblogs.com/mchina/archive/2013/01/04/2842275.html

## 4.2 Linux上跑Spark算法包缓存清除

=========================================================

查看磁盘占用情况：du -h --max-depth=1 /data/hadoop/yarn/local/usercache/root/filecache/

/data/hadoop/yarn/local/usercache/root/filecache

df -h /

## 4.3 zip/unzip命令

zip -r mydata.zip mydata #压缩mydata目录

把/home目录下面的mydata.zip解压到mydatabak目录里面

unzip mydata.zip -d mydatabak

## 4.4 tar解压，编译文件

(一) 将tgz文件解压到指定目录

比如将/source/kernel.tgz解压到 /source/linux-2.6.29 目录

tar -zxvf /source/kernel.tgz -C /source/ linux-2.6.29

(三)然后执行 ./configure 配置安装环境， --prefix=/usr/local/test 指定安装目录。 [进行安装设置,这时会出现一大堆的字符,等一段时间吧]

(四)等到配置完成后执行 make [编译中,也要等一会]

(五)编译完后 make install [安装]

(六)最后 make clean [清除掉配置编译的一些文件]

4.5 Linux编辑网卡配置

nm\_controlled=yes|no:参考：<http://www.2cto.com/os/201208/149576.html>

vi /etc/udev/rules.d/70-persistent-net.rules

编辑网卡网络：

vi /etc/sysconfig/network-scripts/ifcfg-eth0

添加：DNS1=192.168.222.2 DNS2=8.8.8.8 后可以ping通www.baidu.com（8.8.8.8是Google公司提供的免费DNS服务器的IP地址。）

source /etc/sysconfig/network-scripts/ifcfg-eth0

重启网络：service network restart

## 4.6 Shell脚本

在shell脚本的开头往往有一句话来定义脚本的类型，和使用哪种sh解释器来解释脚本。

/bin/目录下的程序：

(1) #!/bin/sh

(2) #!/bin/bash

脚本test.sh内容：

#!/bin/sh

source pcy.sh #pcy.sh并不存在

echo hello

执行./test.sh，屏幕输出为：

./test.sh: line 2: pcy.sh: No such file or directory

由此可见，在#!/bin/sh的情况下，source不成功，不会运行source后面的代码。

修改test.sh脚本的第一行，变为#!/bin/bash，再次执行./test.sh，屏幕输出为：

./test.sh: line 2: pcy.sh: No such file or directory

hello

由此可见，在#!/bin/bash的情况下，虽然source不成功，但是还是运行了source后面的echo语句。

但是紧接着我又试着运行了一下sh ./test.sh，这次屏幕输出为：

./test.sh: line 2: pcy.sh: No such file or directory

表示虽然脚本中指定了#!/bin/bash，但是如果使用sh 方式运行，如果source不成功，也不会运行source后面的代码。

## 4.7 Linux挂载

1， 插入U盘后，把鼠标点击在Linux里面u盘会在linux中出现，此时windows中没有。

之后输入fdisk -ll命令,查看U盘使用情况。

表明该设备已经被识别，并且该设备的名称为sdb1。

2， 建立一个挂载usb的挂载目录，（可以自定义目录）

mkdir /mnt/usb

3, 将U盘挂载在到/mnt/usb目录下

mount命令格式：mount [-参数] [设备名称] [挂载点] [其他参数]

mount -t vfat /dev/sdb1/ /mnt/usb -o iocharset=gb2312

-t 指定设备的文件系统类型，在此，使用 -t vfat 是由于所使用U盘的文件

系统类型是 fat32，如果是ntfs的， 则使用-t ntfs-3g，大家可根据自己的

文件系统类型灵活修改。

-o 指定挂载文件系统时的选项，在此， -o iocharset=gb2312 设定中文字符集。

假设你的locale是zh\_CN.UTF-8，相应命令应为 -o iocharset=utf8。

那么U盘挂载成功，这样就可以在目录/mnt/usb目录下查看U盘的内容。

4，U盘的卸载

命令umount /mnt/usb可以将U盘从指定目录下卸载。

需要注意的是，在卸载U盘时，终端必须从当前U盘的挂载目录下退出来，否则会显示用户忙。

注意：ntfs-3g文件系统的u盘需要安装一个包，ntfs-3g\_ntfsprogs-2014.2.15.tgz。

## 4.8 将windows中的文件夹共享到linux中

===============================================================================

192.168.0.101 为windows系统，linux系统192.168.0.118 相互ping同，关闭防火墙。

share 为windows共享文件夹。z为linux 已创建目录。

mount //192.168.0.101/share ./z -o username=admin,password=sa

## 4.9 Windows远程连接Linux桌面

地址：<http://blog.csdn.net/wandrong/article/details/20393733>

## 4.10 Centos6.5 安装桌面环境

<http://jingyan.baidu.com/article/ca2d939dd1dabbeb6c31ce24.html>

## Linux硬件环境查看

CPU型号： less /proc/cpuinfo |grep model

fdisk: 磁盘

env

Cat /proc/cpuinfo

du -h 理解。

mkdir -p /user/root/data 递归创建目录。

ps -ef

Last

Crontal -l

route -n

uptime 运行时间，用户，负载

mount | column -t 查看挂载分区情况

物理CPU个数cat /proc/cpuinfo| grep "physical id"| sort| uniq| wc -l

每个物理CPU的核数cat /proc/cpuinfo| grep "cpu cores"| uniq

逻辑CPU的总数（包括所有物理CPU、超线程）：

cat /proc/cpuinfo| grep "processor"| wc -l

一般来说，物理CPU个数×每颗核数就应该等于逻辑CPU的个数，如果不相等的话，则表示服务器的CPU支持超线程技术

一般情况，我们认为一颗cpu可以有多核，加上intel的超线程技术(HT), 可以在逻辑上再分一倍数量的cpu core出来  
逻辑CPU数量=物理cpu数量 x cpu cores 这个规格值 x 2(如果支持并开启ht)

查看CPU信息

          当我们 cat /proc/cpuinfo 时、

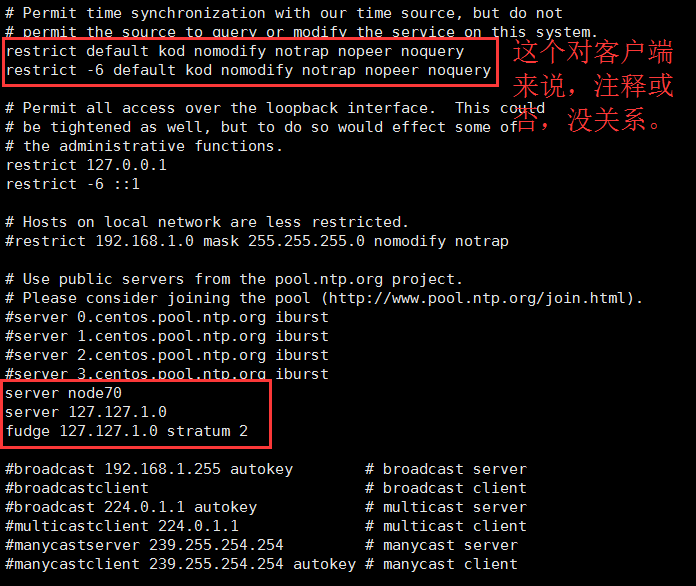
          具有相同core id的CPU是同一个core的超线程

          具有相同physical id的CPU是同一个CPU封装的线程或核心

4核8线程解释：

每个单位时间内，一个CPU只能处理一个线程（操作系统：thread），以这样的单位进行，如果想要在一单位时间内处理超过一个线程是不可能的，除非是有两个CPU的实体单元。多核心技术是将多个一样的CPU放置于一个封装内（或直接将两个CPU做成一个芯片），而英特尔的HT技术是在CPU内部仅复制必要的资源、让CPU模拟成两个线程；也就是一个实体核心，两个逻辑线程，在一单位时间内处理两个线程的工作，模拟实体双核心、双线程运作。  
将以上原理拓展：  
四核即代表着该CPU具有四个“CPU”，八线程则意味着每个CPU存有两个逻辑线程，总共有八个线程。

## 4.12 本地Ntp服务器



**server 210.72.145.44 prefer # 优先 中国国家受时中心**

**server s1a.time.edu.cn #北京邮电大学**

1）调整时区：直接拷贝Shanghai 到 localtime 中。或：

**ln -sf /usr**/share/zoneinfo/posix/Asia/Shanghai /etc/localtime # f会强制替换原有localtime。

显示硬件时间：hwclock --show

硬件时间设置成系统时间：hwclock --hctosys

系统时间设置成硬件时间：hwclock --systohc

修改硬件时间：hwclock --set --date=”mm/dd/yy hh:mm:ss” # 05/13/16 20:00:00 2016年5月13号

修改系统时间：date -s ”mm/dd/yy hh:mm:ss”

1. 服务端和客户端 都要安装ntp服务，配置/etc/ntp.cnf

安装服务：yum install ntpdate ntp -y #ntp对应的是ntpd服务。

chkconfig --level 2345 ntpd on

**配置**：node70,node71,node72，node70作为时间服务器，给node71,node72提供时间。

1. **ln -sf /usr**/share/zoneinfo/posix/Asia/Shanghai /etc/localtime # f会强制替换原有localtime。
2. 修改硬件时间：

hwclock --set --date=”mm/dd/yy hh:mm:ss” # 05/13/16 20:00:00 2016年5月13号

1. 硬件时间设置成系统时间：hwclock --hctosys
2. 检查三台机器是否都安装ntpd，安装服务：yum install ntpdate ntp -y #ntp对应的是ntpd服务。
3. 按如下截图编辑三个/etc/ntp.conf文件。

node70服务端配置/etc/ntp.conf：

node70用本地时间作为自己的时间源。

server 127.127.1.0

fudge 127.127.1.0 stratum 10

Node71,node72客户端配置/etc/ntp.conf：

Server node70

server 127.127.1.0

fudge 127.127.1.0 stratum 2 #stratum 2 指127.127.1.0 时间源级别为第二，当node70不行时，用它。

1. 在node70上开启ntpd服务: service ntpd start

7. 在node71,node72分别上执行，ntpdate node70，hwclock --systohc。

8. 在node71,node72上开启ntpd服务：service ntpd start

9. 在三台机器上执行：ntpq -p，查看状态（三台机器都要先开启ntpd服务）。

10 设置三台机器，ntpd服务开机自启。

Ntpdate命令 ：是跳跃式同步时间，直接将本地时间与服务器改成一致。

Ntpd 服务：是逐渐矫正客户端时间与服务器保持一致，开始使用ntpd服务同步时间时先要使用ntpdate 命令同步一下服务器时间。

Ntpdate 命令 + crontab -e 定时任务，也能达到与ntpd服务一样的效果。

其中parameter的参数主要有：  
ignore        ：    拒绝所有类型的ntp连接  
nomodify    ：    客户端不能使用ntpc与ntpq两支程式来修改服务器的时间参数  
noquery        ：    客户端不能使用ntpq、ntpc等指令来查询服务器时间，等于不提供ntp的网络校时  
notrap        ：    不提供trap这个远程时间登录的功能  
notrust        ：    拒绝没有认证的客户端  
nopeer        ：    不与其他同一层的ntp服务器进行时间同步

\* 主服务器

+ 辅助服务器

- 不合格的服务器

X 服务器不可用

参考：<http://www.360doc.com/content/10/1020/17/834950_62500025.shtml>

http://blog.sina.com.cn/s/blog\_539438c70101j3z7.html

## 4.13 vmware Linux磁盘压缩

1.xshell连接虚拟机，执行：

dd if=/dev/zero of=/0bits bs=20971520

sudo rm /0bits

之后关闭linux虚拟机系统。

2.在dos中，进入Vmware软件安装目录：

cd C:\Program Files (x86)\VMware\VMware Workstation

例如执行命令：vmware-vdiskmanager -k "D:\Virtual-Machines3\tipdm\tipdm.vmdk"

如：vmware-vdiskmanager -k "D:\Virtual-Machines3\tipdm2\tipdm-cl1.vmdk"

如：vmware-vdiskmanager -k " D:\Virtual-Machines3\tipdmInner\tipdm-cl2.vmdk"

**vmware Linux磁盘扩展**

参考：<http://blog.chinaunix.net/uid-26495963-id-3473072.html>

考参：

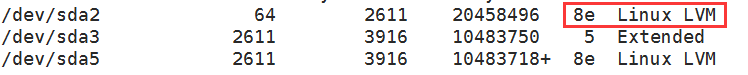
<http://wenku.baidu.com/link?url=1ycxh7morbEiX4F21Cq7GYA403aKrMyWORqNx0mn82wgG6TYSqE6gdF7lVlrtbufxiFBndLCj19rv7RpT_lyB1K0f0HNiAMzWed3aUQuOTW>

磁盘扩展两种方式：一种为新磁盘新建分区，单独挂载一个目录。另一种将新磁盘分区加到已有分区中。

1. 扩展磁盘。Vmware界面中操作或vmware-vdiskmanager -x 30Gb “D:\Virtual-Machines3\hive\_jiaoCai\hive\_jiaoCai.vmdk”
2. root登录，对扩展的磁盘分区，划分为主分区，或扩展分区，扩展分区需要再在其下面分逻辑分区，逻辑分区才可以格式化。
3. 重启。
4. 格式化分区（即指定文件系统）
5. 挂载：

这里新磁盘分区的使用有两种方式：

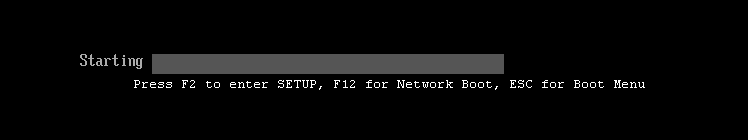
1. 将新磁盘空间，单独建立划分一个分区，指定文件系统，挂载到新的目录。
2. 将新的磁盘分区，添加到已有的分区中。
3. 添加新分区到已有分区中有两种方式：
4. 原分区使用LVM管理。类似：



1. 原分区类型为：



Linux进入BIOS，与主板有关，与系统无关，因为那时系统还未加载。



## 4.14 VMware虚拟系统文件夹下几个常用文件的介绍

1. vmx文件：表示虚拟系统文件，我们用虚拟机程序打开这个文件以启动虚拟系统。

2. vmem文件：表示虚拟内存文件，与pagefile.sys（亦称分页文件）同。当虚拟系统执行关机操作后，vmem文件消失，但挂起关闭时，不消失。

3. vmdk文件：表示虚拟机的一个虚拟磁盘。

4. vmss文件：执行挂起操作后产生的文件。（相当于一个快照文件？）

5. 当运行一个“虚拟系统”时，为防止该系统被另外一个VMware程序打开，导致数据被修改或损坏，VMware会自动在该“虚拟系统”所在的文件夹下，生成三个锁定文件

（虚拟系统锁定，虚拟磁盘锁定，虚拟内存锁定），分别为：

”systemType.vmx.lck”,”systemType.vmdk.lck”,”systemTyep.vmem.lck”。

当该虚拟系统正常关机后，VMware解锁，”systemTyep.vmdk.lck”和”systemType.vmem.lck”文件夹消失。

常见问题

虽然VMware这种锁定机制能够很好地防止同一个虚拟系统文件被多个VMware运行程序运行，避免了数据被破坏。但它也带来了一些问题。即当出现断电或其它意外情况，可能导致某个虚拟系统文件无法正常打开。原因往往在于：该虚拟系统文件没有解锁。解决办法：只要把三个lck文件夹删去即可。

## 4.15 LINUX定时任务：

参考：http://www.blogjava.net/freeman1984/archive/2010/09/23/332715.html

## 4.16 apache、httpd、tomcat区别

Apache是web服务器，Tomcat是应用（java）服务器，它只是一个servlet容器，是Apache的扩展。

Apache只支持静态网页，但像asp,php,cgi,jsp等动态网页就需要Tomcat来处理。

Apache和Tomcat整合使用：如果客户端请求的是静态页面，则只需要Apache服务器响应请求；如果客户端请求动态页面，则是Tomcat服务器响应请求；

apache和httpd是一个，到apache2后就叫httpd了，也更符合其http server的特性。而apache现在成为 apache基金会下几十种开源项目的标识。。

httpd启动：

Redhat系：包括 CentOS，Fedora是这样的：

service httpd start/stop/restart

Debian系（包括 Ubuntu）可以是这样的：

/etc/init.d/apache2 start/stop/restart

## 4.17 Ganglia部署

ganglia部署(单节点): http://boendev.iteye.com/blog/1750615

http://blog.csdn.net/a15039096218/article/details/7857308

setenforce 0

# for Ganglia 3.1.7 support

\*.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31

\*.sink.ganglia.period=10

\*.sink.ganglia.supportsparse=true

\*.sink.ganglia.slope=jvm.metrics.gcCount=zero,jvm.metrics.memHeapUsedM=both

\*.sink.ganglia.dmax=jvm.metrics.threadsBlocked=70,jvm.metrics.memHeapUsedM=40

namenode.sink.ganglia.servers=node51:8649

datanode.sink.ganglia.servers=node55:8649

resourcemanager.sink.ganglia.servers=node53:8649

nodemanager.sink.ganglia.servers=node55:8649

#jobtracker.sink.ganglia.servers=node53:8649

#tasktracker.sink.ganglia.servers=node55:8649

#maptask.sink.ganglia.servers=fc20:8649

#reducetask.sink.ganglia.servers=fc20:8649

ganglia安装步骤：

1.安装依赖包，ganglia-web,gmond,gmetad,httpd。http://blog.csdn.net/a15039096218/article/details/7857308

2.时间同步，ganglia初始化有误，会有缓存数据，默认路径，/var/lib/ganglia/rrds，/var/lib/ganglia/dwoo/cache，删除，重启服务即可。

3.ganglia 3.1之后使用hadoop-metrics2.properties文件

4.分组时，gmetad.conf文件中，data\_source默认端口也要加上，如：node51:8649,不然hadoop集群无图像。

集群分组：http://slaytanic.blog.51cto.com/2057708/808981/

yum安装：http://boendev.iteye.com/blog/1750615

gmetad.conf配置文件：http://www.tuicool.com/articles/rYV3uu http://book.2cto.com/201309/32333.html

详细配置：http://www.2cto.com/os/201409/338518.html

## 4.18 Linux Myeclipse卡死解决

删除原工作空间的.metadata/目录（即恢复默认工作空间），打开即可。切换到原有工作空间目录，再把项目导入即可。

参考：<http://www.blogjava.net/tinguo002/archive/2013/02/05/395137.html>

## 4.19 Oracle安装

默认的用户名和密码：

scott 密码是 tiger

sys 密码是 change\_on\_install

system 密码是 manager

sysman 密码是 oem\_temp

所有用户：select \* from all\_users;

自定义用户：select \* from user\_users;





查看数据库名、实例名、服务名：

1）sqlplus gxms/gxms@orcl

2）show parameter name;

.ora文件：cd /home/oracle/app/oracle/product/11.2.0/dbhome\_1/dbs

## 4.20 Centos67安装中文支持

挂载：

mount /dev/dvd /media

1. 安装中文支持包

yum groupinstall chinese-support

2. 修改默认语言

vim /etc/sysconfig/i18n

LANG="zh\_CN.UTF-8"

3. 重启

reboot

## 4.21 Centos6.7 SVN安装及配置

node80有配置svn。

参考：http://www.iitshare.com/linux-svn-installation-and-configuration.html

# 5. Hbase

Hbase命令

============================================================

start-hbase.sh/stop-hbase.sh启动、停止集群

hbase-daemon.sh start/stop master/regionserver

help,version,list,desc 't1'

exists 't1'

is\_enabled 't1',is\_disabled 't1'

count 't1'

创建表：

create 't1', 't1\_id', 'address', 'info'

删除表：disable 't1'

drop 't1'

删除一个列族：disable 't1'

alter 't1',{NAME=>'t1\_id',METHOD=>'delete'}

enable 't1'

插入数据：对应字段：表名，rowkey,列（列族+列），。。。

put 't1','wei','address:country','china'

put 't1','wei','address:province'，'henan'

put 't1','wei','address:city','zhoukou'

put 't1','wei','info:age','27'

put 't1','wei','info:birthday','2015-10-1'

put 't1','wei','info:company','baidu'

获取数据：

get 't1','wei'//获取一行数据

get 't1','wei','address'

get 't1','wei','address:country'

更新数据：虽然有几个版本，默认只会显示最近的新数据

put put 't1','wei','info:age','40'

全表扫描：scan 't1'

删除数据：

delete 't1','wei','info:age'

## 5.1 矩阵相乘

|a b| |e f g| |ae+bh af+bi ag+bk|

|c d| \* |h i k| = |ce+dh cf+di cg+dk|

# 6. MySql

## 6.1 navicat中null与字符串

navicat 中，（默认）值null 的字段，填入了数据再删除保存，该字段由null变为空字符串，用is null 查不到，需用 ='';

## 6.2 Sql语句执行顺序及例子

sql语句写的顺序：select ... from... where.... group by... having... order by.. limit [offset,] (rows)

sql语句执行顺序：from... where...group by... having.... select ... order by... Limit

例子：

select b.areaname,a.num from (select left(acode,2) t ,sum(num) num from starv\_area\_0821 group by t) a ,starv\_area b where concat(a.t,'0000')=b.areacode;

select c.qq,sum(c.ss) from (select b.r qq,b.t,b.y,floor(b.t/2),floor(b.t/2)\*b.y ss from (select a.w r,a.e t,count(\*) y from (select mac q,app w,count(\*) e from starv\_xl\_0825 where app like '%腾讯视频%' and apptype=2 group by mac,app) a group by a.w,a.e) b) c;

update channel\_heilongjiang a,(select id q,concat('000',conv(Tsid,10,16)) r,concat('000',conv(sid,10,16)) t from channel\_heilongjiang) b set a.localcode = lower(concat(right(b.r,4),'\_',right(b.t,4))) where a.id=b.q ;

ceil(7/5)=2;floor(7/5)=1;

## 6.3 Mysql修改最大连接数

Mysql5.6安装目录:C:\ProgramData\MySQL\MySQL Server 5.6\my.ini,修改其中的max\_connections=1000,

再通过界面 Manage Server Instances ——>>System Profile->my.ini路径

linux 对mysql大小写敏感。windows 忽略mysql大小写。数据库从windows导入到linux后，出错。关闭linux对mysql大小写。vi /etc/my.cnf lower\_case\_table\_names =1

mysql登录报错：ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock'

解决办法：ln -s /var/lib/mysql/mysql.sock /tmp/mysql.sock

无法连接的话，进行卸载，重新安装。

## mysql卸载

centos6.7离线安装MySQL:

1）rpm -ivh MySQL-server-5.6.28-1.el6.x86\_64.rpm MySQL-client-5.6.28-1.el6.x86\_64.rpm

rpm -ivh MySQL-devel-5.6.28-1.el6.x86\_64.rpm

(可选 rpm -ivh libaio-0.3.107-10.el6.x86\_64.rpm yum install numactl)

2）service mysql start

3）用户名：root,密码：cat /root/.mysql\_secret

4) mysql -uroot -p随机密码

5)修改密码：set password=password('root');

1)rpm -qa | grep -i mysql

2)rpm -e --nodeps mysql-libs-5.1.52-1.el6\_0.1.x86\_64(自己的版本)

或rpm -ev 包名

3)find / -name mysql

1.删除老版本mysql的开发头文件和库

命令：rm -fr /usr/lib/mysql

rm -fr /usr/include/mysql

2.注意：卸载后/var/lib/mysql中的数据及/etc/my.cnf不会删除，如果确定没用后就手工删除

rm -f /etc/my.cnf

　　 rm -fr /var/lib/mysql

## 6.5 Mysql连接/远程连接权限

进入 use mysql;

delete from user where 1=1; #或删除原来用户名为root的那行记录

## GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY 'hhly123456' WITH GRANT OPTION;

## FLUSH PRIVILEGES;

========================================

mysql -h 192.168.0.147 -uroot -proot linux远程连接mysql。

Centos7安装Mysql,添加远程权限:http://www.linuxidc.com/Linux/2016-09/135288.htm

<http://blog.csdn.net/zhihui1017/article/details/54897498>

Ubuntu远程连接：http://blog.csdn.net/qiyueqinglian/article/details/52778230

## 6.6 MySQL数据目录迁移

在/var/lib/目录下：

1) mv /var/lib/mysql /home 将mysql整个目录移动到home下面。

2) ln -s /home/mysql/ /var/lib/ 为上面的mysql目录创建链接 至 /var/lib/，这时会在/var/lib/下面多一个 mysql -> /home/mysql/。

注意:

1）/var/lib/mysql下的文件权限必须是：mysql:mysql，chown mysql:mysql /var/lib/mysql/ -R

2）可能需要关闭linux防火墙，SElinux，Mysql启动不了，再关闭。

getenforce 查看状态

setenforce 0 暂时设置关闭

/etc/selinux/config 永久关闭

链接：ln -s /home/mysql/ /var/lib == ln -s /home/mysql /var/lib/

一般只针对：目录对目录，文件对文件进行链接。



# 7. Windows

## docs常用命令

netstat -ano 查看进程端口

Notepad 记事本

打开网络连接：ncpa.cpl

远程桌面：mstsc

services.msc 打开服务列表

net share 查看共享文件 及 目录。

在资源管理器中，输入 \\计算机名，就可查看网络共享文件。

专用网络防火墙:netsh firewall set opmode enable/disable

公用网络防火墙：netsh advfirewall set publicprofile state on/off

========================================================================

修改使用ipv4:ping localhost -4 返回 127.0.0.1 ，默认ipv6优先。

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip6\Parameters 下新建 DWORD(32位) :key:DisabledComponents value:ffffffff

## 修改默认浏览器

在控制面板——>默认程序——>修改默认的google浏览器,同时需要修改360中的锁。

## 共享权限配置

http://zhidao.baidu.com/link?url=-kV58e3wnoPtJM9jKHfrZYp4CpuWotZ4cr39pPjKJkidV29ZdCwKzsmNPpHvhUMmSGY9KiFkAZViXumIDehImK

# 8. Tomcat

## 8.1 关闭tomcat进程

管理员身份进入cmd：

netstat -ano | findstr "8080" ，或者 tasklist 找到PID

taskkill /f /pid pid号， /F表示强行终止。

## 8.2 Linux启动Tomcat报错

Linux 启动Tomcat报错，[error] ResourceManager : unable to find resource ‘VM\_global\_library.vm’ in any resource loader.

解决：

直接修改Velocity的jar包里的属性文件即可，方法如下：

1）进入velocity-1.5.jar

2）打开org\apache\velocity\runtime\defaults\velocity.properties

3）修改内容如下：

runtime.log.logsystem.class = org.apache.velocity.runtime.log.NullLogSystem

## 8.3 Tomcat VM与java VM优化

1. **Myeclipse启动时 解决JAVA内存溢出问题：**  **windows-->preference-->MyEclipse-->Servers-->Tomcat-->Tomcatx.x-->JDK  
      在右边的Optional Java VM arguments里加上:-Xms256m -Xmx512m  
      数值可以根据自己的电脑配置自己设定.**
2. **Linux 中直接startup.sh 启动时解决内存溢出：**

**在catalina.sh中239行，添加：**

**JAVA\_OPTS="-server -Xms4096m -Xmx4096m -XX:-UseGCOverheadLimit -XX:PermSize=1024M -XX:MaxPermSize=4096m -Djava.awt.headless=true"**

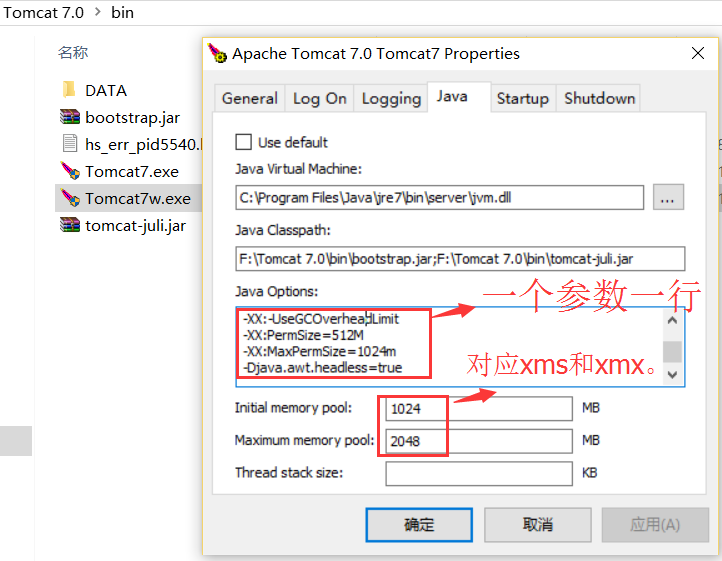
**3.Windows**

**在catalina.bat最前面加入set JAVA\_OPTS=-Xms128m -Xmx350m**

**如果用startup.bat启动tomcat,OK设置生效.**

**3.1 Windows集成版Tomcat**

**在Tomcat7w.exe中设置：**



**4.一个Linux上安装三个tomcat配置方法，在server.xml中修改三个端口即可：**

**tomcat\_r\_hadoop：8005 8080 8009（默认端口）**

**tomcat\_hadoop： 8006 8081 8010**

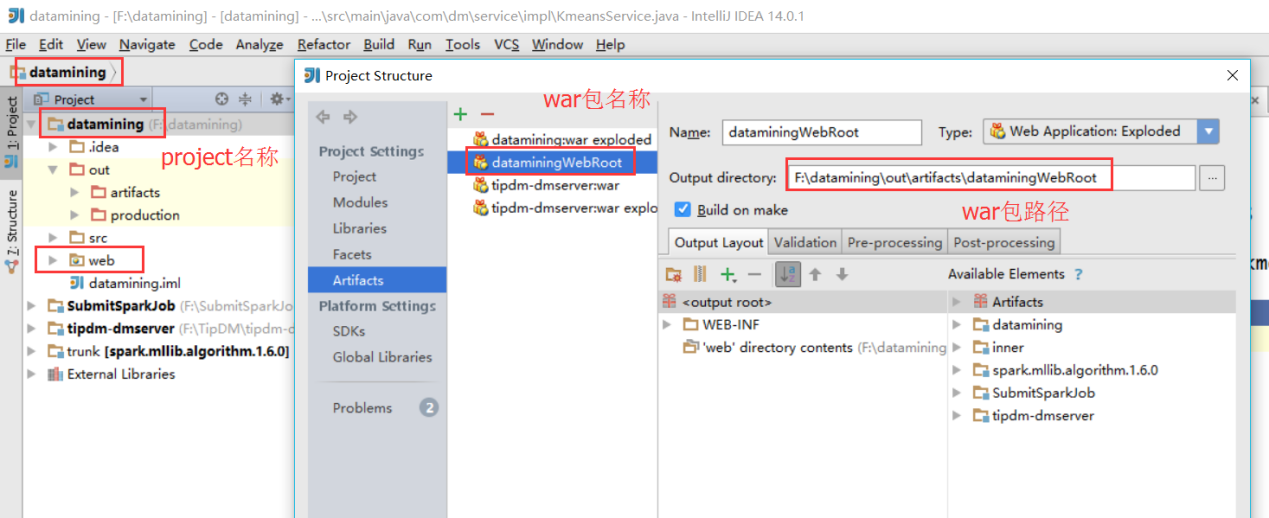
**tomcat\_r： 8007 8082 8011**

**tomcat\_54\_v1： 8008 8083 8012**

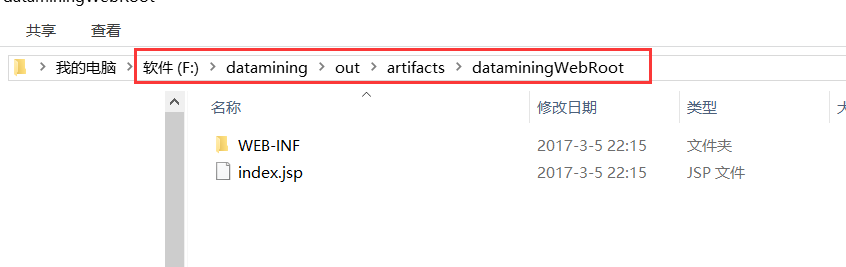
**tomcat\_54\_v2： 8013 8084 8014**

**tomcat\_jingsai 9990 9991 9042**

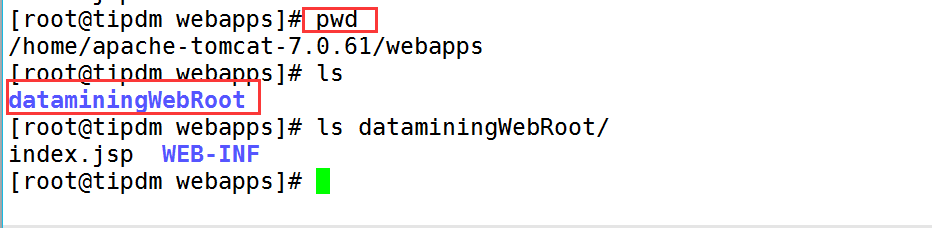
## **8.4 Linux tomcat 部署路径问题**



**建立IDEA项目，Project名称为datamining, build的 war名称是dataminingWebRoot。**

**在Linux中两种部署方式：**

**1）直接将dataminingWebRoot文件夹拷贝到Linux的Tomcat目录下的webapps下,访问路径为：  
tipdm:9991/dataminingWebRoot/index.jsp**

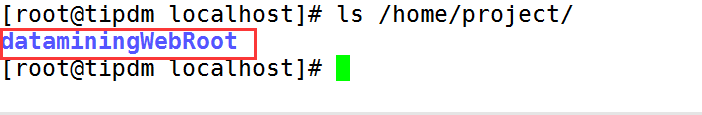
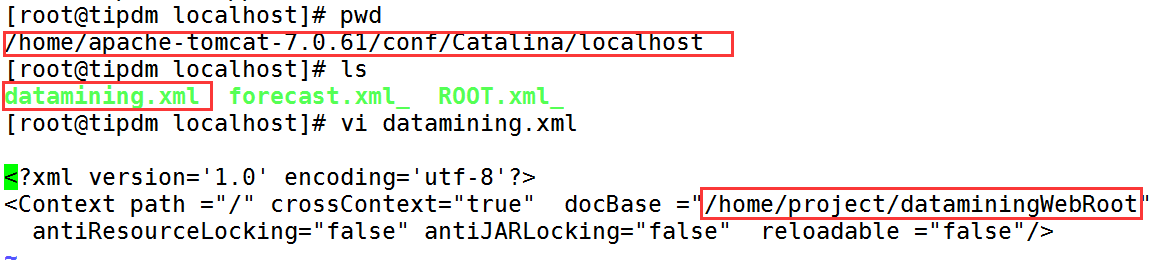


**2)使用配置文件加载WEB项目：**

**1.将dataminingWebRoot文件夹上传到Linux的/home/project目录下。**

**2.** 在/home/apache-tomcat-7.0.61/conf/Catalina/localhost目录下编写datamining.xml，内容指向**/home/project/dataminingWebRoot。**

**3.启动访问：tipdm:9991/datamining/index.jsp。（在tomcat的work目录下可看到对应的文件夹，即访问前缀）**

**总结：两种方式的区别是：访问路径前缀不同。**

1. **配置文件方式前缀为IDEA中该Web项目的Project名称。**
2. **直接将build的war文件夹放到webapps这种部署方式的访问前缀为：webapps下该war文件夹的名称。**

# 9. Hadoop

## 9.1 集群中的四个配置文件

环境变量配置示例：

export HADOOP\_HOME=/usr/local/hadoop/hadoop-2.2.0/

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

Hadoop配置文件================================================================================

### 9.1.1. core-site.xml:

<name>fs.defaultFS</name>

<value>hdfs://node80:8020</value>

### 9.1.2. hdfs-site.xml:

<property>

<name>dfs.namenode.name.dir</name>

<value>file:////data/hadoop/hdfs/name</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:///data/hadoop/hdfs/data</value>

</property>

<property>

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

<value>node80:50090</value>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

### 9.1.3. mapred-site.xml:

<property>

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

<value>yarn</value>

</property>

<property>

<name>mapreduce.jobhistory.address</name>

<value>node80:10020</value>

</property>

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>node80:19888</value>

</property>

### 9.1.4. yarn-site.xml:

<name>yarn.resourcemanager.hostname</name>

<value>node80</value>

</property>

<property>

<description>The address of the applications manager interface in the RM.</description>

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

<value>${yarn.resourcemanager.hostname}:8032</value>

</property>

<property>

<description>The address of the scheduler interface.</description>

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

<value>${yarn.resourcemanager.hostname}:8030</value>

</property>

<property>

<description>The http address of the RM web application.</description>

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

<value>${yarn.resourcemanager.hostname}:8088</value>

</property>

<property>

<description>The https adddress of the RM web application.</description>

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

<value>${yarn.resourcemanager.hostname}:8090</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8031</value>

</property>

<property>

<description>The address of the RM admin interface.</description>

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

<value>${yarn.resourcemanager.hostname}:8033</value>

</property>

<property>

<description>List of directories to store localized files in. An

application's localized file directory will be found in:

${yarn.nodemanager.local-dirs}/usercache/${user}/appcache/application\_${appid}.

Individual containers' work directories, called container\_${contid}, will

be subdirectories of this.

</description>

<name>yarn.nodemanager.local-dirs</name>

<value>/data/hadoop/yarn/local</value>

</property>

<property>

<description>Whether to enable log aggregation</description>

<name>yarn.log-aggregation-enable</name>

<value>true</value>

</property>

<property>

<description>Where to aggregate logs to.</description>

<name>yarn.nodemanager.remote-app-log-dir</name>

<value>/data/tmp/logs</value>

</property>

<property>

<description>the valid service name should only contain a-zA-Z0-9\_ and can not start with numbers</description>

<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.log.server.url</name>

<value>http://node80:19888/jobhistory/logs/</value>

</property>

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

<description>Whether virtual memory limits will be enforced for containers</description>

</property>

### 9.1.5. hadoop-env.sh:

export JAVA\_HOME=/opt/jdk1.8.0\_51

## 9.2. eclipse 中运行mapreduce 程序，java程序配置

1、Configuration config = new Configuration();

config.set("fs.defaultFS", "hdfs://node1:8020");

config.set("yarn.resourcemanager.hostname", "node1");

config.set("mapred.jar", "C:\\Users\\bc\\Desktop\\test2.jar");

2、前提：把core-site.xml,hdfs-site.xml,mapred-site.xml,yarn-site.xml放到src下：

Configuration config = new Configuration();

config.set("mapred.jar", "C:\\Users\\bc\\Desktop\\test2.jar");

3:在集群上运行，hadoop jar jar包

Configuration config = new Configuration();

总之三种方法，都需要jar包，都需要指定配置文件。

## 9.3. hadoop jar/java -jar执行jar包，已指定主类

hadoop jar test2.jar

java -jar format.jar

## 9.4. HDFS

mapreduce 中，路径默认是从 /user/root/ 开始的。

strInputPath = "DATA/4028aae95063f81401506407c8e40004/\*"

pOutputPath = "ALGORITHM/0208/input"

FileInputFormat.addInputPath(job, new Path(strInputPath));

FileOutputFormat.setOutputPath(job, pOutputPath);

以上写法，mapreduce会自动从/user/root/DATA/4028aae95063f81401506407c8e40004/\*读取，

向/user/root/ALGORITHM/0208/input写入。

Runnable向spark算法中传入时，要转一下路径为hdfs上的路径。

使用Eclipse在非hadoop运行的用户下进行写入hdfs文件系统中时，由于sunqw对"/"目录没有写入权限，

所以导致异常的发生。解决方法即开放hadoop中的HDFS目录的权限，命令如下：hadoop fs -chmod 777 / 。

新建的用户/opt 没有足够的权限，程序拒绝访问。

## 9.5. Hadoop程序运行，ToolRunner

ToolRunner完成以下2个功能：

（1）为Tool创建一个Configuration对象。

（2）使得程序可以方便的读取参数配置。

关于ToolRunner典型的实现方法

1、定义一个类（如上图中的MyClass），继承configured，实现Tool接口。

2、在main()方法中通过ToolRunner.run(...)方法调用上述类的run(String[]方法）

它们均是静态方法，即可以通过类名调用。

（1）public static int run([Configuration](http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/conf/Configuration.html" \t "http://blog.csdn.net/jediael_lu/article/details/_blank" \o "class in org.apache.hadoop.conf) conf,[Tool](http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/util/Tool.html" \t "http://blog.csdn.net/jediael_lu/article/details/_blank" \o "interface in org.apache.hadoop.util) tool, [String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true" \t "http://blog.csdn.net/jediael_lu/article/details/_blank" \o "class or interface in java.lang)[] args)

这个方法调用tool的run(String[])方法，并使用conf中的参数，以及args中的参数，而args一般来源于命令行。

（2）public static int run([Tool](http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/util/Tool.html" \t "http://blog.csdn.net/jediael_lu/article/details/_blank" \o "interface in org.apache.hadoop.util) tool, [String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true" \t "http://blog.csdn.net/jediael_lu/article/details/_blank" \o "class or interface in java.lang)[] args)

这个方法调用tool的run方法，并使用tool类的参数属性，即等同于run(tool.getConf(), tool, args)。

ToolRunner中有两个run方法。

Tool自身只有一个方法run(String[])。

public class RandomForestTestMRJob extends Configured implements Tool

int ret = ToolRunner.run(HadoopUtils.getConf(), new RandomForestTestMRJob(), args);

## 9.6. MapReduce程序测试，MRTest.java

例如：RandomForest算法，MRTest中测试，只需要上传下面的几个类：

1. **public** **class** RandomForestTestMRJob **extends** Configured **implements** Tool中的RandomForestTestMRJob 类。
2. 上面的类用到的Map和Reduce类。如**public** **class** RandomForestTestMapper **extends** Mapper<LongWritable,Text,Text,NullWritable> 中的RandomForestTestMapper 类。

## 9.7 Sqoop安装及hadoop-yarn-common-2.2.0.jar修改

1. 下载sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha.tar.gz文件；

2. 解压到/usr/local/sqoop目录；tar -zxf sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha.tar.gz

3. 配置/etc/profile文件

export SQOOP\_HOME=/usr/local/sqoop/sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha

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

export HADOOP\_HOME=/usr/local/hadoop/hadoop-2.2.0

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

添加修改以下内容：否则无sqoop无法提交任务，node80:19888日志:Error: Could not find or load main class org.apache.hadoop.mapreduce.v2.app.MRAppMaster

(1) vi /etc/profile ，添加：export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

(2) 修改/opt/hadoop-2.2.0/share/hadoop/yarn/hadoop-yarn-common-2.2.0.jar中的yarn-default.xml文件，

<name>yarn.application.classpath</name>中添加$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*,$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*

4. 在 $SQOOP\_HOME/lib目录，并把mysql驱动拷贝到此文件夹中；

错误：

1. mysql-connector-java-5.1.32-bin.jar， 要使用此版本+，

否则出错：

java.io.IOException: No columns to generate for ClassWriter

下载地址 http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.32.tar.gz

2. 权限

hadoop fs -chmod -R 777 /tmp

注意：

1. sqoop所在的机器需要是一个hadoop的节点，同时需要配置：

export HADOOP\_HOME=/usr/local/hadoop/hadoop-2.2.0

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export SQOOP\_HOME=/usr/local/sqoop/sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha

2. 所配置的数据源需要在sqoop所在的机器访问，所以sqoop所在的机器可以访问数据源（针对caic工程必须）；

# Java

## 10.1. GB2312/GBK

GB2312是中国规定的汉字编码，也可以说是简体中文的字符集编码;GBK 是 GB2312的扩展 ,除了兼容GB2312外，它还能显示繁体中文，还有日文的假名

## 10.2. 输入流/输出流

先关闭处理流，再关闭节点流。

br.close();isr.close();fis.close();

bw.close(); osw.close();fos.close();

=====================================

HibernateDmWebSMJobDaoImpl dmWebSMJobDao = (HibernateDmWebSMJobDaoImpl)SpringUtil.getBean("dmWebSMJobDao");

dmWebSMJob = dmWebSMJobDao.findUniqueBy("jobId", dmCloudJob.getJobId());

## 10.3. 加密类：com.caic.pub.util.DESUtils

public static void main(String[] args) {

String name = "root";

String password = "root";

String encryname = getEncryptString(name);

String encrypassword = getEncryptString(password);

System.out.println(encryname);

System.out.println(encrypassword);

System.out.println(getDecryptString(encryname));

System.out.println(getDecryptString(encrypassword));

}

## 10.4. Java环境变量

classpath只跟java运行有关，跟编译无关。

CLASSPATH =.;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar

jdk1.6以上就不需要配置classpath了；系统会自动帮你配置好； 您只需要配置JAVA\_HOME和Path就可以了 例如我的配置:

JAVA\_HOME C:\Program Files\Java\jdk1.7.0\_13

PATH %JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;

## 10.5. 转义

"|"在Java中是特殊字符，如果不加“\\”就按特殊含义处理，加上转义字符“\\”就说明“|”是字符串而不是特殊字符。

'\'在Java中是特殊字符, 需要进行转义后才能识别出来;而且“\”本身就代表转义的意思，“\\”当然就是把一个"\"转成真的意义上的“\”。

# 11. 常见英文单词

explained variance 解释方差

evaluate 评价

compute 计算，推断。

metric 度量标准;

accuracy 准确性。

algorithm 算法; 计算程序，英[ˈælgərɪðəm]

fault 错误。

quantile 分位数; 分位点;

variance 方差;

compute 计算;

covariance 协方差;

correlation 相关系数。

# 12. R语言基础

help(plot)

R.home()安装目录

getwd()工作空间

setwd("F://Workspace\_R")

search() 已经load进来的包

.packages(all.available = TRUE) R知道所有的包

install.packages("RODBC") 安装包

library(RODBC) load包。

help(plot) == ?plot 函数帮助

library(help=RODBC) 包帮助

source("./111111.R") 执行R文件

search()

options() 系统环境变量

ls() 查看内存中的变量

列表》数据框》数组》矩阵》向量

data <- 123456789 为data赋值

save(data,file = "d://zz") 将内存中变量data写入磁盘。

save.image(file="d://myproject") 将工作空间 写入磁盘。

(a = "hello,world!") 赋值并输出。

read.table("t1.txt",header = TRUE,sep = "") 文件要求：从excel中保存为 制表符分割的txt文件。

read.csv("t2.csv",header = TRUE,sep = ",/;") 文件要求：从excel中保存为逗号分隔的csv文件。

conn <- odbcConnect("mydsn",uid = "root",pwd = "root") 建立连接。

data("USArrests") 数据加载到内存

sqlSave(conn,USArrests,rownames = "country",addPK = TRUE) 保存数据到数据库中。

sqlDrop(conn,"usarrests") 删除数据库表

rm(USArrests) 从内存删除数据集

sqlQuery(conn,"select \* from USArrests limit 10") 查询

sqlFetch(conn,"USArrests",rownames = "country")

close(conn) 关闭连接

saledata[!complete.cases(saledata),]$"销量" 不写列，就默认所有列。

pdf() 和 png()输出图形

var() 求方差 s^2=[(x1-x)^2 +...(xn-x)^2]/n

sd() 求标准差 s=sqrt(s^2)

cov() 求协方差

cor() 求相似度。

方差分为population variance总体方差和sample variance样本方差，

区别是总体方差除以N，样本方差除以N-1。

1）定义用户数组（向量）

user1<-c(5.0, 3.0, 2.5)

user5<-c(4.0, 3.0, 2.0)

2）计算方差

var(user1)=sum((user1-mean(user1))^2)/(3-1)=1.75

var(user5)=sum((user5-mean(user5))^2)/(3-1)=1

3）计算标准差

sd(user1)=sqrt(var(user1))=1.322876

sd(user5)=sqrt(var(user5))=1

4）计算协方差

cov(user1, user5)

=sum((user1-mean(user1))\*(user5-mean(user5)))/(3-1) = 1.25 注意：R中cov()除以N-1。

5）计算相似度

cor(user1, user5)

=cov(user1, user5) / (sd(user1)\*(sd(user5))) = 0.9449112

# 13. 算法Service\_页面\_参数

Spark K均值算法

Spark SVM算法

Spark 线性回归

Spark 逻辑回归

Spark朴素贝叶斯算法

Spark SVD 算法

Spark PCA算法

Mahout 协同过滤算法

Mahout RandomForest算法（随机森林算法）

Mahout 模糊聚类算法

1、 无test

CloudFuzzyKmeansService Mahout-FuzzyKMeans算法 /forecast/flex/MahoutFuzzyKmeans.html

train(Long jobId, DmScheme scheme, String distanceMeasure, int maxIteration,

double clusterClassificationThreshold, double convergenceDelta,

int clusterNumbers,float fuzzyness)

1）距离计算方法 distanceMeasure ：

{label:"欧氏距离", data:"EuclideanDistanceMeasure"},

{label:"欧氏距离(不采取平方根)", data:"SquaredEuclideanDistanceMeasure"},

{label:"马氏距离", data:"ManhattanDistanceMeasure"},

{label:"余弦距离", data:"CosineDistanceMeasure"},

{label:"系数距离", data:"TanimotoDistanceMeasure"}

2） 最大迭代次数：maxIteration 3

3）阈值：clusterClassificationThreshold 0.5

4）收敛系数：convergenceDelta 0.5

5）聚类数：clusterNumbers 3

6）模糊参数：fuzzyness 2

2、无test

dmCloudRecommenderService Mahout-Collaborative\_Filtering算法 /forecast/flex/MahoutRecommender.html

train(Long jobId, DmScheme scheme,int numrecommend,int maxPrefsPerUser,

int minPrefsPerUser, int maxSimilaritiesPerItem,int maxPrefsInItemSimilarity,

String similarityClassname)

1）推荐个数：numrecommend 3

2）每用户最大参考阈值 maxPrefsPerUser 10

3）每用户最小参考阈值 minPrefsPerUser 1

4）每项目最大相似个数 maxSimilaritiesPerItem 100

5）最大相似个数阈值 maxPrefsInItemSimilarity 50

6)相似距离 similarityClassname :

{name:"SIMILARITY\_EUCLIDEAN\_DISTANCE",label:'欧氏距离'},

{name:"SIMILARITY\_COSINE",label:'余弦距离'},

{name:"SIMILARITY\_PEARSON\_CORRELATION",label:'皮尔斯距离'},

{name:"SIMILARITY\_COOCCURRENCE",label:'共生距离'},

{name:"SIMILARITY\_LOGLIKELIHOOD",label:'对数系数距离'},

{name:"SIMILARITY\_TANIMOTO\_COEFFICIENT",label:'广义Jaccard系数距离'},

{name:"SIMILARITY\_CITY\_BLOCK",label:'块距离'}

3、

cloudRandomForestService Mahout-RandomForest算法 /forecast/flex/MahoutRandomForest.html

train(Long jobId, String fielddesc, int nrtrees, int m)

1)描述字符串 fielddesc "I 10 N L"

2)决策树数目 nrtrees 5

3)选择属性个数 m 3

4、

dmSparkKmeansService Spark-Kmeans算法 /forecast/flex/SparkKmeans.html

train(Long jobId, DmScheme scheme, int maxIteration,int clusterNumbers)

1）最大迭代次数：maxIteration 3

2）聚类数：clusterNumbers 3

5、dmSparkNaiveBayesService Spark--NaiveBayes算法 /forecast/flex/SparkNaiveBayes.html

train(Long jobId, String labelIndex, Double lambda,Double trainPercent,Long seed)

1）输出下标 labelIndex

{id:"first",label:'在输入前'},{id:"last",label:'在输入后'}

2）平滑指数 lambda 1.0

3）训练比例 trainPercent 0.8

4）随机种子 seed 11

6、dmSparkLinearRegressionService Spark-LinearRegression算法 /forecast/flex/SparkLinearRegression.html

train(Long jobId, String labelIndex, Integer iteration,Double trainPercent,

Long seed,Double stepSize,Double miniBatchFraction)

1）输出下标 labelIndex

{id:"first",label:'在输入前'},{id:"last",label:'在输入后'}

2）循环次数 iteration 100

3）训练比例 trainPercent 0.8

4）随机种子 seed 11

5）梯度下降步长 stepSize 1.0

6）循环数据比例 miniBatchFraction 1.0

7、dmSparkLogisticRegressionService Spark-LogisticRegression算法 /forecast/flex/SparkLogisticRegression.html

train(Long jobId, String labelIndex, Integer iteration,Double trainPercent,Long seed,Double stepSize,Double miniBatchFraction)

1）输出下标 labelIndex

{id:"first",label:'在输入前'},{id:"last",label:'在输入后'}

2）循环次数 iteration 100

3）训练比例 trainPercent 0.8

4）随机种子 seed 11

5）梯度下降步长 stepSize 1.0

6）循环数据比例 miniBatchFraction 1.0

8、无test

dmSparkPCAService Spark-PCA分析 /forecast/flex/SparkPCA.html

train(Long jobId, int k)

1）主成分个数 k 3

9、无test

dmSparkSVDService Spark-SVD分析 /forecast/flex/SparkSVD.html

train(Long jobId, int k)

1)奇异值个数 k 3

10、dmSparkSVMService Spark-SVM算法 /forecast/flex/SparkSVM.html

train(Long jobId, String labelIndex, Integer iteration,Double trainPercent,Long seed,Double stepSize,Double regParm,

Double miniBatchFraction)

1）输出下标 labelIndex

{id:"first",label:'在输入前'},{id:"last",label:'在输入后'}

2）循环次数 iteration 100

3）训练比例 trainPercent 0.8

4）随机种子 seed 11

5）梯度下降步长 stepSize 1.0

6)规范化系数 regParm 0.01

7）循环数据比例 miniBatchFraction 1.0

11、无test

DmCloudPreprocessNullHandleService空值处理 /forecast/flex/HadoopNullHandle.html

train(Long jobId, DmScheme scheme,String insertStrategy)

1)插值策略 insertStrategy:

{name:"mean",label:'平均值插值'},

{name:"lagrangian",label:'拉格朗日'},

{name:"newton",label:'牛顿插值'}

12、无test

dmCloudPreprocessTimeSeriesInsertService 时间序列插值 /forecast/flex/HadoopTimeSeriesPreprocess.html

train(Long jobId, DmScheme scheme,String insertStrategy,String format,

String selectedTimeCol,String timeType,String timeInterval)

1)插值策略 insertStrategy

{name:"mean",label:'平均值插值'},

{name:"lagrangian",label:'拉格朗日'},

{name:"newton",label:'牛顿插值'}

2)时间格式 format：yyyy/MM/dd HH:mm:ss:yyyy/MM/dd HH:mm:ss,yyyy年MM月dd日 HH时mm分ss秒:yyyy年MM月dd日 HH时mm分ss秒,yyyy-MM-dd HH:mm:ss:yyyy-MM-dd HH:mm:ss,yyyyMMddHHmmss:yyyyMMddHHmmss

yyyy/MM/dd-all:yyyy/MM/dd-all,yyyy年MM月dd日-all:yyyy年MM月dd日-all,yyyy-MM-dd-all:yyyy-MM-dd-all,yyyyMMdd-all:yyyyMMdd-all

{name:"yyyy/MM/dd HH:mm:ss",label:'yyyy/MM/dd HH:mm:ss'},

{name:"yyyy年MM月dd日 HH时mm分ss秒",label:'yyyy年MM月dd日 HH时mm分ss秒'},

{name:"yyyy-MM-dd HH:mm:ss",label:'yyyy-MM-dd HH:mm:ss'},

{name:"yyyyMMddHHmmss",label:'yyyyMMddHHmmss'}

3)时间字段 selectedTimeCol 0

4)时间类型 timeType

{name:"SS",label:'秒'},

{name:"MI",label:'分钟'},

{name:"H",label:'小时'},

{name:"D",label:'日'},

{name:"W",label:'周'},

{name:"M",label:'月'},

{name:"Y",label:'年'}

5)时间间隔 timeInterval 5

13、 无test

dmCloudPreprocessSelectRecordsService 记录选择 /forecast/flex/HadoopSelectRecord.html

train(Long jobId, DmScheme scheme,String columnIndexes,String operators,String conditions)

1）列输入 columnIndexes 1,3

2）关系选择 operators equals,>:可以选择equals,contains,notequals,>,<,=,>=,<=,!=

3）条件输入 conditions BGB,10.65

14、无test

dmCloudPreprocessHandleAbnormalService 异常值处理 /forecast/flex/HadoopHandleAbnormal.html

train(Long jobId, DmScheme scheme,String handleAbnormalMethod,String handleAbnormalValue,String selectRecords)

1）异常值处理策略 handleAbnormalMethod

{name:"box",label:'箱型法'},

{name:"threedelta",label:'正态分布法'}

2）异常值处理方式 handleAbnormalValue

{name:"empty",label:'置空'},

{name:"delete",label:'删除'}

3）选择记录数(100~1000) selectRecords 100

15、无test

dmCloudPreprocessDiscretizationService 离散化处理 /forecast/flex/HadoopDiscretization.html

train(Long jobId, DmScheme scheme,String method,String group,String selectRecords)

1）离散化策略 method

{name:"equi\_frequency",label:'等频法'},

{name:"equi\_distance",label:'等距法'}

2）组数(1~n) group 2

3）选择记录数(100~1000) selectRecords 100

16、dmCloudPreprocessNullHandleService 空值处理 (牛顿、拉格朗日、平均插值法) /forecast/flex/HadoopNullHandle.html

train(Long jobId, DmScheme scheme,String insertStrategy)

1)插值策略 insertStrategy:

{name:"mean",label:'平均值插值'},

{name:"lagrangian",label:'拉格朗日'},

{name:"newton",label:'牛顿插值'}

# 14. Spark运行命令样例

1、local单机部署模式：

结果xshell可见：

./bin/spark-submit --class org.apache.spark.examples.SparkPi --master local[1] ./lib/spark-examples-1.3.1-hadoop2.4.0.jar 100

2、standalone集群部署模式：

driver在哪个机器上运行，执行进度可在(node1,2,3:4040)中看到，客户端模式执行，driver在本地node1上,没有提交到集群上，

node1:8080可以看到上面没有对应的driver, 先执行driver,在执行application，两者都有对应的id,提交一个程序到

集群上后，一般driver,application不在同一个节点上执行，driver提交到哪个节点，结果就在哪个节点。每个节点上

都记录了他一共被提交了多少个driver，状态如何等信息。

1)client运行模式：

driver在本地执行，结果在本地显示。

结果xshell可见：

./bin/spark-submit --class test6.SparkPi2 --master spark://node1:7077 --executor-memory 2G --total-executor-cores 4 /root/t1.jar 100

2)cluster运行模式：

结果node1:8080里面可见！

./bin/spark-submit --class test6.SparkPi2 --master spark://node1:7077 --deploy-mode cluster --supervise --executor-memory 1G --total-executor-cores 1 /root/t1.jar 100

3、Yarn集群部署模式：需要启动start-yarn.sh

1)client运行模式：

结果xShell里面可见！

./bin/spark-submit --class test6.SparkPi2 --deploy-mode client --executor-memory 1G --num-executors 1 /root/t1.jar 100

2)cluster运行模式：

结果node1:8088里面可见！在resourcemanager界面可以看到程序运行。

./bin/spark-submit --class test6.SparkPi2 --master yarn-cluster --executor-memory 1G --num-executors 1 /root/t1.jar 100

# 15. Hive

1、启动hadoop集群：

start-all.sh

2、修改mysql远程连接配置：

1）检查mysql服务是否启动，没有则启动mysql服务：service mysqld start

2)进入mysql数据库：mysql -uroot -p root

3)use mysql;

4)修改远程连接权限：

GRANT ALL ON \*.\* TO 'root'@'%' IDENTIFIED BY 'root' WITH GRANT OPTION;

delete from user where host!="%";

flush privileges;

3、启动hive：hive

===============================================================================================

Hive 的启动方式

hive 命令行模式，直接输入/hive/bin/hive的执行程序，或者输入 hive --service cli， 用于linux平台命令行查询，查询语句基本跟mysql查询语句类似

hive web界面的启动方式，hive --service hwi ，用于通过浏览器来访问hive，提供基本的基于web的hive查询服务，可以看作是hive数据平台的demo

hive 远程服务 (端口号默认10000) 启动方式，hive --service hiveserver2

hive 远程后台启动(关闭终端hive服务不退出): nohup hive --service hiveserver2 &

用java等程序实现通过jdbc等驱动的方式访问hive就用这种起动方式了，这个是程序员最需要的方式了,

default是hive中默认的数据库，进入hive,show databases,可查看。

当hive使用mysql作为元数据库的时候mysql的字符集要设置成latin1。

为了保存那些utf8的中文，要将mysql中存储注释的那几个字段的字符集单独修改为utf8。

修改字段注释字符集

alter table COLUMNS modify column COMMENT varchar(256) character set utf8;

修改表注释字符集

alter table TABL\_PARAMS modify column PARAM\_VALUE varchar(4000) character set utf8;

最后别忘记修改hive的jdbc连接，将字符集修改为utf8，如

javax.jdo.option.ConnectionURL

jdbc:mysql://192.168.0.128:3306/hive?characterEncoding=UTF-8

1. 以下是hive常用命令:

Mysql数据操作

添加：INSERT INTO [表名] VALUES(值1，值2,...);

查询: SELECT \* FROM [表名] WHERE ([条件]);

建立索引:CREATE INDEX [索引文件名] ON [表名] ([字段名]);

删除：DELETE FROM [表名] WHERE ([条件]);

修改：UPDATE [表名] SET [修改内容如name = 'Mary'] WHERE [条件]；

数据库操作：

查看库：show databases;

创建库：create database 库名;

删除库: drop database 库名;

使用库：use 库名;

删除表所有数据：delete from 表名；

Hive表操作：

查看表：show tables;

删除表: drop table 表名;

desc 表名；

create table 新表 like 旧表（复制表的结构）

加载数据：

2）数据在linux本地，表无分区(先将所有数据导入student表)：load data local inpath '/root/test1.txt' overwrite into table test1;

3) 按日期字段查询，并导入对应的日期文件目录下：

insert overwrite local directory '/opt/2015-09-01' row format delimited fields terminated by ',' select \* from student where day='2015-09-01';

insert overwrite local directory '/opt/2015-09-02' row format delimited fields terminated by ',' select \* from student where day='2015-09-02';

insert overwrite local directory '/opt/2015-09-03' row format delimited fields terminated by ',' select \* from student where day='2015-09-03';

### 3）数据在hdfs上,同时表有分区：load data inpath '/user/test\_part.txt' overwrite into table test\_part partition (month='2010-01');

load data local inpath '/opt/2015-09-02/000000\_0' overwrite into table student\_part partition (day='2015-09-02');

load data local inpath '/opt/2015-09-03/000000\_0' overwrite into table student\_part partition (day='2015-09-03');

2、原始数据在hdfs上：

创建表：1)创建表(无分区)：create table student\_hdfs (id string,name string,age int,phone string,day string) row format delimited fields terminated by ',' stored as textfile;

2)创建表(有分区)：create table student\_part\_hdfs (id string,name string,age int,phone string) partitioned by (day string) row format delimited fields terminated by ',' stored as textfile ;

1)上传数据到hdfs：

hadoop fs -put /opt/student\_all.txt /user

2）先将数据导入student\_hdfs表:load data inpath '/user/student\_all.txt' overwrite into table student\_hdfs;

3) 按日期字段查询，并导入数据到对应的hdfs日期文件目录下：

insert overwrite directory '/user/2015-09-01' row format delimited fields terminated by ',' select \* from student\_hdfs where day='2015-09-01';

insert overwrite directory '/user/2015-09-02' row format delimited fields terminated by ',' select \* from student\_hdfs where day='2015-09-02';

insert overwrite directory '/user/2015-09-03' row format delimited fields terminated by ',' select \* from student\_hdfs where day='2015-09-03';

4)数据在hdfs上,同时表有分区：load data inpath '/user/2015-09-01/000000\_0' overwrite into table student\_part\_hdfs partition (day='2015-09-01');

load data inpath '/user/2015-09-02/000000\_0' overwrite into table student\_part\_hdfs partition (day='2015-09-02');

load data inpath '/user/2015-09-03/000000\_0' overwrite into table student\_part\_hdfs partition (day='2015-09-03');

## 15.1 Hive中日期时间，时间戳转换

从1970-01-01 00:00:00 UTC到指定时间的秒数。

总结：时间戳到日期时间，日期时间到时间戳，日期时间到日期。

获取时间戳：select distinct unix\_timestamp() from xl;

时间戳>>>>日期：select distinct from\_unixtime(1441565203,'yyyy/MM/dd HH:mm:ss') from xl;

日期时间>>>>>>时间戳：默认格式为“yyyy-MM-dd HH:mm:ss“：

select distinct unix\_timestamp('2015-09-07 02:46:43') from xl;

指定日期时间>>>>>>时间戳:

select distinct unix\_timestamp('20111207 13:01:03','yyyyMMdd HH:mm:ss') from xl;

日期时间>>>>>>日期:

select distinct to\_date('2011-12-08 10:03:01') from xl;

结果：2011-12-08

日期时间>>>>>>年/月/日/时/分/秒: year(string date)，month(),day(),hour(),minute(),second()。

select distinct year('2011-12-08 10:03:01') from xl;

结果：2011

Date date = new Date();

SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");

Date dt = sdf.parse("20150828012628");

System.out.println(dt);

SimpleDateFormat sdf2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

System.out.println(sdf2.format(dt));

## 16. Pig

copyFromLocal /root/labs/demos/pigdemo.txt demos/

copyFromLocal /root/labs/demos/ demos/

employees2 =load 'pigdemo.txt' as (state，name);//employees后加空格。

salaries = LOAD 'salary.data' USING PigStorage('，') AS (gender， age， income， zip);。

describe employees;

DUMP employees;

ca\_only = FILTER employees BY (state=='CA');

DUMP ca\_only;

emp\_group = GROUP employees2 BY state;

describe emp\_group;

DUMP emp\_group;

=========================

STORE emp\_group INTO 'emp\_group';

STORE emp\_group INTO 'emp\_group\_csv' USING PigStorage('，');

encode/decode 编码/解码

parallel，并行。

Substitution，替换，替代，引用。

guaranteed，有保证的。

refer to，参考; 指的是，引用

syntax，语法; 句法;

aliases，别名。

alias，别名。

trigger，引发，触发；

pig数据类型：

tuple 元组，数组

bag 袋子；

nested：嵌套的；

clause：从句，分句。

数据类型：

int

• long

• float

• double

• chararray

• bytearray

• boolean

• datetime

• bigdecimal

• biginteger

Tuple:"ordered"set"of"values"

(OH，Mark，Twain，31225)"

• Bag:"unordered"collecIon"of"tuples"

{"

" "(OH，Mark，Twain，31225)，"

(UK，Charles，Dickens，42207)，"

(ME，Robert，Frost，11496)""

}"

• Map:"collecIon"of"key"value"pairs"

[state#OH，name#Mark"Twain，zip#31225]

嵌套：

(CA，{(CA，Ulf)，(CA，manish)，(CA，Brian)})

==============

customers = LOAD 'customer\_data' AS (firstname:

chararray，lastname:chararray，house\_number:int，

street:chararray，phone:long，payment:double);

字段不够或多余：补充null，或截断。

corresponding：对应的；

=============================

group:

1.salaries = LOAD 'salaries.txt' USING PigStorage('，') AS

(gender:chararray， age:int，salary:double，zip:int);

2.salariesbyage = GROUP salaries BY age;

3.DUMP salariesbyage;

(17，{(F，17，0.0，95050)，(M，17，0.0，95103)，(M，17，0.0，95102)})

mygroup = GROUP salaries BY (gender，age);

describe mygroup -> {group: (gender: chararray，age: int)，salaries: {(gender: chararray，age: int，salary: double，zip: int)}}

其中mygroup有两个字段，名字分别为：group，salaries;

> dump mygroup;

((M，17)，{(M，17，0.0，95103)，(M，17，0.0，95102)})

((M，19)，{(M，19，0.0，95050)})

group all:

salaries\_group = GROUP salaries ALL;

2016-4-26:==================================================================

pig入门博客：http://blackproof.iteye.com/blog/1791980

pig几种执行模式：http://www.open-open.com/lib/view/open1385173242385.html

pig –x mapreduce script.pig

Important:

Field names and relation names are case sensitive in your Pig

Latin scripts. User Defined Functions (UDFs) are also case sensitive.

However， Pig Latin keywords (like LOAD and AS) are not case sensitive.

1.a = load a0 as dump a employees = LOAD 'pigdemo.txt' AS (state， name); 默认的分隔符为'\t'，pigdemo.txt数据分隔符为'\t'，因此不需指定。

2.b = filter a by dump b ca\_only = FILTER employees BY state=='CA';

3.c = group b by dump c emp\_group = GROUP employees BY state;

4.store c into d d默认位置/user/root， STORE emp\_group INTO 'emp\_group';

5.store c into e using PigStorage('，'); 使用，分隔。STORE emp\_group INTO 'emp\_group\_csv' USING PigStorage('，');

6.f = load 'mydata.txt' using TextLoader(); TextLoader simply creates a tuple for each line of text ， and it uses a

single chararray field that contains the entire line. TextLoader()不管每行格式如何，都可加载。describe f 可查看f没有结构。

7.salaries = load 'salary.data' using PigStorage('，') as (gender， age， income， zip);

8.highsalaries = FILTER salaries BY income > 1000000;

以下指定字段类型：（如果不指定数据类型，默认为bytearray）

9.customers = LOAD 'customer\_data' AS (firstname:

chararray，lastname:chararray，house\_number:int，

street:chararray，phone:long，payment:double);

10.指定复杂类型：

数据例如：

F，58，{(95000，95103，(60000，95105)}

M，85，{(14000，95102)，(0，95105)，(2000，94040)}

语句：

salaries = LOAD 'bag\_demo.txt' AS (gender:chararray，

age:int， details:bag{b:(salary:double，zip:long)});

注意：b是在details中的元组的名称。

1———>7———>9——>，加分隔符，字段指定类型，字段指定复杂类型。

A = load 'whitehouse\_visits.txt' using TextLoader ();

A\_limit = limit A 10;

B = LOAD '/user/root/whitehouse/visits.txt' USING

PigStorage(',') AS (

lname:chararray,

fname:chararra,

mname:chararray,

id:chararray,

status:chararray,

state:chararray,

arrival:chararray

);

注意：visits.txt中，虽然数据有数字，日期，字符串等，但统一指定字段类型为chararray，也可加载到B中，而hive就没有chararray这样的类型。

这样的数据hive就无法加载到表里面。

store B into 'whouse\_tab' using PigStorage('\t');

store B into 'whouse\_json' using JsonStorage();

Relations without a Schema：

If you do not define a schema， then the fields of a relation are specified using an index

that starts at $0。

salaries = LOAD 'salaries.txt' USING PigStorage('，');

salariesgroup = GROUP salaries BY $3;

FOREACH...GENERATE：

The FOREACH operator is a great tool for transforming your data into different data

sets. The expression in a FOREACH can contain fields， constants， mathematical

expressions， the result of invoking a Pig function， and many other variations and

nestings.

salaries = LOAD 'salaries.txt' USING PigStorage('，') AS

(gender:chararray， age:int，salary:double，zip:int);

A = FOREACH salaries GENERATE age， salary;

B = FOREACH salaries GENERATE salary， salary \* 0.07;

B1= FOREACH salaries GENERATE zip，salary \* 0.10 AS bonus; //取别名。

B2= foreach salaries generate gender as gender1，age as age1; //取别名。

C = FOREACH salaries GENERATE age..zip;

D = FOREACH salaries GENERATE age..;

E = FOREACH salaries GENERATE ..salary;

无结构的relations：

customer = LOAD 'data/customers';

F = FOREACH customer GENERATE $12..$23;

Pig的操作和命令是不区分大小写的，比如load，group。而函数是区分大小写的，比如MAX，COUNT。

set job.name 'J' //设置job名称，不修改会一直是J。

salariesbygender = GROUP salaries BY gender; //salariesbygender有两个字段，group和salariesbygender。

J = FOREACH salariesbygender GENERATE group，COUNT(salaries);

K = FOREACH salariesbygender GENERATE group，MAX(salaries.salary);

K1 = FOREACH salariesbygender GENERATE group，SUM(salaries.salary);

H = FILTER salaries BY gender == 'F' AND age >= 50;

employees = LOAD 'pigdemo.txt' AS (state:chararray，name:chararray);

no\_b = FILTER employees BY NOT name MATCHES '.\*b|B.\*';

Lab5.2:

A = load '/user/root/whitehouse/visits.txt' using TextLoader ();

B = GROUP A ALL;

A\_count = FOREACH B GENERATE 'rowcount'， COUNT(A); //添加'rowcount'常量。

visits = LOAD '/user/root/whitehouse/visits.txt' USING PigStorage('，');

potus = FILTER visits BY $19 MATCHES 'POTUS';

potus\_details = FOREACH potus GENERATE

(chararray) $0 AS lname:chararray，

(chararray) $1 AS fname:chararray，

(chararray) $6 AS arrival\_time:chararray，

(chararray) $19 AS visitee:chararray;

potus\_details\_ordered = ORDER potus\_details BY lname ASC;

为何group all 之后，group的类型为chararray，因为group的值为all，all是字符串，因此为chararray类型。

====================================================================================================

Unit 6: Advanced Pig Programming

1. ORDER BY:

byage = order salaries by age asc;

agesalary = ORDER salaries BY age ASC， salary ASC;

A = ORDER visitors BY lastname DESC;

B = FILTER A BY age >= 21;

The records in B are no longer guaranteed to be ordered by lastname descending.

2. CASE Operator

bonuses = FOREACH salaries GENERATE salary，(

CASE

WHEN salary >= 70000.00 THEN salary \* 0.10

WHEN salary < 70000.00 AND salary >= 30000.0

THEN salary \* 0.05

WHEN salary <= 30000.0 THEN 0.0

END) AS bonus;

3. Parameter Substitution 参数替换，在shell命令行执行脚本文件。

注意：input默认当前目录为: /user/root，默认是mapreduce方式执行，可省略-x mapreduce。

脚本：名为test.pig，内容如下:

A = LOAD '$input' AS (state:chararray,name:chararray);

SET job.name 'test';

STORE A INTO '$output' USING PigStorage('\t');

执行：

pig -p input=demos/pigdemo.txt -p output=/demos/result test

或 pig -x mapreduce -p input=demos/pigdemo.txt -p output=/demos/output2 test

或 pig -param\_file test.params test //test.params为参数文件，内容为：input=demos/pigdemo.txt output=/demos/output3

4. DISTINCT 操作：

employees = LOAD 'employees.txt';

unique\_emp = DISTINCT employees;

5. PARALLEL 并行：

PARALLEL 语句作用:指定reducer的个数.

A = LOAD 'data1';

B = LOAD 'data2';

C = JOIN A by $1， B by $3 PARALLEL 20;

D = ORDER C BY $0 PARALLEL 5;

指定parallel(reducer)有意义的语句： GROUP， ORDER BY， DISTINCT， JOIN， LIMIT and COGROUP.

指定parallel(reducer)无意义的语句： LOAD， FOREACH， FILTER， SAMPLE。

===============================================================

invoke:调用。

===============================================================

6.Flatten Operator，FLATTEN函数大写，前面加空格。

数据如下：（制表符分割）

Rich remote {(SD)，(CA)}

Ulf onsite {(CA)}

Tom remote {(OH)，(NY)}

Barry remote {(NV)，(NY)}

注意：如果数据以，分割，则不能使用以下语句创建结构。因为包里面也

有，号，会冲突。包中的元组中的字段为一个才有意义。为两个时，FLATTEN语句有问题。

employees = LOAD 'locations.txt' AS (

name:chararray，

location:chararray，

states:bag{t:tuple(state:chararray)});

flat\_employees = FOREACH employees GENERATE name，

location， FLATTEN(states) AS state;

结果：

(Rich，remote，SD)

(Rich，remote，CA)

(Ulf，onsite，CA)

(Tom，remote，OH)

(Tom，remote，NY)

注意：数据(1， (2， 3)) flat语句后，变为(1，2，3)；

Lab6.1: Splitting a Dataset

1.1) visits = LOAD 'visits.txt' USING PigStorage('，');

1.2) not\_null\_25 = FILTER visits BY ($25 IS NOT NULL); //is not null

1.3) comments = FOREACH not\_null\_25 GENERATE $25 AS comment;

1.4) comments\_sample = SAMPLE comments 0.001; //sample语句

4.2) SPLIT visits INTO congress IF($25 MATCHES

'.\* CONGRESS .\*')， not\_congress IF (NOT($25 MATCHES //split语句

'.\* CONGRESS .\*'));

7. Nested FOREACH

语法：

dailyA = LOAD 'NYSE\_daily\_prices\_A.csv' USING PigStorage('，') AS (exchange，symbol);

dailyA\_grp = GROUP dailyA BY exchange;

unique\_symbols = FOREACH dailyA\_grp {

symbols = dailyA.symbol;

unique\_symbol = DISTINCT symbols;

GENERATE group， COUNT(unique\_symbol);

};

dailyA = LOAD 'NYSE\_daily\_prices\_A.csv' USING

PigStorage('，') AS (exchange，symbol，date);

dailyA\_grp = GROUP dailyA BY symbol;

result = FOREACH dailyA\_grp {

sorted = ORDER dailyA BY date ASC;

GENERATE group， CONCAT(symbol，date);

};

8.Performing an Inner Join（内连接，只显示匹配的记录）

语法：

locations = LOAD 'pigdemo.txt' AS (state:chararray，firstname:chararray);

departments = LOAD 'joindemo.txt' AS (firstname:chararray，dept:chararray);

innerjoin = JOIN locations BY firstname， departments BY firstname;

结构：

innerjoin:{

locations::state: chararray，

locations::firstname: chararray，

departments::firstname: chararray，

departments::dept: chararray

}

9. Performing an Outer Join （外连接，left|right|full outer）

左连接：左表全部记录，及匹配的记录。全连接：左右表所有数据。

outerjoin = JOIN locations BY firstname FULL OUTER，departments BY firstname;

leftjoin = JOIN locations BY firstname LEFT OUTER，departments BY firstname;

# 服务器介绍

塔式的个头类似于PC机，属于入门级及工作组级服务器应用，成本低，适合中小企业。如果需要采用多台塔式才能满足工作需求时，塔时不便管理。

机架式能够将多台机架式服务器放入一个机柜，占用的空间小，便于统一管理。通常有1U、2U、3U、4U、5U、7U几种标准。不便之处在于扩充性和散热性比较受限制。适合远程存储和网络服务等。机架服务器的宽度为19英寸，高度以U为单位（1U=1.75英寸=44.45毫米），通常有1U，2U，3U，4U，5U，7U几种标准的服务器。  
 刀片式主要结构是一大型主体机箱，里面可以插入需多“刀片”，每块刀片就是独立的服务器，可以独立运行自己的系统，服务于不同的用户，互不干涉。它比机架式服务器更省空间，当然散热问题也更明显。但其满足下一代服务器新要求“节约空间、便于集中管理、易于扩展和提供不间断的服务”。具有良好的市场情景。

1. 塔式服务器



1. 机架式服务器



1. 刀片服务器



# 18. Hadoop HA搭建，及平台部署

主从机构

主：jobtracker

从：tasktracker

四个阶段：

1. Split
2. Mapper：key-value（对象）
3. Shuffle
   1. 分区（partition，HashPartition：根据key的hashcode值 和 Reduce的数量 模运算），可以自定义分区，运算速度要快。一定要解决数据倾斜和reduce的负载均衡。
   2. 排序：默认按照字典排序。WriterCompartor（比较）
   3. 合并：减少当前mapper输出数据，根据key相同（比较），把 value 进行合并。
   4. 分组（key相同（比较），value组成一个集合）（merge）
4. Reduce
   1. 输入数据：key + 迭代器

Hadoop2.5 HA搭建

四台机器：hadoop1，hadoop2，hadoop3，hadoop4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | NN | DN | ZK | ZKFC | JN | RM | NM（任务管理） |
| Hadoop1 | Y |  | Y | Y |  |  |  |
| Hadoop2 | Y | Y | Y | Y | Y | Y | Y |
| Hadoop3 |  | Y | Y |  | Y |  | Y |
| Hadoop4 |  | Y |  |  | Y |  | Y |

1. core-site.xml

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://bjsxt</value>

</property>

<property>

<name>ha.zookeeper.quorum</name>

<value>hadoop1:2181,hadoop2:2181,hadoop3:2181</value>

</property>

<property>

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

<value>/opt/hadoop</value>

</property>

</configuration>

1. hdfs-site.xml

<configuration>

<property>

<name>dfs.nameservices</name>

<value>bjsxt</value>

</property>

<property>

<name>dfs.ha.namenodes.bjsxt</name>

<value>nn1,nn2</value>

</property>

<property>

<name>dfs.namenode.rpc-address.bjsxt.nn1</name>

<value>hadoop1:8020</value>

</property>

<property>

<name>dfs.namenode.rpc-address.bjsxt.nn2</name>

<value>hadoop2:8020</value>

</property>

<property>

<name>dfs.namenode.http-address.bjsxt.nn1</name>

<value>hadoop1:50070</value>

</property>

<property>

<name>dfs.namenode.http-address.bjsxt.nn2</name>

<value>hadoop2:50070</value>

</property>

<property>

<name>dfs.namenode.shared.edits.dir</name>

<value>qjournal://hadoop2:8485;hadoop3:8485;hadoop4:8485/bjsxt</value

>

</property>

<property>

<name>dfs.client.failover.proxy.provider.bjsxt</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverPr

oxyProvider</value>

</property>

<property>

<name>dfs.ha.fencing.methods</name>

<value>sshfence</value>

</property>

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name>

<value>/root/.ssh/id\_dsa</value>

</property>

<property>

<name>dfs.journalnode.edits.dir</name>

<value>/opt/hadoop/data</value>

</property>

<property>

<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

</configuration>

1. 准备zookeeper3.4.6
   1. 三台zookeeper：hadoop1，hadoop2，hadoop3
   2. 编辑zoo.cfg配置文件
      1. 修改dataDir=/opt/zookeeper
      2. server.1=hadoop1:2888:3888

server.2=hadoop2:2888:3888

server.3=hadoop3:2888:3888

* 1. 在dataDir目录中创建一个myid的文件，文件内容为1，2，3

1. 配置hadoop中的slaves，datanode的主机名，编辑hadoop-env.sh，添加export JAVA\_HOME=/usr/local/jdk1.7.0\_79。
2. 启动三个zookeeper：./zkServer.sh start，之后在其中一个上检验，执行zkCli.sh 看是否报错。
3. 启动三个JournalNode：./hadoop-daemon.sh start journalnode
4. 在其中一个namenode上格式化：hdfs namenode -format
5. 把刚刚格式化之后的元数据拷贝到另外一个namenode上
   1. 启动刚刚格式化的namenode，hadoop-daemon.sh start namenode
   2. 在没有格式化的namenode上执行：*hdfs namenode –bootstrapStandby*
   3. 启动第二个namenode，hadoop-daemon.sh start namenode
6. 在其中一个namenode上初始化zkfc：hdfs zkfc –formatZK
7. 停止上面节点：stop-dfs.sh
8. 全面启动：start-dfs.sh

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | NN | DN | ZK | ZKFC | JN | RM | NM（任务管理） | secondaryNode | jobhistory |
| Node70 | Y，MySQL,R服务，tomcat |  | Y | Y |  |  |  |  |  |
| Node71 | Y | Y | Y | Y | Y | Y | Y | Y |  |
| Node72 |  | Y | Y |  | Y |  | Y |  | Y |

注意：如果是vmware虚拟出来的机器，则先安装一台：  
1）配置机器名、hosts，静态IP，删除/etc/udev/rules.d/70-persistent-net.rules。

1. 上传安装jdk，hadoop包，zookeeper包，sqoop包，mysql驱动包，上传包caic-forecast-gdt.jar到 /usr/local/hadoop-2.2.0/share/hadoop/mapreduce，上传servlet-api-remaining.jar，mahout-\*.jar包共5个到/usr/local/hadoop-2.2.0/share/hadoop/mapreduce/lib下，windows提交还需上传另外三个到/opt下面。
2. 克隆，修改对应机器名，静态IP，重启网络，修改myid。

如果是物理机或网络互通则可以直接安装：

1. 机器名、hosts，静态IP，免密码，jdk，hadoop，zookeeper，sqoop，配置hadoop环境变量。

机器名：

/etc/sysconfig/network:

NETWORKING=yes

HOSTNAME=node70

hosts文件：

192.168.222.70 node70 node70.example.com

192.168.222.71 node71 node71.example.com

192.168.222.72 node72 node72.example.com

静态IP：

rm -f /etc/udev/rules.d/70-persistent-net.rules

Vi /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0

TYPE=Ethernet

ONBOOT=yes

NM\_CONTROLLED=yes

BOOTPROTO=static

IPADDR=192.168.222.70

NETMASK=255.255.255.0

GATEWAY=192.168.222.2

DNS1=192.168.222.2

DNS2=8.8.8.8

重启三个电脑，使生成新的网卡（使用vmware克隆）

免密码：

生成秘钥：ssh-keygen ,三个回车。

添加公钥到其他机器：ssh-copy-id node71

添加自己公钥：ssh-copy-id node70 或 cat id\_rsa.pub >> authorized\_keys

之后，都相互ssh一下，相互添加到konws文件。

export JAVA\_HOME=/usr/local/jdk1.7.0\_79

export CLASSPATH=.:$JAVA\_HOME/lib/tools.jar:$JAVA\_HOME/lib/dt.jar

export R\_HOME=/usr/local/R-3.2.0

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

export HADOOP\_HOME=/usr/local/hadoop-2.2.0

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_HOME\_LIB=$HADOOP\_HOME/share/hadoop/mapreduce

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib"

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export SQOOP\_HOME=/usr/local/sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha

export ZOO\_HOME=/usr/local/zookeeper-3.4.6

export PATH=$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin:$PATH:$SQOOP\_HOME/bin:$ZOO\_HOME/bin

2、集群配置本地时间服务器，ntpd服务。

3、zkServer.sh start 启动之后 输入命令：jps 查看控制台是否有QuorumPeerMain进程存在。

成功后启动另外的机器，然后zkCli.sh进入，看是否正常。

4、hadoop2.2 start-dfs.sh要在namenode上启动，start-yarn.sh必须要在resourcemanager节点启动，因此需要写启动脚本start\_cluster.sh，hadoop2.6则可以全在一个节点启动，参看脚本start\_cluster.sh 和stop\_cluster.sh。

Zookeeper，zkfc，namenode具体步骤参考上面4个节点的步骤，及node51-node57的配置文件或node70-node71配置文件。

5、caic-forecast-gdt.jar，mahout-\*.jar等包。

6、tomcat在windows的话，opt下面有三个包！

7、

添加环境变量：

export HADOOP\_HOME=/usr/local/hadoop-2.2.0

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_HOME\_LIB=$HADOOP\_HOME/share/hadoop/mapreduce

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib"

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export SQOOP\_HOME=/usr/local/sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha

export ZOO\_HOME=/usr/local/zookeeper-3.4.6

export PATH=$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin:$PATH:$SQOOP\_HOME/bin:$ZOO\_HOME/bin

Linux提交任务（tomcat,项目在linux上）：

1、增加环境变量：

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

2、配置Hadoop集群时，在/usr/local/hadoop-2.2.0/etc/hadoop下面，yarn-site.xml，添加

$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*,

$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*

3、/home/project/caic-forecast-gdt/WebRoot/WEB-INF/lib/spark-assembly-1.2.1-hadoop2.3.0.jar包替换：

用F:\Workspaces\caic-forecast-gdt\resource下的core-default.xml,hdfs-default.xml,yarn-default.xml文件替换spark-assembly-1.2.1-hadoop2.3.0.jar中的原有.xml文件。

替换之后，spark-assembly.jar中不包含yarn-site.xml文件。

Windows 提交任务（tomcat,项目在windows上，伪分布式可以，HA（spark算法不可以，mr可以）。）：

1、增加环境变量：

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

2、配置Hadoop集群时，在/usr/local/hadoop-2.2.0/etc/hadoop下面，yarn-site.xml，添加

$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*,

$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*

3、

hadoop-yarn-common-2.2.0.jar 只添加HADOOP\_MAP\_RED

$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*,

$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*,

yarn.resourcemanager.hostname默认为0.0.0.0，不修改。

4、/opt下上传jar包，spark-assembly-1.2.1-hadoop2.3.0.jar，Spark\_algorithm1.0.jar和servlet-api.jar。

1）其中spark-assembly.jar包，使用Linux方式替换后的jar包，并复制yarn-default.xml,修改为yarn-site.xml，

添加yarn.application.classpath的值为：

$HADOOP\_CONF\_DIR,$HADOOP\_COMMON\_HOME/share/hadoop/common/\*,

$HADOOP\_COMMON\_HOME/share/hadoop/common/lib/\*,

$HADOOP\_HDFS\_HOME/share/hadoop/hdfs/\*,$HADOOP\_HDFS\_HOME/share/hadoop/hdfs/lib/\*,

$HADOOP\_YARN\_HOME/share/hadoop/yarn/\*,$HADOOP\_YARN\_HOME/share/hadoop/yarn/lib/\*,

$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*,$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*

2）修改yarn-site.xml，yarn.resourcemanager.hostname为node71。

5、修改数据库，spark.jar值为：

/opt/spark-assembly-1.2.1-hadoop2.3.0.jar:/opt/Spark\_algorithm1.0.jar:/opt/servlet-api.jar

11、sqoop配置：

1）mysql驱动包，拷贝到sqoop的lib中。

2）数据源配置，sqoop在虚拟机上，导出到本地windows中mysql，数据源配置IP为：192.168.222.1，而不是当前无线IP。

# 19. Centos6.7 配置Hadoop2.6.x

1. 通过xmanager的Xftp上传hadoop-2.6.0.tar.gz文件到/opt目录

2. 解压缩hadoop-2.6.0.tar.gz 文件

tar -zxf hadoop-2.6.0.tar.gz

解压后即可，看到/opt/hadoop-2.6.0.tar.gz文件夹

3.配置ssh 无密码登录

输入命令：ssh-keygen -t rsa

接着按三次Enter键，如下

[root@centos67 opt]# ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Created directory '/root/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

22:ec:f0:b6:2b:dc:54:d6:4f:ae:a0:a8:e6:3d:55:84 root@centos67

The key's randomart image is:

+--[ RSA 2048]----+

| . |

| E . |

| .. |

| . o.. . |

| . ooo S+ |

| +.o.. o |

| . +=. . . |

| .=oo. . |

|+o o+. |

+-----------------+

修改ip与hostname映射,修改/etc/hosts(删除所有，只添加下面一行)

192.168.0.128 centos67

输入命令：ssh-copy-id -i /root/.ssh/id\_rsa.pub centos67,依次输入yes,123456（root用户的密码）

，如下：

[root@centos67 opt]# ssh-copy-id -i /root/.ssh/id\_rsa.pub centos67

The authenticity of host 'centos67 (192.168.0.128)' can't be established.

RSA key fingerprint is 09:7a:e4:ad:28:ce:ac:b6:0f:ea:99:82:fa:62:25:96.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'centos67,192.168.0.128' (RSA) to the list of known hosts.

root@centos67's password:

Now try logging into the machine, with "ssh 'centos67'", and check in:

.ssh/authorized\_keys

to make sure we haven't added extra keys that you weren't expecting

验证：ssh centos67 ，直接登录即可

[root@centos67 opt]# ssh centos67

Last login: Tue Nov 3 18:39:41 2015 from 192.168.0.1

4. 配置Hadoop

进入目录：

cd /opt/hadoop-2.6.0/etc/hadoop/

依次修改下面的文件：

4.1 core-site.xml

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs:// tipdmInner:8020</value>

</property>

<property>

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

<value>/var/log/hadoop/tmp</value>

</property>

</configuration>

4.2 hadoop-env.sh

export JAVA\_HOME=/opt/jdk1.7.0\_55

4.3 hdfs-site.xml

<property>

<name>dfs.namenode.name.dir</name>

<value>file:///data/hadoop/hdfs/name</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:///data/hadoop/hdfs/data</value>

</property>

<property>

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

<value>tipdmInner:50090</value>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

4.4 mapred-site.xml

<property>

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

<value>yarn</value>

</property>

<!-- jobhistory properties -->

<property>

<name>mapreduce.jobhistory.address</name>

<value>tipdmInner:10020</value>

</property>

<property>

<name>mapreduce.jobtracker.http.address</name>

<value>tipdmInner:50030</value>

</property>

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>tipdmInner:19888</value>

</property>

4.5 yarn-site.xml

<property>

<name>yarn.resourcemanager.hostname</name>

<value>tipdmInner</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8032</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8030</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8088</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8090</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8031</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8033</value>

</property>

<property>

<name>yarn.nodemanager.local-dirs</name>

<value>/data/hadoop/yarn/local</value>

</property>

<property>

<name>yarn.log-aggregation-enable</name>

<value>true</value>

</property>

<property>

<name>yarn.nodemanager.remote-app-log-dir</name>

<value>/data/tmp/logs</value>

</property>

<property>

<name>yarn.log.server.url</name>

<value>http://tipdmInner:19888/jobhistory/logs/</value>

<description>URL for job history server</description>

</property>

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

</property>

<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>

4.6 yarn-env.sh

export JAVA\_HOME=/opt/jdk1.7.0\_55

4.7 slaves

hivejiaoCai

5. 格式化NameNode

进入目录

cd /opt/hadoop-2.6.0/bin

执行格式化

./hdfs namenode -format

6.启动集群

进入目录

cd /opt/hadoop-2.6.0/sbin

执行启动：

./start-dfs.sh

./start-yarn.sh

./mr-jobhistory-daemon.sh start historyserver

使用jps，查看进程

[root@centos67 sbin]# jps

3672 NodeManager

3301 DataNode

3038 NameNode

4000 JobHistoryServer

4058 Jps

3589 ResourceManager

3408 SecondaryNameNode

浏览器查看：

首先，关闭防火墙：

service iptables stop

chkconfig iptables off

http://192.168.0.128:50070

http://192.168.0.128:8088

Clone of centos6.6.vmx

ps -ef | grep java | grep apache-tomcat-6.0.35

echo "kill tomcat"

PID=`ps au | grep vmware|grep bin/vmware |awk '{print $2}'`

for id in $PID

do

kill -9 $id

done

echo "kill myeclipse"

kill -9 `ps |grep vmware |awk '{print $1}'`

ID2=`jps|grep org.eclipse |awk '{print $1}'`

for id2 in $ID2

do

done

ps -ef | grep ambari

service ambari-server status

service ambari-server start

192.168.222.10:8080 admin/admin登录，hdp集群都是借此启动/关闭。

进去节点不能点击，好像IP只能是127.0.0.1才能。

# 20. Centos6.7虚拟机配置Hive1.2.1

1. 通过xmanager的Xftp上传apache-hive-1.2.1-bin.tar.gz文件到/opt目录

2. 解压缩apache-hive-1.2.1-bin.tar.gz 文件

tar -zxf apache-hive-1.2.1-bin.tar.gz

解压后即可，看到/opt/apache-hive-1.2.1-bin文件夹

3. 配置hive

进入目录：

cd /opt/apache-hive-1.2.1-bin/conf

3.1

3.1.1 使用derby，存储hive元信息，直接使用hive-default.xml.template，无需修改。

3.1.2 使用Mysql数据库，存储hive元信息。

新建hive-site.xml文件，添加内容如：

hive-site.xml，jdbc:mysql://centos67:3306/hive中的hive数据库需要提前创建。

3.2 配置hive-env.sh（从hive-env.sh.template拷贝）

内容如：

HADOOP\_HOME=/opt/hadoop-2.6.0

编辑profile,添加HADOOP\_HOME路径。

export HADOOP\_HOME=/opt/hadoop-2.6.0

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

3.3 拷贝mysql连接jar包到hive的lib目录

1）通过Xftp上传mysql-connector-java-5.1.25-bin.jar到/opt/apache-hive-1.2.1-bin/lib目录

3.4 拷贝jline.jar 到目录:

mv $HADOOP\_HOME/share/hadoop/yarn/lib/jline-0.9.94.jar $HADOOP\_HOME/share/hadoop/yarn/lib/jline-0.9.94.jar.bak

cp $HIVE\_HOME/lib/jline-2.12.jar $HADOOP\_HOME/share/hadoop/yarn/lib/

4. 运行 hive

进入目录

cd /opt/apache-hive-1.2.1-bin/bin

运行

./hive

WARN conf.HiveConf: HiveConf of name hive.metastore.local does not exist

1

这是由于在0.10 0.11或者之后的HIVE版本 hive.metastore.local 属性不再使用。将该参数从hive-site.xml删除即可。

# 21. Spark运行命令样例

1、local单机部署模式：

结果xshell可见：

./bin/spark-submit --class org.apache.spark.examples.SparkPi --master local[1] ./lib/spark-examples-1.3.1-hadoop2.4.0.jar 100

2、standalone集群部署模式：

driver在哪个机器上运行，执行进度可在(node1,2,3:4040)中看到，客户端模式执行，driver在本地node1上,没有提交到集群上，

node1:8080可以看到上面没有对应的driver, 先执行driver,在执行application，两者都有对应的id,提交一个程序到

集群上后，一般driver,application不在同一个节点上执行，driver提交到哪个节点，结果就在哪个节点。每个节点上

都记录了他一共被提交了多少个driver，状态如何等信息。

1)client运行模式：

driver在本地执行，结果在本地显示。

结果xshell可见：

./bin/spark-submit --class test6.SparkPi2 --master spark://node1:7077 --executor-memory 2G --total-executor-cores 4 /root/t1.jar 100

2)cluster运行模式：

结果node1:8080里面可见！

./bin/spark-submit --class test6.SparkPi2 --master spark://node1:7077 --deploy-mode cluster --supervise --executor-memory 1G --total-executor-cores 1 /root/t1.jar 100

3、Yarn集群部署模式：需要启动start-yarn.sh

1)client运行模式：

结果xShell里面可见！

./bin/spark-submit --class test6.SparkPi2 --deploy-mode client --executor-memory 1G --num-executors 1 /root/t1.jar 100

2)cluster运行模式：

结果node1:8088里面可见！在resourcemanager界面可以看到程序运行。

./bin/spark-submit --class test6.SparkPi2 --master yarn-cluster --executor-memory 1G --num-executors 1 /root/t1.jar 100

# 22. Eclipse配置Hadoop开发环境

方式1：

hadoop jar方式运行；

1. 新建java 工程，添加build path ，添加Hadoop lib的主要包；

2. 开发MR程序，并打包到Hadoop client机器；

3. 使用hadoop jar的方式运行MR程序；

方式2：

eclipse hadoop plugin：

1. 使用hadoop-eclipse-Plugin.jar 在eclipse中添加Map/Reduce视图，

并指明Hadoop的安装目录（和eclipse同样安装目录的hadoop安装包解压目录即可）

2. 添加新的Hadoop Location ,其中：

a. Map/Reduce(V2) master 对应的是mapred-site.xml 中的

<property>

<name>mapreduce.jobtracker.http.address</name>

<value>node100:50030</value>

</property>

b. dfs master 对应 core-site.xml中的

<property>

<name>fs.defaultFS</name>

<value>hdfs://node100:8020</value>

</property>

c. Advanced parameters 最后一项：

yarn.resourcemanager.address 修改为： node100:8032

3. 新建java工程，并拷贝集群中$HADOOP\_HOME/etc/hadoop 目录下面的

core-site.xml

hdfs-site.xml

mapred-site.xml

yarn-site.xml

log4j.properties

文件到src目录下；

4. 新建Configuration时，需指明下面的配置：

conf.set("mapreduce.app-submission.cross-platform", "true");

conf.set("mapreduce.job.jar",JarUtil.jar(JarUtil.class));

其中，JarUtil参考本目录JarUtil.java文件；

5. 开发MR程序，直接运行Run on Hadoop即可（不需要打包上传，在JarUtil中已经做了这个工作）。

方式3

maven web

1. 配置pom文件，添加hadoop的相关依赖包，如hadoop-maven.pom.xml所示；

2. 配置Configuration，添加相关参数，如下：

conf.setBoolean("mapreduce.app-submission.cross-platform", "true");// 配置使用跨平台提交任务

conf.set("fs.defaultFS", "hdfs://node100:8020");// 指定namenode

conf.set("mapreduce.framework.name","yarn"); // 指定使用yarn框架

conf.set("yarn.resourcemanager.address","node100:8032"); // 指定resourcemanager

conf.set("yarn.resourcemanager.scheduler.address", "node100:8030");// 指定资源分配器

conf.set("mapreduce.jobhistory.address","node100:10020");

conf.set("mapreduce.job.jar",JarUtil.jar(JarUtil.class));// 如果是web工程需要考虑是否需要每次都打包，

// 还是设置在每次tomcat启动时打包一次即可 ,

// 可以设置初始化函数，初始化Configuration时打包即可；Configuration只初始化一次

3. 拷贝log4j.properties 文件到src/main/resources目录

4. 根据固定应用（web？）设置JarUtils打包的路径

在JarUtils中获取输入的方式：

1>

public static String jar(Class<?> cls){// 验证ok

String outputJar =cls.getName()+".jar";

String input = cls.getClassLoader().getResource("").getFile();

input= input.substring(0,input.length()-1);

input = input.substring(0,input.lastIndexOf("/")+1);

input =input +"classes/";

jar(input,outputJar);

return outputJar;

}

2>

public static String jar(Class<?> cls){ // 未验证

String outputJar =cls.getName()+".jar";

String intput=ContextLoader.getCurrentWebApplicationContext().getServletContext().getRealPath("/");

jar(input,outputJar);

return outputJar;

}

3>

public static String jar(Class<?> cls){// 验证ok

String outputJar =cls.getClass().getName()+".jar";

jar(cls.getClassLoader().getResource("").getFile(),outputJar);

return outputJar;

}

1. 直接部署web程序，运行MR即可。

# 23 .HortonWorks开发者培训

1. 准备：

1） 拷贝虚拟机和相关文档：

2）如何使用虚拟机；

3） 准备IDE、xmanager

第一次培训：2016/4/5

用户名/密码：root/hadoop , train/hadoop

Ambari Server : service ambari-server start|stop|status ; (root 用户)

Ambari Agent : service ambari-agent start|stop|status;

访问浏览器： http://192.168.0.128:8080/ admin/admin

以pdf为准：

首次课程，简单介绍；

后面每次：总结前周内容，简单介绍下周内容；

主要以自学为主；

第二次培训：2016/4/8

1. 内部资料

1.1 问题？

1）2.6.0？ 2.2.0

查看hadoop版本： 进入/usr/lib ，然后进入hadoop对应目录，即可查看对应jar包版本；

2)

1.2 环境配置：

1) plugin : eclipse\dropins

2) 小象/MapReduce

添加小象：

a. DFS location：

/etc/hadoop/conf/core-site.xml :

<name>fs.defaultFS</name>

<value>hdfs://sandbox.hortonworks.com:8020</value>

b. Map/Reduce

/etc/hadoop/conf/yarn-site.xml :

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

<value>sandbox.hortonworks.com:8050</value>

3）HDP关闭不需要使用的服务

包括： Hive、 HBase 、Zookeeper 、WebHCat、 Oozie

4） 相关资源：

/home/train/labs/Solutions： 完整实验；

/home/train/labs/Lab\* : 学生实验；

2. 概念：

HDFS

MapReduce

2.1 HDFS api实验

1)源码：/home/train/Lab/Solution

2) 添加 jdk；

export JAVA\_HOME=/usr/share/jdk1.6.0\_14

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

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

3. 一个WordCount 实例

1） 把src下面的源码（liux）下载到ide；

2） 编译程序，打jar包；

3） 拷贝jar包到linux,运行 hadoop jar test.jar demo.WordCount <input> <output>

4） java.lang.UnsupportedClassVersionError: wordcount/WordCountJob : Unsupported major.minor version 51.0

解决方法（有问题，不使用，直接使用6）的解决方法）：

a. 进入目录 /etc/hadoop/conf ；

b. 修改 hadoop-env.sh ， yarn-env.sh 文件中的JAVA\_HOME 为自己配置的JDK；

5）org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory hdfs://sandbox.hortonworks.com:8020/user/root/wc\_00 already exists

解决方法：

a. 删除对应目录； hadoop fs -rm -r /user/root/wc\_00

b. 使用其他输出目录

c. 在程序中 编写对应的程序，进行对输出目录的删除；

6) org.apache.hadoop.util.Shell$ExitCodeException

问题查看： 去浏览器查看8088端口，查看具体日志，错误其实还是4） UnsupportedClassVersionError

解决方法：

解决方法一：（未试验过）

1. 4）中的JDK相关配置还原；/etc/profile；同时，还有/etc/hadoop/conf里面的hadoop-env.sh ,yarn-env.sh中的JAVA\_HOME还原；

2. 修改IDE中的jdk为1.6版本；

3. 再次打包运行，即可

解决方法二：（试验过，可以）

1. 4）中的JDK相关配置还原；/etc/profile;同时，还有/etc/hadoop/conf里面的hadoop-env.sh ,yarn-env.sh中的JAVA\_HOME还原

2. 查看虚拟机中的jdk；

发现有两个jdk： 1. openjdk1.6.0\_24 ; 2. /usr/jdk64/jdk1.6.0\_31

3. 使用jdk1.7 去替换 /usr/jdk64/jdk1.6.0\_31的所有内容；

第三次培训 2016/4/15

今次课程检查作业内容：

1. 完成WordCount程序，并打包；

2. WordCount 使用 hadoop jar运行，并查看结构（50070,8088）；

3. 视频意见？先理论，再实践？

目标：

Use combiners and in-map aggregation

1. mapreduce代码复习；

2. map phase、reduce phase

3. YARN

4. Lab2.3 Inversed Index 视频； （作业）

运行程序时会启动ApplicationMaster， jps

1) Counter里面，为什么map读取的记录和输出的记录不一样？14，76怎么得到的 （？？？）

5. Lab3.1 Using a Combiner ，log日志，视频；

1） useCombiner 设置为false，再次观察实验；（？？？）

2）Counter， Job自带的

6. Lab 3.2: Computing an Average（作业）

第四次培训 2016/4/22

作业： counter记录？ log日志？ Lab3.2 ？

目标：

Write a custom partitioner to avoid data skew on reducers

Perform a secondary sort

1. Partitioner简介， 工作原理，默认Partitioner

2. 自定义Partitioner

实现步骤

3. TotalOrderPartioner 应用场景；

实现步骤

视频 Lab4.2 全局排序，排前排后对比

4. Secondary Sort

1）自定义key，value格式；

2）custom Grouping Comparator 组比较器

3）custom Partitioner 分区器

视频 Lab4.3

作业：DividendChange不加这个toString方法有什么问题？

番外篇： Eclipse远程调用Hadoop集群

1）JarUtil.java ，eclipse配置Hadoop环境；

2）conf.set(\*); 程序中需要设置的

第五次培训：2016/5/6

作业：DividendChange 自定义键值类型 ，复写toString方法？

目标：

Write a custom MapReduce input and output format

Optimize a MapReduce job

1. InputFormat

a. key/value如何产生

b. 内置类型

c. splits : 逻辑分片，blocks ：物理分片

1)单个记录在物理逻辑分割，可以在逻辑上合并；

2) 两个物理分片合并为一个逻辑分片，减小map数；

d. 自定义RecordReader

e. CombineFileInputFormat （多reducer）

f. lab5.1 看过就好

2. OutputFormat

a. key/value如何写入

b. 内置类型

c. 序列文件

d. MultipleOutputs: key/value类型是否可变？

e. lab5.2 有一定实用价值

3. lab5.3 移动平均

视频

4. lab6.2

第六次培训： 2016/5/13

目标：

Advanced MapReduce Features

MR Unit Testing

HBase

1. 资源本地化：

两种方式：1） addCache , 2) 设置变量

2. join三种方式：

1） map端；

2） reduce端；

3. bloom filter

4. Unit testing ：

填空、选择

5. HBase

1） 架构

2）数据存储；

3）列簇、Version

4）HBase编程：

a. hdfs->hbase

b. hbase-hdfs

6. 视频：

Lab9.1

Lab9.2

=====================================

第七次培训： 2016/5/20

hive:

运行：

1）客户端：hive

2）hive -e "一条语句" 或 hive -f 脚本。

3)远程服务:nohup hive --service hiveserver2 & (提供java-JDBC远程调用)

命令：

创建表：

1）create table names (

id int,

name string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

2）create external table external\_names (

id int,

name string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LOCATION '/user/root/hivedemo';

create table part\_names (id int, name string) partitioned by (state string) row format delimited fields terminated by '\t';

4）create table test as select \* from .. where .. ;

加载数据：

load data local inpath '/user/student\_all.txt' overwrite into table student\_hdfs;

load data inpath '/user/test\_part.txt' overwrite into table test\_part partition (month='2010-01');

存储数据：

=================================

pig:

运行：

1）local方式：pig -x local

2）mapreduce方式：pig 或（pig -x mapreduce）

3）pig 脚本

数据加载：

数据存储：

数据转换：

命令替换：test.pig

A = LOAD '$input' AS (state:chararray,name:chararray);

SET job.name 'test';

STORE A INTO '$output' USING PigStorage('\t');

执行：pig -x mapreduce -p input=demos/pigdemo.txt -p output=/demos/result test.pig

2. 课程：（一次一周）

Describe Hadoop 2.X and the Hadoop Distributed File System

Describe the YARN framework

Develop and run a Java MapReduce application on YARN

Use combiners and in-map aggregation

Write a custom partitioner to avoid data skew on reducers

Perform a secondary sort

Recognize use cases for built-in input and output formats

Write a custom MapReduce input and output format

Optimize a MapReduce job

Configure MapReduce to optimize mappers and reducers

Develop a custom RawComparator class

Distribute files as LocalResources

Describe and perform join techniques in Hadoop

Perform unit tests using the UnitMR API

Describe the basic architecture of HBase （最后两次是否合并为一次课程）

Write an HBase MapReduce application

List use cases for Pig and Hive

Write a simple Pig script to explore and transform big data

Write a Pig UDF (User-Defined Function) in Java

Write a Hive UDF in Java

Use JobControl class to create a MapReduce workflow

Use Oozie to define and schedule workflows

3. 实验：（参考课程对应实验）

Configuring a Hadoop Development Environment

Putting data into HDFS using Java

Write a distributed grep MapReduce application

Write an inverted index MapReduce application

Configure and use a combiner

Writing custom combiners and partitioners

Globally sort output using the TotalOrderPartitioner

Writing a MapReduce job to sort data using a composite key

Writing a custom InputFormat class

Writing a custom OutputFormat class

Compute a simple moving average of stock price data

Use data compression

Define a RawComparator

Perform a map-side join

Using a Bloom filter

Unit testing a MapReduce job

Importing data into HBase

Writing an HBase MapReduce job

Writing User-Defined Pig and Hive functions

Defining an Oozie workflow

# 24. HortonWorks考试

考试相关介绍：

https://hortonworks.com/training/certification/hdpcdjava-certification/#section\_1

0. 考试环境：

1） 基于HDP2.2 ；

2） 预安装Eclipse环境，可以直接编写Java代码；

3） 需要注册webcam（https://www.examslocal.com//） 和较好的网络环境；

4） 考试时间：2小时；

考试网站：https://www.examslocal.com/ScheduleExam/Home/CompatibilityCheck#

广州市萝岗区科学城科珠路232号益民科技园区2栋405

1. 相关内容：

会配置以及开发下面的技术：

1）combiners

2）partitioners；

3）custom keys；

4）custom sorting；

joining of datasets（主要看我介绍的两种，第二种可以忽略）；

详细描述：

Task Resource(s)

1）Write a Hadoop MapReduce application in Java http://hadoop.apache.org/docs/r2.6.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

2）Run a Java MapReduce application on a Hadoop cluster http://hadoop.apache.org/docs/r2.6.0/hadoop-yarn/hadoop-yarn-site/YarnCommands.html#jar

3）Write and configure a Combiner for a MapReduce job https://developer.yahoo.com/hadoop/tutorial/module4.html#functionality

4）Write and configure a custom Partitioner for a MapReduce job https://developer.yahoo.com/hadoop/tutorial/module5.html#partitioning

5）Sort the output of a MapReduce job http://hadoop.apache.org/docs/r2.6.0/api/org/apache/hadoop/mapreduce/Job.html #setGroupingComparatorClass(java.lang.Class)

6）Write and configure a custom key class for a MapReduce job http://hadoop.apache.org/docs/r2.6.0/api/org/apache/hadoop/io/WritableComparable.html

7）Configure the input and output formats of a MapReduce job http://hadoop.apache.org/docs/r2.6.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Job\_Input

8）Perform a join of two or more datasets

Many online examples and resources

9）Perform a map-side join of two datasets Many online examples and resources

eclipse查看mapreduce结果，乱码解决：

使用weibo\_utf8.txt,在eclipse里，点击"Windows" -> "Preference.." ，选择"General" -> "Workspace"，修改为utf-8。

max.split(100M)

min.split(10M)

block(64M)

max(min.split,min(max.split,block))

FileInputFormat分片源码：

List<InputSplit> splits = new ArrayList<InputSplit>();

long blockSize = file.getBlockSize();

long splitSize = computeSplitSize(blockSize, minSize, maxSize);==>>max(min.split,min(max.split,block))

long bytesRemaining = length; //length为文件大小，SPLIT\_SLOP=1.1

while (((double) bytesRemaining)/splitSize > SPLIT\_SLOP) {

int blkIndex = getBlockIndex(blkLocations, length-bytesRemaining);

splits.add(makeSplit(path, length-bytesRemaining, splitSize,

blkLocations[blkIndex].getHosts())); //添加FileSplit文件，一个FileSplit对应一个map。

bytesRemaining -= splitSize;

参考：http://ronxin999.blog.163.com/blog/static/42217920201279112163/

溢写文件大小设置，观察combiner，即有多个溢出文件时，combiner记录的是否包含合并？

多个map，没有溢写时，combiner个数。

只调整分片大小：可以控制分片个数，产生多个maptask，每个map有一个溢出文件，combiner输出有增加，因为多个map结果之间，无法合并，就多产生了combiner out 个数。

只调整溢出内存：只产生一个map,但可产生多个溢出文件，combiner输出有增加，Reduce input不变，因为只有一个map。相同map可以合并。

conf.set("io.sort.mb","1");

conf.setDouble("io.sort.spill.percent",0.5 );

conf.setLong("dfs.block.size",5\*1024\*1024);

conf.setLong("mapreduce.input.fileinputformat.split.maxsize",40\*1024 );

String minsize = conf.get("mapreduce.input.fileinputformat.split.minsize");

Map input records=1065

Map output records=28333

Map output bytes=829709

Map output materialized bytes=743427

Input split bytes=108

Combine input records=28333

Combine output records=23125

Reduce input groups=23119

Reduce shuffle bytes=743427

Reduce input records=23125

Reduce output records=23119

Spilled Records=69375

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

mapreduce二次排序，及mapreduce的常规实现。

缓存，config.set();

翻墙参考：http://blog.my-eclipse.cn/host-google.html

/home/Workspaces/caic-forecast-gdt/WebRoot/rserve/model

/home/Workspaces/caic-forecast-gdt/WebRoot/rserve/csv

du -h --max-depth=1 /root/.m2/repository/

报名信息：https://www.examslocal.com/ScheduleExam/Payment/DisplayConfirmation?reservationId=617ed337-871e-4a82-8e45-032f6868e650

1、

1）把所有的微博内容分词，并且统计tf，输出：key：词+微博ID value ：1

\* 并且统计N:所有的微博总数

2）统计微博总数，统计词的tf。把统计两个结果分别输出到两个不同的文件中

\* c1\_001,2

\* c2\_001,1

\* count,10000

3）统计微博总数，统计词的tf。把统计两个结果分别输出到两个不同的文件中。分区。

\* 分成两类区：

\* tf：n个区

\* N ：1个区

4）//统计df：词在多少个微博中出现过。

# 25. HortWorks –MapReduce总结

block太小：

我们做个假设 如果 1T 的盘 ，我们上传 小文件到 hdfs ，那么我们是不是只能上传1632个小文件，这个和我们预期远远不相符，hadoop确实是处理大文件有优势，但是也有处理小文件的能力，那就是在shuffle阶段会有合并的，且map的个数和块的数量有关系，如果块很小，那其不是有很多块 也就是很多map了

所以 hdfs中的块，会被充分利用，如果一个文件的大小小于这个块的话那么这个文件 不会占据整个块空间

block太大：

但是该参数也不会设置得过大。MapReduce中的map任务通常一次处理一个块中的数据，因此如果任务数太少（少于集群中的节点数量），作业的运行速度就会比较慢。

HADOOP讲解过程：http://langyu.iteye.com/blog/992916?page=3#comments

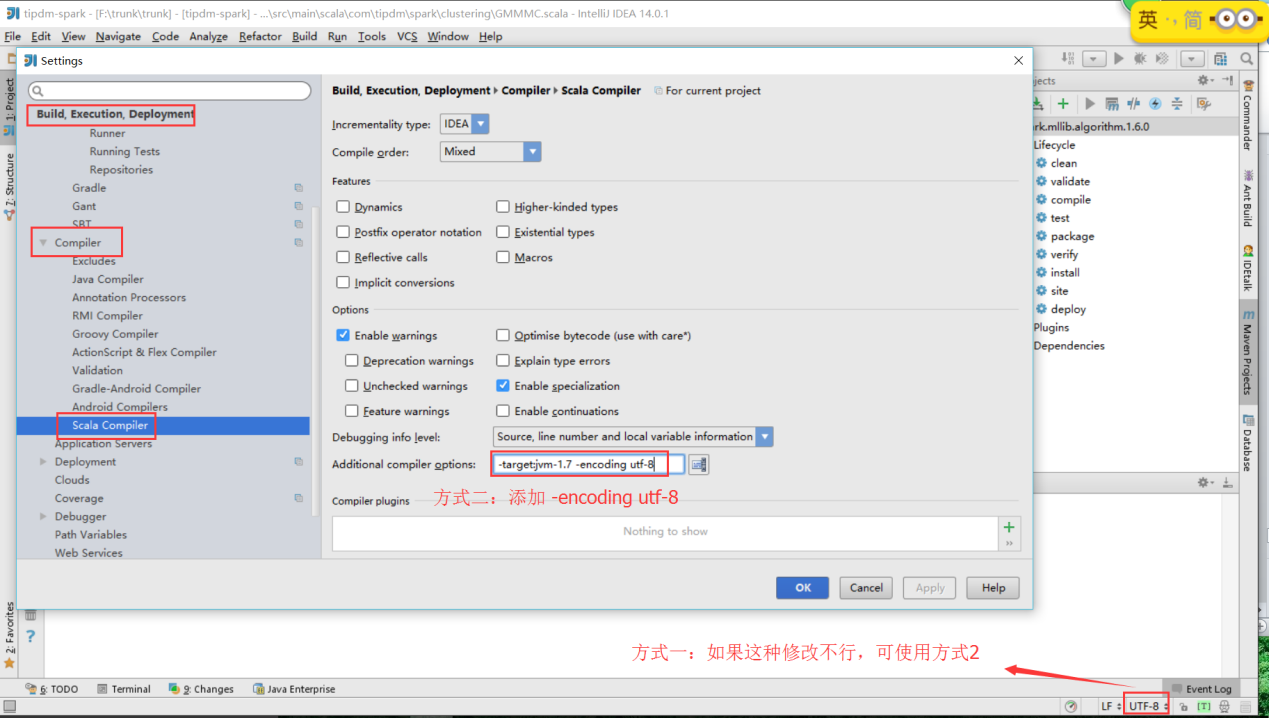
先分区。

combiner的调用是多次的。

reduce的输入输出类型对应一致时，才可考虑用combiner。

hadoop教程：

# 26 IDEA 修改26. scala编码(encoding编码)



# 27. spark 任务提交（包含SparkSQL任务）

伪分布式提交spark任务

1. spark-submit 的yarn-client方式提交任务：

命令：spark-submit --name test116 --master yarn-client --class com.tipdm.spark.clustering.KMeansMC example-1.0-SNAPSHOT.jar "false" "kmeans\_train" "x1,x2,x3" "/user/root/model/kmeans11" "3" "10" "true"

注意：

1）不需要设置：--jars 参数添加datanucleus-api-jdo-3.2.6.jar、datanucleus-rdbms-3.2.9.jar、datanucleus-core-3.2.10.jar。

2）也不需要设置：spark.driver.extraClassPath或在spark-env.sh中添加SPARK\_CLASSPATH路径导入datanucleus-api-jdo-3.2.6.jar、datanucleus-rdbms-3.2.9.jar、datanucleus-core-3.2.10.jar

3）yarn-client可以正确执行，命令行日志并未看到上传sparkSQL执行用到的三个包：datanucleus-api-jdo-3.2.6.jar、datanucleus-rdbms-3.2.9.jar、datanucleus-core-3.2.10.jar，说明这种方式spark内部能自动识别读取datanucleus-api-jdo-3.2.6.jar、datanucleus-rdbms-3.2.9.jar、datanucleus-core-3.2.10.jar。

1. spark-submit 的yarn-cluster方式提交任务（不知为何没有输出文件到HDFS，client方式有输出文件到HDFS）：

1）命令：spark-submit --name test7 --master yarn-cluster --class com.tipdm.spark.clustering.KMeansMC --jars /opt/spark-1.6.1-bin-hadoop2.6/lib/datanucleus-api-jdo-3.2.6.jar,/opt/spark-1.6.1-bin-hadoop2.6/lib/datanucleus-rdbms-3.2.9.jar,/opt/spark-1.6.1-bin-hadoop2.6/lib/datanucleus-core-3.2.10.jar example-1.0-SNAPSHOT.jar "false" "kmeans\_train" "x1,x2,x3" "/user/root/model/kmeans10" "3" "10" "true"

注意：yarn-cluster方式提交需要手动指定datanucleus-api-jdo-3.2.6.jar、datanucleus-rdbms-3.2.9.jar、datanucleus-core-3.2.10.jar三个包，不然报错：javax.jdo.JDOFatalUserException: Class org.datanucleus.api.jdo.JDOPersistenceManagerFactory was not found.

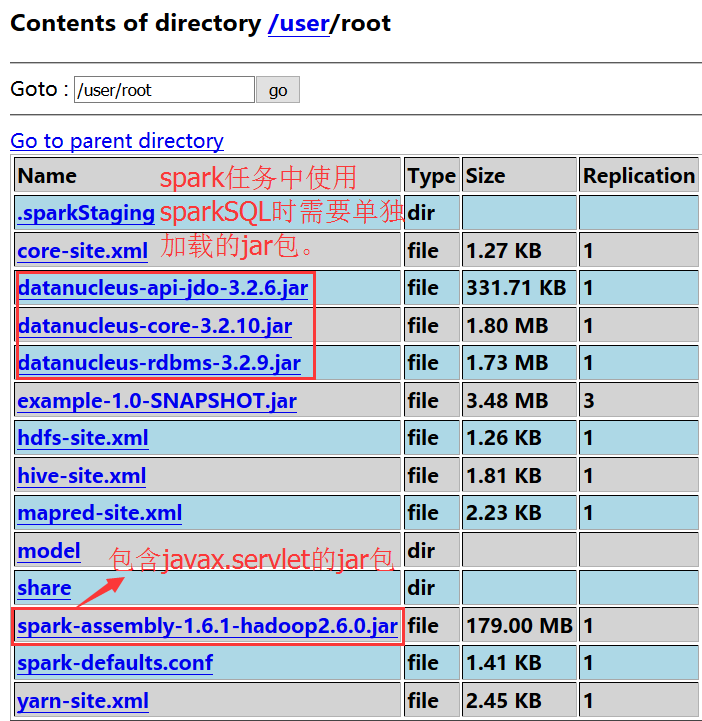
手动指定额外的jar包一般有三种方式，但这里只有两种方式见效，如下说明：

方式一：就是上面的—jars方式，提交时指定。

方式二：编辑spark-defaults.conf,添加：spark.driver.extraClassPath=/opt/spark-1.6.1-bin-hadoop2.6/lib/datanucleus-api-jdo-3.2.6.jar:/opt/spark-1.6.1-bin-hadoop2.6/lib/datanucleus-rdbms-3.2.9.jar:/opt/spark-1.6.1-bin-hadoop2.6/lib/datanucleus-core-3.2.10.jar

注意：在spark-env.sh中添加SPARK\_CLASSPATH引入三个jar包，不行，日志中的classpath内容不包含三个jar包，说明没有添加上去。

3、Windows IDEA java yarn-client方式提交任务：







构造参数时需添加：--addJars，指定sparkSQL的三个包，不然报错：javax.jdo.JDOFatalUserException: Class org.datanucleus.api.jdo.JDOPersistenceManagerFactory was not found.





3、Windows Tomcat、Linux Tomcat 方式提交任务类似：  
几点需要注意：

1）需要把spark-assembly-1.6.1-hadoop2.6.0.jar放到WEB-INF/lib下面，不然报错：YarnClient找不到！但原生的spark-assembly-1.6.1-hadoop2.6.0.jar会和tomcat中的servlet-api.jar冲突，

因此需删除原生spark-assembly-1.6.1-hadoop2.6.0.jar中javax中的servlet文件夹，再放到WEB-INF/lib下面。

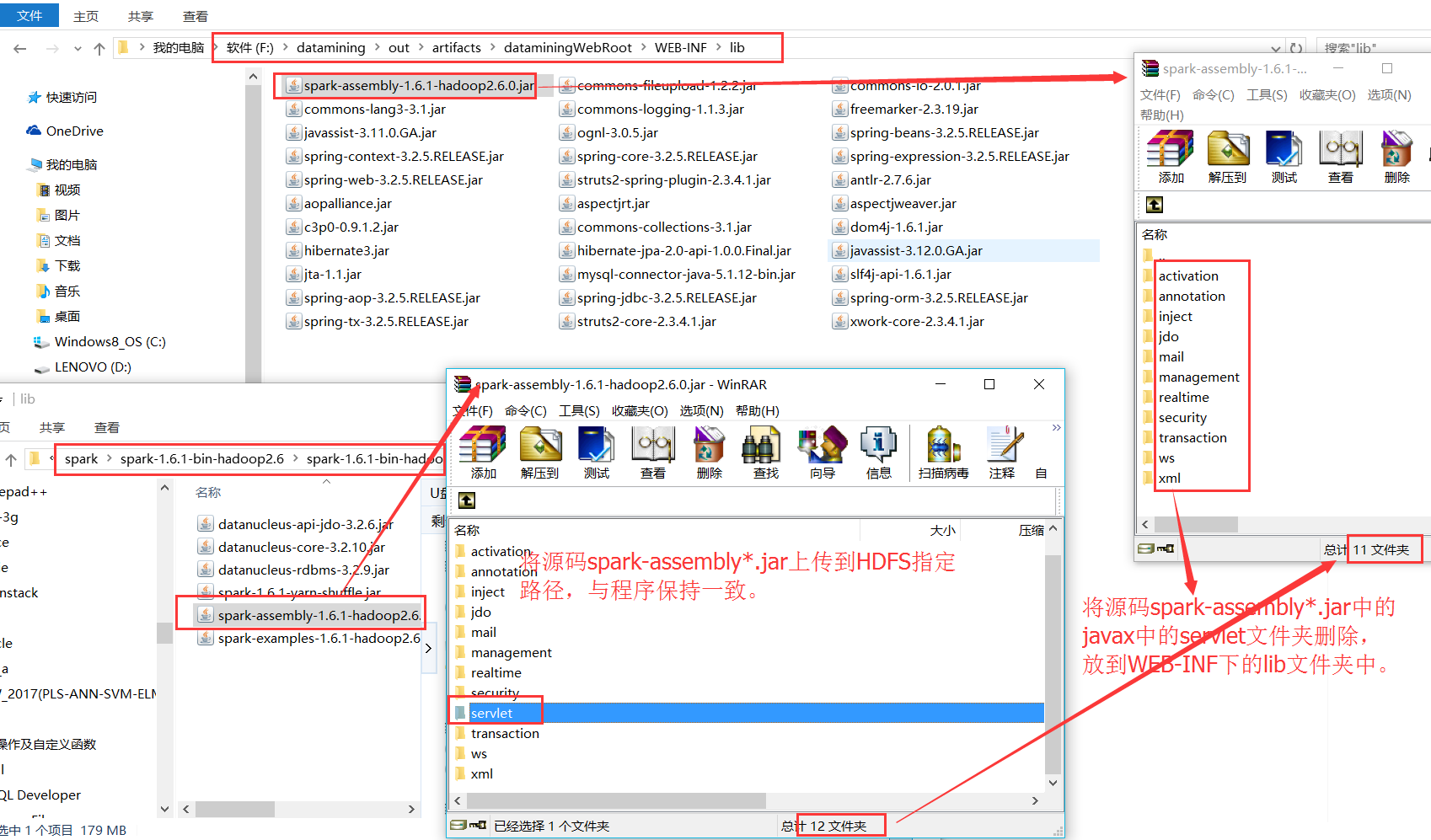
1. 但如果真的使用删除javax.servlet后的spark-assembly-1.6.1-hadoop2.6.0.jar又会报下面的错误，如下图：
2. 因此需要使用设置spark.yarn.jar，可以理解为欺骗tomcat，首先保证本地有删除后的spark-assembly-1.6.1-hadoop2.6.0.jar包，HDFS有原生的，启动IDEA Tomcat可见日志如下：

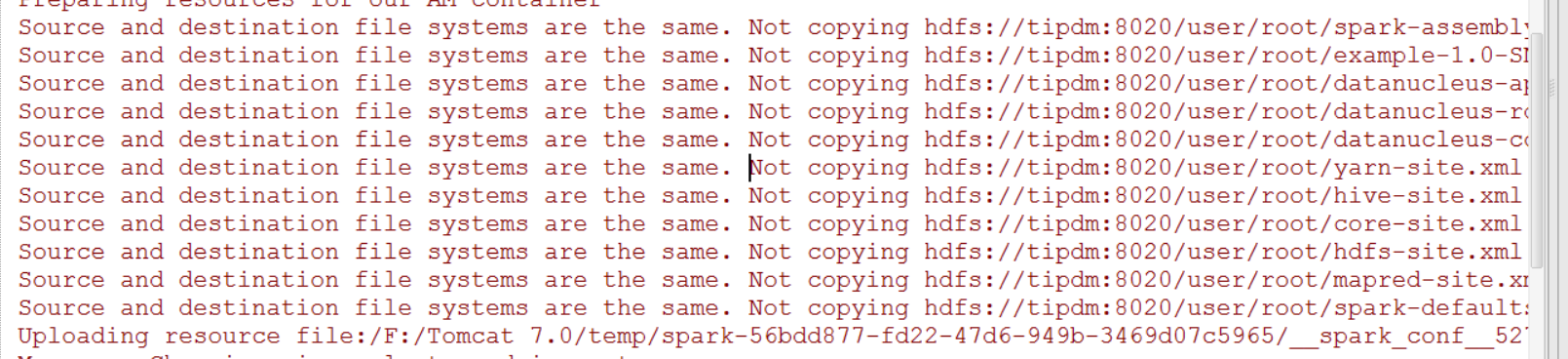
Source and destination file systems are the same. Not copying hdfs://tipdm:8020/user/root/spark-assembly-1.6.1-hadoop2.6.0.jar

可见就使用HDFS上的原生包了！

5）程序相关内容与windows java 提交方式一样。







spark.addJar中的可以是本地路径（windows，linux,或是放在WEB-INF/lib下面，程序中动态获取）。

Windows中的Tomcat 提交时还可以这样指定路径：spark.addJar=F:\\spark\\spark-1.6.1-bin-hadoop2.6\\spark-1.6.1-bin-hadoop2.6\\lib\\datanucleus-api-jdo-3.2.6.jar,F:\\spark\\spark-1.6.1-bin-hadoop2.6\\spark-1.6.1-bin-hadoop2.6\\lib\\datanucleus-rdbms-3.2.9.jar,F:\\spark\\spark-1.6.1-bin-hadoop2.6\\spark-1.6.1-bin-hadoop2.6\\lib\\datanucleus-core-3.2.10.jar

1. 任务提交几种方式：

|  |
| --- |
| 方式1:  val conf = new SparkConf().setAppName("submit job!").setMaster("spark://tipdm:7077")  .setJars(Array("F:\\SubmitSparkJob\\out\\artifacts\\test\\test.jar"))  ============================================================  方式二：  String[] arg0=new String[]{  "--master","spark://tipdm:7077",  "--deploy-mode","client", //此为cluster时不行，找不到JavaSubmitSpark02类，待解决？  "--name","test java submit job to spark22",  "--class","sca01.JavaSubmitSpark01\_", //JavaSubmitSpark01\_中设置的job名称优先于这里的--name值。  "--executor-memory","1G",  "F:\\SubmitSparkJob\\out\\artifacts\\test\\test.jar",  "hdfs://tipdm:8020/root/log.txt",  "hdfs://tipdm:8020/root/log\_java6"  };  SparkSubmit.main(arg0);  ============================================================  方式三：java中使用Client提交，本质就是yarn-cluster  SparkConf sparkConf = new SparkConf();  sparkConf.set("spark.yarn.jar", "hdfs://tipdm:8020/spark-assembly-1.6.1-hadoop2.6.0.jar");  ClientArguments cArgs = new ClientArguments(arg0,sparkConf);  new Client(cArgs,conf,sparkConf).run();  ============================================================  命令行提交：Spark-submit  1．spark-submit方式提交首先是执行org.apache.spark.deploy.SparkSubmit，如果是client模式（不管是yarn或standalone）则通过反射直接执行用户类（--class后面的类）中的main方法。  2. 如果是standalone cluster的方式，则通过反射可能执行org.apache.spark.deploy.rest.RestSubmissionClient或org.apache.spark.deploy.Client中的main方法。其中Client的main方法中调用：  val args = new ClientArguments(argStrings, sparkConf)  new Client(args, sparkConf).run() 运行程序！   1. 如果是yarn-cluster的方式，则通过反射调用org.apache.spark.deploy.Client中的main方法，main方法中再调用：   val args = new ClientArguments(argStrings, sparkConf)  new Client(args, sparkConf).run() 运行程序！  总结：java中的Client方式 或spark-submit的cluster方式最终都要经过（cluster模式通过反射调用）Client类封来执行任务。client方式通过反射直接执行用户类main方法，不经过Client类封装。 |

Spark运行原理及优化

|  |
| --- |
| ===========================================================================  yarn 和 standalone 都可用：  --driver-cores：只在cluster下可用，默认为1。  --executor-cores：cluster和client都可用，(Default: 1 in YARN mode,  or all available cores on the worker in standalone mode)  --driver-memory ：cluster和client都可用，Memory for driver (e.g. 1000M, 2G) (Default: 1024M).  --executor-memory ：cluster和client都可用，Memory per executor (e.g. 1000M, 2G) (Default: 1G).  yarn-only:  --num-executors NUM Number of executors to launch (Default: 2).    yarn模式：<https://www.iteblog.com/archives/1191.html>  cluster模式：<https://www.iteblog.com/archives/1189.html>  yarn与cluster模式区别：<https://www.iteblog.com/archives/1223.html>  如下链接整个系列都不错：  <http://www.cnblogs.com/hseagle/p/3664933.html>  http://blog.csdn.net/wang1127248268/article/details/53089957  spark-stage划分原理：<http://www.cnblogs.com/bonelee/p/6039469.html>  spark运行原理、调优概述：<http://www.cnblogs.com/bonelee/p/6042263.html>  运行原理：  我们使用spark-submit提交一个Spark作业之后，这个作业就会启动一个对应的Driver进程。根据你使用的部署模式（deploy-mode）不同，Driver进程可能在本地启动，也可能在集群中某个工作节点上启动。Driver进程本身会根据我们设置的参数，占有一定数量的内存和CPU core。而Driver进程要做的第一件事情，就是向集群管理器（可以是Spark Standalone集群，也可以是其他的资源管理集群，美团•大众点评使用的是YARN作为资源管理集群）申请运行Spark作业需要使用的资源，这里的资源指的就是Executor进程。YARN集群管理器会根据我们为Spark作业设置的资源参数，在各个工作节点上，启动一定数量的Executor进程，每个Executor进程都占有一定数量的内存和CPU core。  在申请到了作业执行所需的资源之后，Driver进程就会开始调度和执行我们编写的作业代码了。Driver进程会将我们编写的Spark作业代码分拆为多个stage，每个stage执行一部分代码片段，并为每个stage创建一批task，然后将这些task分配到各个Executor进程中执行。task是最小的计算单元，负责执行一模一样的计算逻辑（也就是我们自己编写的某个代码片段），只是每个task处理的数据不同而已。一个stage的所有task都执行完毕之后，会在各个节点本地的磁盘文件中写入计算中间结果，然后Driver就会调度运行下一个stage。下一个stage的task的输入数据就是上一个stage输出的中间结果。如此循环往复，直到将我们自己编写的代码逻辑全部执行完，并且计算完所有的数据，得到我们想要的结果为止。  Spark是根据shuffle类算子来进行stage的划分。如果我们的代码中执行了某个shuffle类算子（比如reduceByKey、join等），那么就会在该算子处，划分出一个stage界限来。可以大致理解为，shuffle算子执行之前的代码会被划分为一个stage，shuffle算子执行以及之后的代码会被划分为下一个stage。因此一个stage刚开始执行的时候，它的每个task可能都会从上一个stage的task所在的节点，去通过网络传输拉取需要自己处理的所有key，然后对拉取到的所有相同的key使用我们自己编写的算子函数执行聚合操作（比如reduceByKey()算子接收的函数）。这个过程就是shuffle。  当我们在代码中执行了cache/persist等持久化操作时，根据我们选择的持久化级别的不同，每个task计算出来的数据也会保存到Executor进程的内存或者所在节点的磁盘文件中。  因此Executor的内存主要分为三块：第一块是让task执行我们自己编写的代码时使用，默认是占Executor总内存的20%；第二块是让task通过shuffle过程拉取了上一个stage的task的输出后，进行聚合等操作时使用，默认也是占Executor总内存的20%；第三块是让RDD持久化时使用，默认占Executor总内存的60%。  task的执行速度是跟每个Executor进程的CPU core数量有直接关系的。一个CPU core同一时间只能执行一个线程。而每个Executor进程上分配到的多个task，都是以每个task一条线程的方式，多线程并发运行的。如果CPU core数量比较充足，而且分配到的task数量比较合理，那么通常来说，可以比较快速和高效地执行完这些task线程。  资源参数调优： num-executors  * 参数说明：该参数用于设置Spark作业总共要用多少个Executor进程来执行。Driver在向YARN集群管理器申请资源时，YARN集群管理器会尽可能按照你的设置来在集群的各个工作节点上，启动相应数量的Executor进程。这个参数非常之重要，如果不设置的话，默认只会给你启动少量的Executor进程，此时你的Spark作业的运行速度是非常慢的。 * 参数调优建议：每个Spark作业的运行一般设置50~100个左右的Executor进程比较合适，设置太少或太多的Executor进程都不好。设置的太少，无法充分利用集群资源；设置的太多的话，大部分队列可能无法给予充分的资源。  executor-memory  * 参数说明：该参数用于设置每个Executor进程的内存。Executor内存的大小，很多时候直接决定了Spark作业的性能，而且跟常见的JVM OOM异常，也有直接的关联。 * 参数调优建议：每个Executor进程的内存设置4G~8G较为合适。但是这只是一个参考值，具体的设置还是得根据不同部门的资源队列来定。可以看看自己团队的资源队列的最大内存限制是多少，num-executors乘以executor-memory，是不能超过队列的最大内存量的。此外，如果你是跟团队里其他人共享这个资源队列，那么申请的内存量最好不要超过资源队列最大总内存的1/3~1/2，避免你自己的Spark作业占用了队列所有的资源，导致别的同学的作业无法运行。  executor-cores  * 参数说明：该参数用于设置每个Executor进程的CPU core数量。这个参数决定了每个Executor进程并行执行task线程的能力。因为每个CPU core同一时间只能执行一个task线程，因此每个Executor进程的CPU core数量越多，越能够快速地执行完分配给自己的所有task线程。 * 参数调优建议：Executor的CPU core数量设置为2~4个较为合适。同样得根据不同部门的资源队列来定，可以看看自己的资源队列的最大CPU core限制是多少，再依据设置的Executor数量，来决定每个Executor进程可以分配到几个CPU core。同样建议，如果是跟他人共享这个队列，那么num-executors \* executor-cores不要超过队列总CPU core的1/3~1/2左右比较合适，也是避免影响其他同学的作业运行。  driver-memory  * 参数说明：该参数用于设置Driver进程的内存。 * 参数调优建议：Driver的内存通常来说不设置，或者设置1G左右应该就够了。唯一需要注意的一点是，如果需要使用collect算子将RDD的数据全部拉取到Driver上进行处理，那么必须确保Driver的内存足够大，否则会出现OOM内存溢出的问题。  spark.default.parallelism  * 参数说明：该参数用于设置每个stage的默认task数量。这个参数极为重要，如果不设置可能会直接影响你的Spark作业性能。 * 参数调优建议：Spark作业的默认task数量为500~1000个较为合适。很多同学常犯的一个错误就是不去设置这个参数，那么此时就会导致Spark自己根据底层HDFS的block数量来设置task的数量，默认是一个HDFS block对应一个task。通常来说，Spark默认设置的数量是偏少的（比如就几十个task），如果task数量偏少的话，就会导致你前面设置好的Executor的参数都前功尽弃。试想一下，无论你的Executor进程有多少个，内存和CPU有多大，但是task只有1个或者10个，那么90%的Executor进程可能根本就没有task执行，也就是白白浪费了资源！因此Spark官网建议的设置原则是，设置该参数为num-executors \* executor-cores的2~3倍较为合适，比如Executor的总CPU core数量为300个，那么设置1000个task是可以的，此时可以充分地利用Spark集群的资源。  spark.storage.memoryFraction  * 参数说明：该参数用于设置RDD持久化数据在Executor内存中能占的比例，默认是0.6。也就是说，默认Executor 60%的内存，可以用来保存持久化的RDD数据。根据你选择的不同的持久化策略，如果内存不够时，可能数据就不会持久化，或者数据会写入磁盘。 * 参数调优建议：如果Spark作业中，有较多的RDD持久化操作，该参数的值可以适当提高一些，保证持久化的数据能够容纳在内存中。避免内存不够缓存所有的数据，导致数据只能写入磁盘中，降低了性能。但是如果Spark作业中的shuffle类操作比较多，而持久化操作比较少，那么这个参数的值适当降低一些比较合适。此外，如果发现作业由于频繁的gc导致运行缓慢（通过spark web ui可以观察到作业的gc耗时），意味着task执行用户代码的内存不够用，那么同样建议调低这个参数的值。  spark.shuffle.memoryFraction  * 参数说明：该参数用于设置shuffle过程中一个task拉取到上个stage的task的输出后，进行聚合操作时能够使用的Executor内存的比例，默认是0.2。也就是说，Executor默认只有20%的内存用来进行该操作。shuffle操作在进行聚合时，如果发现使用的内存超出了这个20%的限制，那么多余的数据就会溢写到磁盘文件中去，此时就会极大地降低性能。 * 参数调优建议：如果Spark作业中的RDD持久化操作较少，shuffle操作较多时，建议降低持久化操作的内存占比，提高shuffle操作的内存占比比例，避免shuffle过程中数据过多时内存不够用，必须溢写到磁盘上，降低了性能。此外，如果发现作业由于频繁的gc导致运行缓慢，意味着task执行用户代码的内存不够用，那么同样建议调低这个参数的值。   资源参数的调优，没有一个固定的值，需要同学们根据自己的实际情况（包括Spark作业中的shuffle操作数量、RDD持久化操作数量以及spark web ui中显示的作业gc情况），同时参考本篇文章中给出的原理以及调优建议，合理地设置上述参数。 |

# 28 .大数据基本知识普及

IaaS、PaaS、SaaS 为三种是[云计算机](https://www.baidu.com/s?wd=%E4%BA%91%E8%AE%A1%E7%AE%97%E6%9C%BA&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLP1DsPjTvPHmkrHNhnjDY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHcdPjTvPjnY" \t "_blank)的有三种服务模型。  
当前SaaS有多种典型应用。如：在线邮件服务、网络会议、[在线杀毒](https://www.baidu.com/s?wd=%E5%9C%A8%E7%BA%BF%E6%9D%80%E6%AF%92&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLP1DsPjTvPHmkrHNhnjDY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHcdPjTvPjnY" \t "_blank)等各种工具型服务。SaaS是应用软件的一个发展趋势。  
当前PaaS的典型实例有微软的Windows Azure平台、GoogleApp engine 等。  
在这种服务模式中，客户不需要购买底动的硬件和平台软件。只需要利用PaaS平台，就能够创建、测试和部署应用程序。  
当前IaaS，在这个[服务类型](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E7%B1%BB%E5%9E%8B&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLP1DsPjTvPHmkrHNhnjDY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHcdPjTvPjnY" \t "_blank)中，最著名的是[亚马逊](https://www.baidu.com/s?wd=%E4%BA%9A%E9%A9%AC%E9%80%8A&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLP1DsPjTvPHmkrHNhnjDY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHcdPjTvPjnY" \t "_blank)提供的AWS（Amazon Web Services）服务，它通过不同的服务方式把自己庞大数据中心的基础设施对外提供服务，比如通过弹性计算服务（EC2）提供虚拟机租用服务，通过简单储存服务（S3）提供存储租用服务等。  
在这种服务型中，用户不用自己构建一个数据中心，而是通过租用的方式来使用基础设施服务，包括服务器、存储和网络等。

Hive:UDF

1. 自定义UDF需要继承org.apache.hadoop.hive.ql.UDF。  
   b）需要实现evaluate函。

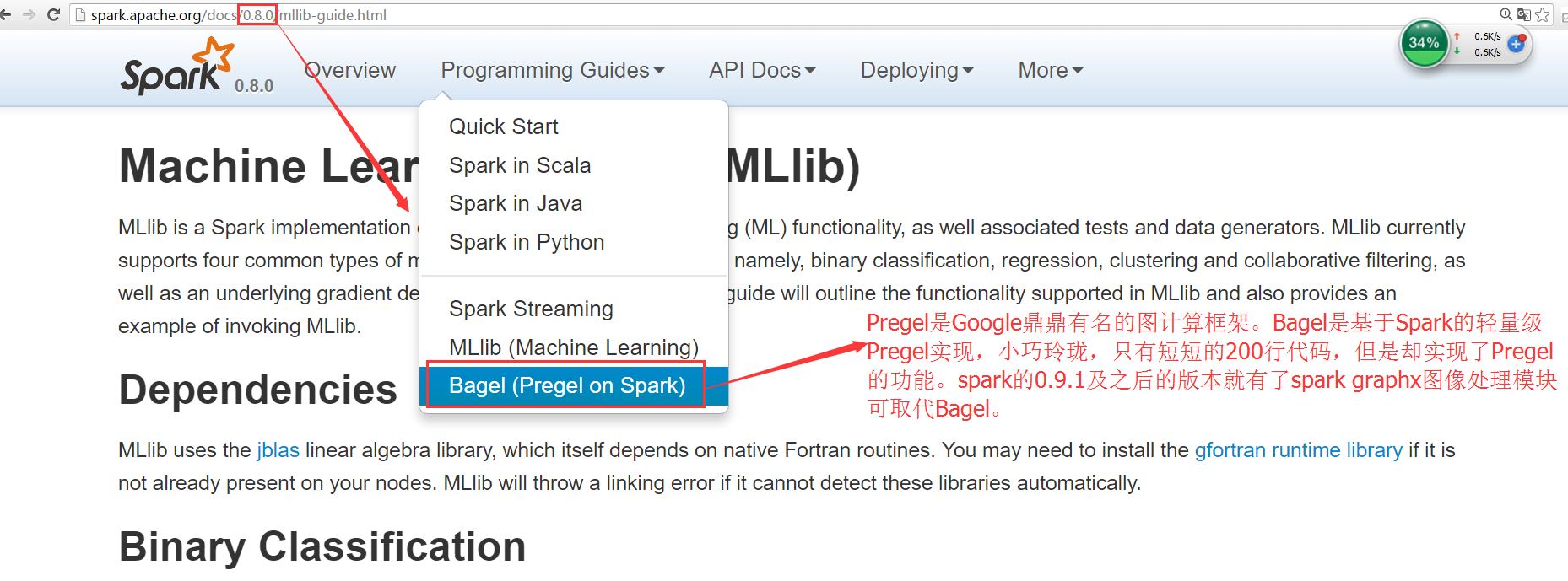
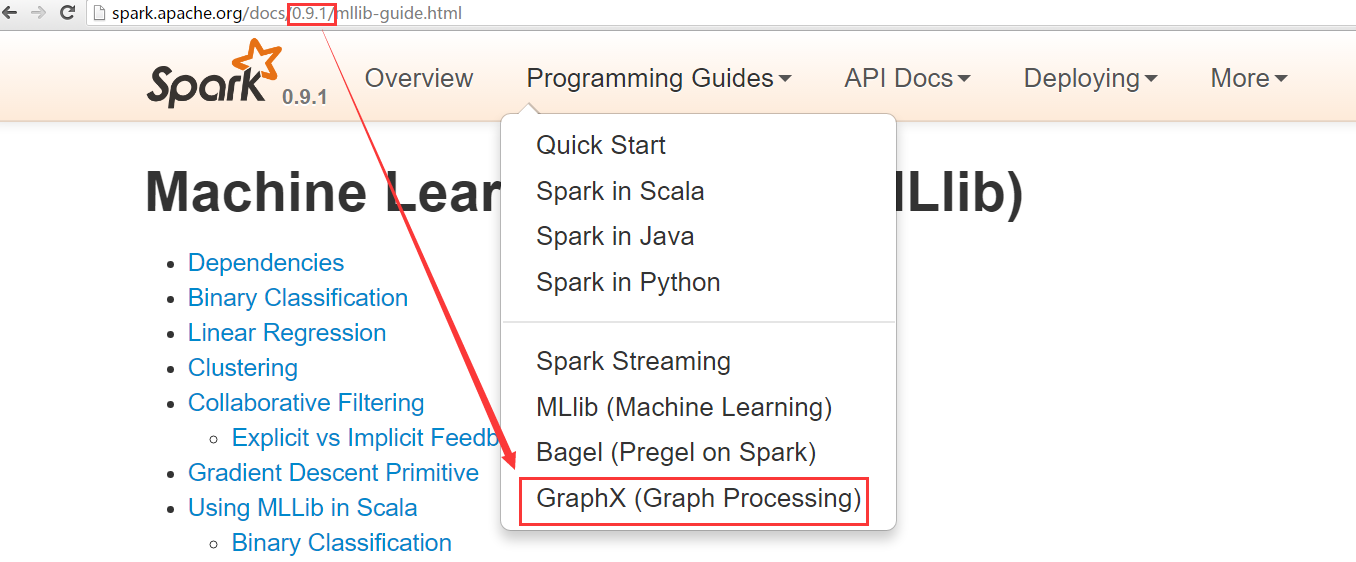
UDAF: User- Defined Aggregation Funcation聚合函数， Sum, Average等。

需要继承UDAF类，内部类Evaluator实UDAFEvaluator接口。

Evaluator需要实现 init、iterate、terminatePartial、merge、terminate这几个函数。

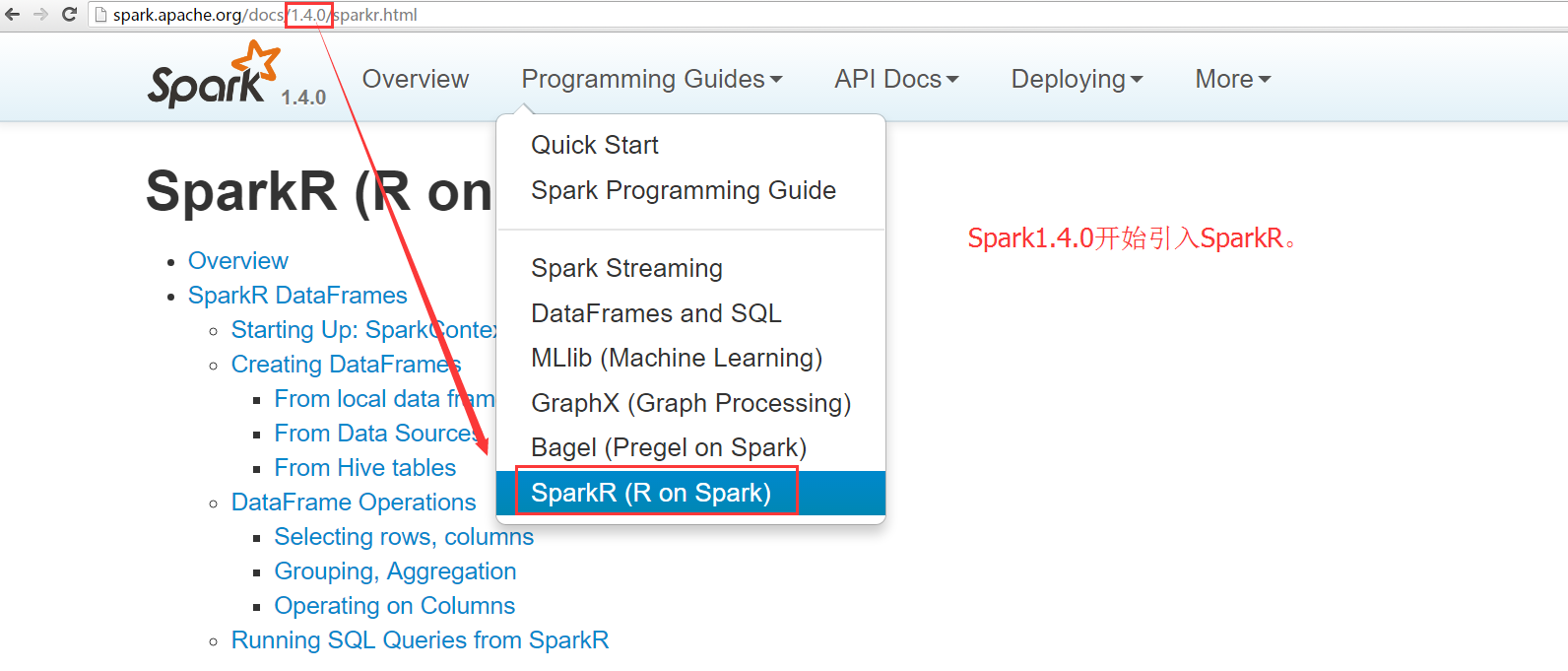
UDTF(User-Defined Table-Generating Functions)  用来解决 输入一行输出多行(On-to-many maping) 的需求。

## 1、Spark图像处理，Graphx 和 Bagel

## 2、Spark 为何会集成R？

R on Spark 是在R里面使用SparkR这个包，来连接Spark集群，读取Hive表，Spark DataFrames。可以说是R使用了Spark的计算引擎，与Hive on Spark一样，是在Hive里使用Spark引擎。R在Spark上，Hive在Spark上。而 Spark 的SparkSQL是在Spark里面读取、操作Hive。



## SparkSQL、Impala、presto对比

## 4、Apache Tez

参考：<http://dongxicheng.org/mapreduce-nextgen/apache-tez/>

# 29．读Spark源码

Spark任务调度图（几种方式）：

# 30. Spark优化

Operations which can cause a shuffle include **repartition** operations like [repartition](http://spark.apache.org/docs/latest/programming-guide.html" \l "RepartitionLink) and [coalesce](http://spark.apache.org/docs/latest/programming-guide.html" \l "CoalesceLink), **‘ByKey** operations (except for counting) like[groupByKey](http://spark.apache.org/docs/latest/programming-guide.html" \l "GroupByLink) and [reduceByKey](http://spark.apache.org/docs/latest/programming-guide.html" \l "ReduceByLink), and **join** operations like [cogroup](http://spark.apache.org/docs/latest/programming-guide.html" \l "CogroupLink) and [join](http://spark.apache.org/docs/latest/programming-guide.html" \l "JoinLink).

# 31. 读Hadoop源码

# 32. Spark算法

算法都是由一些人（数学家/统计学家）研究出来的，写成论文，而spark社区或R社区等只是进行编程实现而已！

1、算法原理及代码：

2、将算法原理抽取成小游戏，类似于捡纸球儿！

3、算法应用，结合电力应用，或者菜品推荐，或者商品推荐，这些应用只要深刻理解算法原理（以及需要注意的问题，比如算法参数，调优RDD分区）就可以去编造应用，总之就是需要**完全清楚大数据的处理过程（数据变化流程）**！数据就是信息，而人，物，事，就是信息的载体，信息的分析就可发现模型，从而预见人事物的发展！

4、算法也是逻辑，他是解决一类现实生活中的问题而产生的通用逻辑，因此可以将算法用现实生活问题反应出来！

## 32.1 KMeans算法

KMeans算法步骤:

第一步，选择 K 个点作为初始聚类中心。

第二步，计算其余所有点到聚类中心的距离，并把每个点划分到离它最近的聚类中心所在的聚类中去。在这里，衡量距离一般有多个函数可以选择，最常用的是欧几里得距离 (Euclidean Distance), 也叫欧式距离。公式如下：

第三步，重新计算每个聚类中所有点的平均值，并将其作为新的聚类中心点。

最后，重复 (二)，(三) 步的过程，直至聚类中心不再发生改变，或者算法达到预定的迭代次数，又或聚类中心的改变小于预先设定的阀值。

KMeans有8个参数，默认值如下：

{k: 2, maxIterations: 20, runs: 1,initializationMode: "k-means||", initializationSteps: 5, epsilon: 1e-4, seed: random}

k:聚类数

runs:并行度,或者选择几组聚类中心.

maxIterations:中止条件之一.

epsilon:阈值,每组中(每个run中)的所有聚类中心（一组有k个聚类中心）与其对应的前一个聚类中心的误差都小于该阈值，才算收敛，只要有一个聚类中心前后误差大于该阈值，则该组（该run）就继续迭代。

initializationMode

initializationSteps

seed

初始化聚类

中心用到的

参数

initialModel

聚类算法有三种方式产生初始聚类中心，如下：

方式一：initRandom(data: RDD[VectorWithNorm])，initializationMode参数选择的是random.

方式二：initKMeansParallel(data), initializationMode参数选择的是k-means||.

方式三：通过设置initialModel参数，就是通过设置训练得到的聚类模型，模型里面有聚类中心。

initializationMode是初始化聚类中心方式，random或k-means||，计算方式如下：

1、random方式计算初始化聚类中心如下：

|  |
| --- |
| val sample = data.takeSample(true, runs \* k, new XORShiftRandom(this.seed).nextInt()).toSeq  Array.tabulate(runs)(r => sample.slice(r \* k, (r + 1) \* k).map { v =>  new VectorWithNorm(Vectors.dense(v.vector.toArray), v.norm)  }.toArray) |

2、k-means||方式计算初始化聚类中心如下：

|  |
| --- |
| 部分代码：  val seed = new XORShiftRandom(this.seed).nextInt()  val sample = data.takeSample(true, runs, seed).toSeq  val newCenters = Array.tabulate(runs)(r => ArrayBuffer(sample(r).toDense))  …..  while (step < initializationSteps) {  ……} |

3、使用已有模型中的聚类信息来作为初始化聚类中心如下：  
注意，需要保证K值相等。如果使用这种方式，很明显，已有模型中只有一组聚类中心，因此runs应为1，算法计算一次（计算多次需要多组聚类中心）。如果设置了该参数，将会覆盖runs,initializationMode ,initializationSteps,seed四个参数，注意runs默认值为1，聚类中心选取逻辑，代码如下：

|  |
| --- |
| val centers = initialModel match {  case Some(kMeansCenters) => {  Array(kMeansCenters.clusterCenters.map(s => new VectorWithNorm(s)))  }  case None => {  if (initializationMode == KMeans.RANDOM) {  initRandom(data)  } else {  initKMeansParallel(data)  } |

|  |
| --- |
| 清单 3. K 选择示例代码片段 val ks:Array[Int] = Array(3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)  ks.foreach(cluster => {  val model:KMeansModel = KMeans.train(parsedTrainingData, cluster,30,1)  val ssd = model.computeCost(parsedTrainingData)  println("sum of squared distances of points to their nearest center when k=" + cluster + " -> "+ ssd)  }) |

问题：

* + - 1. 如何确定K的个数？

通过程序，遍历数组（k的个数），然后每次循环都聚类，并且计算平方和，观察有波动的值！但一般K的个数越大平方和就越小，当K的个数与样本一致时，平方和则为0，但这样的K没有实际意义！

* + - 1. 如何选择初始聚类中心？

使用k-means||算法选取初始聚类中心，原则是保持初始聚类中心间的距离尽可能远！

步骤：

第一步，从数据集 X 中随机选择一个点作为第一个初始点。

第二步，计算数据集中所有点与最新选择的中心点的距离 D(x)。

第三步，选择下一个中心点，使得P(x)=D(x)^2/最大。

第四部，重复 (二),(三) 步过程，直到 K 个初始点选择完成。

遗留问题：随机种子选取k-means||算法原理？

还有没有更好的方法选择随机种子以及确定k的个数？

K值确定、初始点选择方法，KMeans优缺点：<http://www.cnblogs.com/dudumiaomiao/p/5839905.html>

KMeans源码解读、欧氏距离计算：<http://blog.csdn.net/sunbow0/article/details/45673613>

## 32.2 Gaussian Mixture Model（高斯混合模型）

参数5个：

|  |
| --- |
| * *k* * *convergenceTol* is the maximum change in log-likelihood at which we consider convergence achieved. * *maxIterations* * *initialModel* * *seed* |

|  |
| --- |
| val (weights, gaussians) = initialModel match {  case Some(gmm) => (gmm.weights, gmm.gaussians)  case None => {  val samples = breezeData.takeSample(withReplacement = true, k \* nSamples, seed)  (Array.fill(k)(1.0 / k), Array.tabulate(k) { i =>  val slice = samples.view(i \* nSamples, (i + 1) \* nSamples)  new MultivariateGaussian(vectorMean(slice), initCovariance(slice))  })  }  }  ==========================================  var llh = Double.MinValue // current log-likelihood  var llhp = 0.0 // previous log-likelihood  var iter = 0  while (iter < maxIterations && math.abs(llh-llhp) > convergenceTol) {  …  }  //run方法返回值，其实聚类过程就是求weights, gaussians，最后用weights, // gaussians构造GMM模型返回。  new GaussianMixtureModel(weights, gaussians) |

软聚类：每个样本属于类别的概率

GMM优势：

可理解性好(可以看做是多个分布的组合)

速度快(因为EM的高效算法时间复杂度为o(N)）)

学术上比较直观：最大数据似然概率

可以扩展到多个其他分布的混合:多个多项式分布做类别判定。

缺点：与k-means相似，初始化敏感，要手工指定k的个数，不适用于非凸分布数据集。

k-means可以看作是“混合高斯聚类在混合成分方差相等、且每个样本派给一个混合成分时的特例”

处理非凸数据的方法：核方法、基于密度的聚类方法

|  |
| --- |
| GMM算法：（Gaussian Mixture Model）高斯混合模型，其算法原理如下：  首先假设数据服从Mixture Gaussian Distribution，换句话说，数据可以看作是从多个Gaussian Distribution  中生成的（Gaussian分布（也叫做正态(Normal) 分布）），每个 GMM 由 K 个 Gaussian 分布组成，每个  Gaussian 称为一个“Component”，这些 Component 线性加成在一起就组成了GMM 的概率密度函数如下:  K  p(x)= ∑ πk.N(x;μk,σk)  k=1  即是GMM认为数据是从K个高斯函数组合而来的。那么如何用 GMM 来做 clustering 呢？现在有了数据，假定它们是由 GMM 生成出来的，  那么我们只要根据数据推出 GMM 的概率分布来就可以了，然后GMM的K个Component 实际上就对应K 个cluster了。根据数据来推算概率  密度通常被称作 density estimation（密度估计） ，特别地，当我们在已知（或假定）了概率密度函数的形式，而要估计其中的参数的过程被称作“参数估计”。大致思路如下：  现在假设我们有 N 个数据点，并假设它们服从某个分布（记作 p(x) ），现在要确定里面的一些参数的值，例如，在 GMM 中，  我们就需要确定πk（权重），μk,σk。我们的想法是，找到这样一组参数，它所确定的概率分布生成这些给定数据点的概率最大，  而这个概率实际上就等于 prod\_{i=1}^N p(x\_i) ，我们把这个乘积称作似然函数 (Likelihood Function)。通常单个点的概率都很小，许多很小的数  字相乘起来在计算机里很容易造成浮点数下溢，因此我们通常会对其取对数，把乘积变成加和 sum\_{i=1}^N \log p(x\_i)，得到 log-likelihood function  。接下来我们只要将这个函数最大化，亦即找到这样一组参数值，它让似然函数取得最大值，我们就认为这是最合适的参数，这样就完成了参数估计的过程。  GMM模型输出结果：比如聚类数为3  weight=0.500000  mu=[9.099999999999984,9.099999999999984,9.099999999999984]  sigma=  0.006666666666831146 0.006666666666831146 0.006666666666831146  0.006666666666831146 0.006666666666831146 0.006666666666831146  0.006666666666831146 0.006666666666831146 0.006666666666831146  weight=0.210064  mu=[0.10000403125945577,0.10000403125945577,0.10000403125945577]  sigma=  0.006666849264591829 0.006666849264591829 0.006666849264591829  0.006666849264591829 0.006666849264591829 0.006666849264591829  0.006666849264591829 0.006666849264591829 0.006666849264591829  weight=0.289936  mu=[0.09999707927385391,0.09999707927385391,0.09999707927385391]  sigma=  0.006666534351081322 0.006666534351081322 0.006666534351081322  0.006666534351081322 0.006666534351081322 0.006666534351081322  0.006666534351081322 0.006666534351081322 0.006666534351081322  预测结果：可以是该条数据的预测标号，也可是该条数据隶属于每个类别的概率（为一个Double数组）。 |

## 32.3 PIC算法

参考：<http://blog.sina.com.cn/s/blog_573ef4200102wqzi.html>

<http://www.cnblogs.com/haozhengfei/p/82c3ef86303321055eb10f7e100eb84b.html>

原理：样本数据对应图中的每个顶点，定点之间边的权重，代表样本之间的关系（相似度），权重越大，关系越密切，越容易聚到一类。



算法参数：

K

maxIterations

initializationMode: initialization model. This can be either “random”, which is the default, to use a random vector as vertex properties, or “degree” to use normalized sum similarities.

PIC:幂迭代聚类，是一种简单且可扩展的图聚类方法，称之为幂迭代聚类（PIC）。在数据归一化的逐对相似矩阵上，使用截断的幂迭代，PIC寻找数据集的一个超低维嵌入（低纬空间投影，embedding ），这种嵌入恰好是很有效的聚类指标，使它在真实数据集上总是好于广泛使用的谱聚类方法。  
 幂迭代聚类和谱聚类一样，也是基于图的聚类技术，都是先把获得的原始数据集抽象为图结构后（顶点，边和三元组等等）才进一步进行分析的。

算法输入数据必须是：(srcId, dstId, similarity) 三元组数据，且类型为[Long,Long,Double]类型。

聚成3类结果如下：

|  |
| --- |
| 代码：// 输出聚类结果  pic.assignments.foreach { a =>  logger.info(s">>>>>>${a.id} -> ${a.cluster}")  }  pic.save(sc,outputFile)  日志：  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>4 -> 0  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>16 -> 1  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>14 -> 2  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>13 -> 1  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>0 -> 0  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>6 -> 0  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>15 -> 1  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>8 -> 0  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>11 -> 1  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>1 -> 0  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>3 -> 0  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>7 -> 1  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>12 -> 2  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>10 -> 0  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>2 -> 0  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>9 -> 1  16/11/02 02:39:51 INFO clustering.SparkPIC$: >>>>>>5 -> 1 |

## 32.4 BisectingKMeans

源码解析：<http://shiyanjun.cn/archives/1388.html>

参数：

k

maxIterations

minDivisibleClusterSize: the minimum number of points (if >= 1.0) or the minimum proportion of points (if < 1.0) of a divisible cluster (default: 1)

seed: a random seed (default: hash value of the class name)

BisectingKMeans算法：二分（平分）K均值，其中算法参数minDivisibleClusterSize指一个类的最小值（点的最小数目），(>=1是点的数目，<1是占所有点的比例)

以下参考：<http://shiyanjun.cn/archives/1388.html>

|  |
| --- |
| Bisecting k-means聚类算法，即二分k均值算法，它是k-means聚类算法的一个变体，主要是为了改进k-means算法随机选择初始质心的随机性造成聚类结果不确定性的问题，而Bisecting k-means算法受随机选择初始质心的影响比较小。 首先，我们考虑在欧几里德空间中，衡量簇的质量通常使用如下度量：误差平方和（Sum of the Squared Error，简称SSE），也就是要计算执行聚类分析后，对每个点都要计算一个误差值，即非质心点到最近的质心的距离。那么，既然每个非质心点都已经属于某个簇，也就是要计算每个非质心点到其所在簇的质心的距离，最后将这些距离值相加求和，作为SSE去评估一个聚类的质量如何。我们的最终目标是，使得最终的SSE能够最小，也就是一个最小化目标SSE的问题。在n维欧几里德空间，SSE形式化地定义，计算公式如下：  Bisecting k-means聚类算法的基本思想是，通过引入局部二分试验，每次试验都通过二分具有最大SSE值的一个簇，二分这个簇以后得到的2个子簇，选择2个子簇的总SSE最小的划分方法，这样能够保证每次二分得到的2个簇是比较优的（也可能是最优的），也就是这2个簇的划分可能是局部最优的，取决于试验的（kmeans算法运行的，即runs）次数。 Bisecting k-means聚类算法的具体执行过程，描述如下所示：   1. 初始时，将待聚类数据集D作为一个簇C0，即C={C0}，输入参数为：二分试验次数m、k-means聚类的基本参数； 2. 取C中具有最大SSE的簇Cp，进行二分试验m次：调用k-means聚类算法，取k=2，将Cp分为2个簇：Ci1、Ci2，一共得到m个二分结果集合B={B1,B2,…,Bm}，其中，Bi={Ci1,Ci2}，这里Ci1和Ci2为每一次二分试验得到的2个簇； 3. 计算上一步二分结果集合B中，每一个划分方法得到的2个簇的总SSE值，选择具有最小总SSE的二分方法得到的结果：Bj={Cj1,Cj2}，并将簇Cj1、Cj2加入到集合C，并将Cp从C中移除； 4. 重复步骤2和3，直到得到k个簇，即集合C中有k个簇。 |

## 32.5 LDA 算法

参考：

<http://www.jone.tech/?p=30>

<https://www.ibm.com/developerworks/cn/opensource/os-cn-spark-practice6/>

https://github.com/Guadaler/CkoocNLP

<http://blog.csdn.net/pirage/article/details/50219323>

<http://blog.csdn.net/sunbow0/article/details/47336993>

|  |
| --- |
| //1 加载数据，返回的数据格式为：documents: RDD[(Long, Vector)]  // 其中：Long为文章ID，Vector为文章分词后的词向量  // 输入的文件每行用词频向量表示一篇文档  // 可以读取指定目录下的数据，通过分词以及数据格式的转换，转换成RDD[(Long, Vector)]即可  val data = sc.textFile(input)  val parsedData = data.map(s => Vectors.dense(s.trim.split(' ').map(\_.toDouble)))  // Index documents with unique IDs  val corpus = parsedData.zipWithIndex.map(\_.swap).cache()  聚成三类：  val ldaModel = new LDA().  setK(k).  setDocConcentration(docConcentration).  setTopicConcentration(topicConcentration).  setMaxIterations(maxIteration).  setSeed(seed).  setCheckpointInterval(checkpointInterval).  setOptimizer(optimizer).  run(corpus)  val topics = ldaModel.topicsMatrix  for (topic <- Range(0, 3)) {  logger.info("--------Topic " + topic + ":")  for (word <- Range(0, ldaModel.vocabSize)) { logger.info("================" + topics(word, topic)); }  println()  }  Hdfs日志：  16/11/02 04:35:34 INFO clustering.SparkLDA$: --------Topic 0:  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================7.378349741842936  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================6.590818620048125  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================3.493980223689382  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================22.18678814933204  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================5.667853327142138  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================4.669995430025498  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================12.078831427590135  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================2.153918189995087  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================2.925939425775269  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================7.773209994558427  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================19.378736298328157  16/11/02 04:35:34 INFO clustering.SparkLDA$: --------Topic 1:  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================9.730452193749272  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================11.117510817839776  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================4.2030249554947545  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================8.957798409964871  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================10.41486341850608  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================9.060922913795608  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================8.657051356539435  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================4.204204965116604  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================2.4299755637946188  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================7.849742910608576  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================6.390793058572817  16/11/02 04:35:34 INFO clustering.SparkLDA$: --------Topic 2:  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================8.891198064407792  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================11.291670562112099  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================4.302994820815863  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================8.855413440703092  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================8.91728325435178  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================8.269081656178894  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================10.264117215870428  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================3.6418768448883085  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================2.6440850104301115  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================8.377047094832996  16/11/02 04:35:34 INFO clustering.SparkLDA$: ================7.2304706430990215 |

## 32.6 SVD

svd与pca主要用了矩阵的变换计算，删除了那些不重要的维度！

参考：<http://blog.selfup.cn/1243.html>

特征值分解只适用于方阵，而现实世界大多都不是方阵，非方阵可以使用奇异值分解singular value decomposition（SVD），svd可用于pca的过程中。

## 32.7 PCA

参考：<http://blog.selfup.cn/1243.html>

主要的聚类算法可以划分为如下几类：划分方法、层次方法、基于密度的方法、基于网格的方法以及基于模型的方法。

# 33. shell编程

if [] ;then

case var in 1|2|3|4|5) ….;;

esac

for var in $\* do ..done

while [] do … done

expr 1 + 2

a=`expr 1 + 2`

命令：shift 是删除传入脚本的参数，并且的后面的参数向前补齐，比如：

shift

shift

删除脚本的第1、2个参数，3、4的参数填充到1、2，接下来用$1,$2取到的其实是3、4的值。

1、Shell本身是一种用C语言编写的程序，从用户的角度来看，Shell是用户与Linux操作系统沟通的桥梁。用户既可以输入命令执行，又可以利用 Shell脚本编程，完成更加复杂的操作。

2、脚本首行为 #!/bin/sh，或 #!/bin/bash，符号#!告诉系统其后路径所指定的程序即是解释此脚本文件的Shell程序。

3、Shell脚本像高级语言一样，也有变量赋值，也有控制语句。

4、以#开头的行就是注释行。如果一行未完成，可以在行尾加上"，这个符号表明下一行与此行会合并为同一行。shell脚本中只能使用#逐行注释，如果有大段需要注释，可将其放到方法里，下面不调用它即可。

5、执行脚本的时候，要先将脚本文件的属性改为可执行的：chmod +x filename.sh

6、Shell Script是一种弱类型语言，使用变量的时候无需首先声明其类型。

7、a="hello world" 注意：变量赋值时，=号两边不能有空格！

echo $a; 取值的时候要在变量名前加$，$可在引号中用，如：”$a”。果出现混淆的情况，可以使用花括号来区分,例如：echo "Hi, $as" ，$as不存在，需要改为：echo "Hi, ${a}s" 输出 hello worlds。

在shell命令行：

a=”hello world”

echo $a

结果：hello world

a=”hello world”

unset a

echo $a

结果：没有值。

1. #!/bin/sh
2. myUrl="http://see.xidian.edu.cn/cpp/u/xitong/"
3. **unset** myUrl
4. echo $myUrl

上面的脚本没有任何输出。

|  |
| --- |
| echo –e ，其中-e参数指定进行转义。\n换行，\t制表符，[\\反斜杠](file:///\\\\反斜杠)  脚本内容：  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  结果：  www\www\tcom\n  www\www com |

运算符：

参考：<http://c.biancheng.net/cpp/view/2736.html>

字符串：

## 单引号字符串的限制：

* 单引号里的任何字符都会原样输出，单引号字符串中的变量是无效的；
* 单引号字串中不能出现单引号（对单引号使用转义符后也不行）。

## 双引号

双引号的优点：

* 双引号里可以有变量
* 双引号里可以出现转义字符

echo命令：

echo –e “OK! \n” 换行

echo “OK! \c” 不换行

echo `date` 显示命令执行结果

printf命令：

printf 命令的语法：

printf format-string [arguments...]

format-string 为格式控制字符串，arguments 为参数列表。

|  |
| --- |
| * printf 命令不用加括号 * format-string 可以没有引号，但最好加上，单引号双引号均可。 * 参数多于格式控制符(%)时，format-string 可以重用，可以将所有参数都转换。 * arguments 使用空格分隔，不用逗号。  1. # format-string为双引号 2. $ **printf** "%d %s\n" 1 "abc" 3. 1 abc 4. # 单引号与双引号效果一样 5. $ **printf** '%d %s\n' 1 "abc" 6. 1 abc 7. # 没有引号也可以输出 8. $ **printf** %s abcdef 9. abcdef 10. # 格式只指定了一个参数，但多出的参数仍然会按照该格式输出，format-string 被重用 11. $ **printf** %s abc def 12. abcdef 13. $ **printf** "%s\n" abc def 14. abc 15. def 16. $ **printf** "%s %s %s\n" a b c d e f g h i j 17. a b c 18. d e f 19. g h i 20. j 21. # 如果没有 arguments，那么 %s 用NULL代替，%d 用 0 代替 22. $ **printf** "%s and %d \n" 23. and 0 24. # 如果以 %d 的格式来显示字符串，那么会有警告，提示无效的数字，此时默认置为 0 25. $ **printf** "The first program always prints'%s,%d\n'" Hello Shell 26. -bash: **printf**: Shell: invalid number 27. The first program always prints 'Hello,0' 28. $ |

8、条件语句：if while for case 等语句里面可以使用continue和break。

|  |
| --- |
| 三种语法：I  f ... fi 语句；  if ... else ... fi 语句；  if ... elif ... else ... fi 语句  Demo1===========================================   1. a=10 2. b=20 3. **if** [ $a == $b ] 4. **then** 5. echo "a is equal to b" 6. **else** 7. echo "a is not equal to b" 8. **fi**   **Demo2===========================================**   1. a=10 2. b=20 3. **if** [ $a == $b ] 4. **then** 5. echo "a is equal to b" 6. **elif** [ $a -gt $b ] 7. **then** 8. echo "a is greater than b" 9. **elif** [ $a -lt $b ] 10. **then** 11. echo "a is less than b" 12. **else** 13. echo "None of the condition met" 14. **fi** |

Shell Script中if语句的条件部分要以分号来分隔。第三行中的[]表示条件测试，常用的条件测试有下面几种：

[ -f "$file" ] 判断$file是否是一个文件，为目录时条件不成立

[ $a -lt 3 ] 判断$a的数值（数值型）是否小于3，同样-gt和-le,-eq分别表示大于、小于等于、等于。

[ -x "$file" ] 判断$file是否存在且有可执行权限，注意：$file可以是文件或目录

[ "$a" = "$b" ] 判断$a和$b的取值是否相等，字符串或数值，注意：aa 与 “aa”相等。

[ cond1 -a cond2 ] 判断cond1和cond2是否同时成立，-o表示cond1和cond2有一成立

要注意条件测试部分中的空格。在方括号的两侧都有空格，在-f、-lt、=等符号两侧同样也有空格。

|  |  |
| --- | --- |
| if [ -f $1 ];then  echo “hello world”  fi | if [ $1 = $2 -a $# -gt 3 ];then  if [ $1 = $2 -o $# -gt 3 ];then |

多分支结构的case语句，它的语法是：

case 语句匹配一个值或一个模式，如果匹配成功，执行相匹配的命令。case语句格式如下：

case 值 in

模式1)

command1

command2

command3

;;

模式2）

command1

command2

command3

;;

\*)

command1

command2

command3

;;

esac

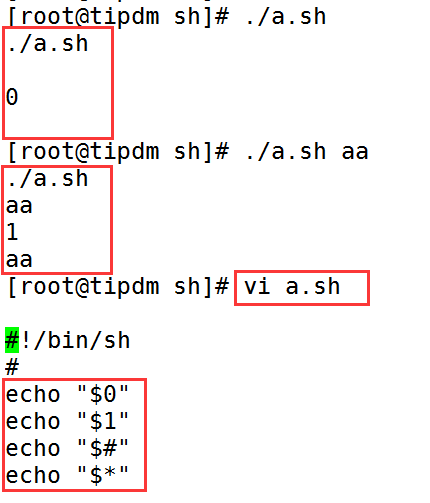
case工作方式如上所示。取值后面必须为关键字 in，每一模式必须以右括号结束。取值可以为变量或常数。匹配发现取值符合某一模式后，其间所有命令开始执行直至 ;;。;; 与其他语言中的 break 类似，意思是跳到整个 case 语句的最后。

取值将检测匹配的每一个模式。一旦模式匹配，则执行完匹配模式相应命令后不再继续其他模式。如果无一匹配模式，使用星号 \* 捕获该值，再执行后面的命令。

|  |
| --- |
| Demo1=====================================  echo 'Input a number between 1 to 4'  echo 'Your number is:'  read aNum  case $aNum in  1) echo 'You select 1'  ;;  2) echo 'You select 2'  ;;  3) echo 'You select 3'  ;;  4) echo 'You select 4'  ;;  \*) echo ‘You do not select a number  between 1 to 4'  ;;  esac  以上case语句等价于(把重复的几句合为一句)：  Read aNum  case $aNum in  1|2|3|4|5) echo “You select $aNum”;;  \*)echo “You do not select a number between 1 to 4”;;  esac  Demo2=================================================   1. option="${1}" 2. **case** ${option} **in** 3. -f) FILE="${2}" 4. echo "File name is $FILE" 5. ;; 6. -d) DIR="${2}" 7. echo "Dir name is $DIR" 8. ;; 9. \*) echo "`basename ${0}`:usage: [-f file] | [-d 10. directory]" 11. **exit** 1 12. ;; 13. **esac** |

$0,$1,$#,$\*,$@,$?区别：

|  |
| --- |
| 特殊变量列表  变量 含义  $0 当前脚本的文件名  $n 传递给脚本或函数的参数。n 是一个数字，表示第几个参数。例如，第一个参数是$1，第二个参数是$2。  $# 传递给脚本或函数的参数个数。  $\* 传递给脚本或函数的所有参数。  $@ 传递给脚本或函数的所有参数。被双引号(" ")包含时，与 $\* 稍有不同，下面将会讲到。  $? 上个命令的退出状态，或函数的返回值。  $$ 当前Shell进程ID。对于 Shell 脚本，就是这些脚本所在的进程ID。 |



1. sh内容如上图：输出四句话。执行./a.sh 和 ./a.sh aa后可见结果，总结可得：
2. $#表示命令行参数的个数，如./a.sh aa中的aa，因此这个命令包含一个参数，在脚本中可用$1获取。（注意：在Shell中，脚本的名称也可看作是一个参数，在脚本中可用$0获取（但不计入参数的总个数中），真实的参数用$1、$2…、${10}、${11}获取）。
3. $\*表示整个参数列表，不包括$0(文件名)，代表的是每个参数本身组成的列表。
4. $@与$\*相同，但是略有区别，请查看Shell特殊变量。
5. $? 函数的返回值.

|  |
| --- |
| $\* 和 $@ 的区别  $\* 和 $@ 都表示传递给函数或脚本的所有参数，不被双引号(" ")包含时，都以"$1" "$2" … "$n" 的形式输出所有参数。  但是当它们被双引号(" ")包含时，"$\*" 会将所有的参数作为一个整体，以"$1 $2 … $n"的形式输出所有参数；"$@" 会将各个参数分开，以"$1" "$2" … "$n" 的形式输出所有参数。   1. #!/bin/bash 2. #使用\转义$输出$\*字符串！ 3. echo "\$\*=" $\* 4. echo "\"\$\*\"=" "$\*" 5. echo "\$@=" $@ 6. echo "\"\$@\"=" "$@" 7. echo "print each param from \$\*" 8. **for** var **in** $\* 9. **do** 10. echo "$var" 11. **done** 12. echo "print each param from \$@" 13. **for** var **in** $@ 14. **do** 15. echo "$var" 16. **done** 17. echo "print each param from \"\$\*\"" 18. **for** var **in** "$\*" 19. **do** 20. echo "$var" 21. **done** 22. echo "print each param from \"\$@\"" 23. **for** var **in** "$@" 24. **do** 25. echo "$var" 26. **done**   执行 ./test.sh "a" "b" "c" "d"，看到下面的结果：  $\*= a b c d  "$\*"= a b c d  $@= a b c d  "$@"= a b c d  print each param from $\*  a  b  c  d  print each param from $@  a  b  c  d  print each param from "$\*"  a b c d  print each param from "$@"  a  b  c  d  $? 可以获取上一个命令的退出状态。所谓退出状态，就是上一个命令执行后的返回结果。退出状态是一个数字，一般情况下，大部分命令执行成功会返回 0，失败返回 1。 |

for循环：

|  |
| --- |
| 格式如下：  for 变量 in 列表  do  command1  command2  ...  commandN  done  列表是一组值（数字、字符串等）组成的序列，每个值通过***空格***分隔。每循环一次，就将列表中的下一个值赋给变量。   1. **for** loop **in** 1 2 3 4 5 2. **do** 3. echo "The value is: $loop" 4. **done**   结果：  The value is: 1  The value is: 2  The value is: 3  The value is: 4  The value is: 5   1. **for** str **in** 'This is a string' //或者去掉单引号’ 2. **do** 3. echo $str 4. **done**   结果：  This is a string  [for var in /root/sh/a\*] #a开头的文件或目录  [for var in /root/sh/\*.sh] #.sh结尾文件或目录  [for var in /root/sh/\*] #该目录下所有文件或目录  [for var in $\*] 执行：./for.sh ./\* ，参数为./\*输出当前目录下内容   1. **for** FILE **in** $HOME/.bash\* 2. **do** 3. echo $FILE 4. **done**   结果：  /root/.bash\_history  /root/.bash\_logout  /root/.bash\_profile  /root/.bashrc |

While循环：

|  |
| --- |
| while [ cond1 ] && { || } [ cond2 ] …; do  …  done   1. COUNTER=0 2. **while** [ $COUNTER -lt 5 ] 3. **do** 4. COUNTER=`expr $COUNTER + 1` 5. echo $COUNTER 6. **done**   expr命令格式：expr argument operator argument  注意运算符（operator）左右都有空格 ，如果没有空格表示是字符串连接。  expr 10 + 10  结果为：20 expr 10 – 5 结果：5  expr 10 / 2 结果：5 expr 10 \\* 2 结果：20 (\转义)  **结果：**  1  2  3  4  5  while [ $# -gt 2 ] && [ $2 = ab ]  do  echo $1  echo $2  echo $3  #exit 1  break  done  echo "........."  测试：./w.sh aa ab ac  结果：  aa  ab  ac  ......... |
| 符号：``可以用来将很多命令的结果保存到一个变量中去。  `expression`表示取出expression的结果。  A=`expression`  COUNTER=`expr $COUNTER + 1` |

跳出循环：

|  |
| --- |
| 1. NUMS="1 2 3 4 5 6 7" 2. **for** NUM **in** $NUMS 3. **do** 4. Q=`expr $NUM % 2` 5. **if** [ $Q -eq 0 ] 6. **then** 7. echo "Number is an even number!!" 8. **continue** 9. **fi** 10. echo "Found odd number" 11. **done** |

Here文档：

|  |
| --- |
| Here用于将多行文本传递给某一命令。Here文档的格式是以<<开始，后跟一个字符串，在 Here文档结束的时候，这个字符串同样也要出现，表示文档结束。在本例中，Here文档被输出给cat命令，也即将文档内容打印在屏幕上，起到显示帮助 信息的作用。  #！/bin/bash  cat <<help  This is content  This is …..  help  测试：./here.sh  结果：  This is content  This is ….. |

if [ $# -lt 3 ]; then  
 4 cat<<HELP  
 5      ren -- renames a number of files using sed regular expressions  
 6   
 7      USAGE: ren 'regexp' 'replacement' fileshttp://www.cnblogs.com/Images/dot.gif  
 8      EXAMPLE: rename all \*.HTM files in \*.html:  
 9      ren 'HTM$' 'html' \*.HTM  
10   
11 HELP  
12      exit 0  
13 fi  
14 OLD="$1"  
15 NEW="$2"  
16 # The shift command removes one argument from the list of  
17 # command line arguments.  
18 shift  
19 shift  
20 # $\* contains now all the files:

for file in $\*; do  
22 if [ -f "$file" ]; then  
23     newfile=`echo "$file" | sed  "s/${OLD}/${NEW}/g"`  
24         if [ -f "$newfile" ]; then  
25             echo "ERROR: $newfile exists already"  
26         else  
27             echo "renaming $file to $newfile http://www.cnblogs.com/Images/dot.gif"  
28             mv "$file" "$newfile"  
29         fi  
30 fi

Shell函数：

Shell Script中也可以使用自定义的函数，其语法形式如下,也可在函数名之前加上function关键字：

[function] functionname()

{

list of commands

[ return value ]

}

函数返回值，可以显式增加return语句；如果不加，会将最后一条命令运行结果作为返回值。Shell 函数返回值只能是整数，一般用来表示函数执行成功与否，0表示成功，其他值表示失败。如果 return 其他数据，比如一个字符串，往往会得到错误提示：“numeric argument required”。  
如果一定要让函数返回字符串，那么可以先定义一个变量，用来接收函数的计算结果，脚本在需要的时候访问这个变量来获得函数返回值。

方法调用及传参：functionname args1 args2

在函数体内部，通过 $n 的形式来获取参数的值，例如，$1表示第一个参数，$2表示第二个参数. 注意，$10 不能获取第十个参数，获取第十个参数需要${10}。当n>=10时，需要使用${n}来获取参数。

像删除变量一样，删除函数也可以使用 unset 命令，不过要加上 .f 选项，如下所示：

unset .f function\_name

如果你希望直接从终端调用函数，可以将函数定义在/etc/profile文件，这样每次登录后，在命令提示符后面输入函数名字就可以立即调用,可以通过unset –f 、unset -v删除env中的函数或变量。

带有return语句的函数：

|  |
| --- |
| 1. funWithReturn(){ 2. echo " get sum of two numbers..." 3. echo "Input first number: " 4. **read** aNum 5. echo "another arg’s number is : $1" #是传进函数的参数 6. echo "The two numbers are $aNum and $anotherNum !"   **return** $(($aNum+$anotherNum))  #也可使用return `expr $n1 + $1`   1. } 2. funWithReturn $1 #这里的$1是脚本在命令行传进的参数，这里传进$1，上边函数中红色$1是找不到值的。（上边的$1不找脚本参数，虽然都是$1，意义不同！） 3. # Capture value returnd by last command 4. ret=$? 5. echo "The sum of two numbers is $ret !" |

shift的作用是将参数列表中的第一个和第二个参数删除，后面的参数依次变为新的第一和第二参数，注意参数列表原本也不包括$0。

输入输出重定向：

一般情况下，每个 Unix/Linux 命令运行时都会打开三个文件：

标准输入文件(stdin)：stdin的文件描述符为0，Unix程序默认从stdin读取数据。

标准输出文件(stdout)：stdout 的文件描述符为1，Unix程序默认向stdout输出数据。

标准错误文件(stderr)：stderr的文件描述符为2，Unix程序会向stderr流中写入错误信息。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | command > file | 将输出重定向到 file。 | | command >> file | 将输出以追加的方式重定向到 file。 | | command < file | 将输入重定向到 file。 | | << tag | 将开始标记 tag 和结束标记 tag 之间的内容作为输入。(成为Here文档) | |  | Here Document 目前没有统一的翻译，这里暂译为”嵌入文档“。Here Document 是 Shell 中的一种特殊的重定向方式，它的基本的形式如下：   1. **command** << delimiter 2. document 3. delimiter |  /dev/null 文件 如果希望执行某个命令，但又不希望在屏幕上显示输出结果，那么可以将输出重定向到 /dev/null：   1. $ **command** > /dev/null   /dev/null 是一个特殊的文件，写入到它的内容都会被丢弃；如果尝试从该文件读取内容，那么什么也读不到。但是 /dev/null 文件非常有用，将命令的输出重定向到它，会起到”禁止输出“的效果。  如果希望屏蔽 stdout 和 stderr，可以这样写：   1. $ **command** > /dev/null 2>&1 |

脚本引用：

|  |
| --- |
| 脚本b.sh引用a.sh中的内容：  脚本a.sh内容：url=[www.www.com](http://www.www.com)  脚本b.sh内容：  #!/bin/bash  . ./a.sh  echo “$url”  结果：  www.www.com |

脚本中执行ssh命令，执行远程命令：

|  |
| --- |
| if [ $# != 1 ]  then  echo "args length must be one!"  echo "Usage:zookeeper start|stop|status"  exit 1  fi  if [ "$1" = "start" ] ;then  echo "node70 zookeeper starting ..."  ssh node70 "source /etc/profile;/usr/local/zookeeper-3.4.6/bin/zkServer.sh start"  echo "node71 zookeeper starting ..."  ssh node71 "source /etc/profile;/usr/local/zookeeper-3.4.6/bin/zkServer.sh start"  echo "node72 zookeeper starting ..."  ssh node72 "source /etc/profile;/usr/local/zookeeper-3.4.6/bin/zkServer.sh start"  fi |

echo和cat

cat a.sh |grep www

echo “aaa” |grep aa

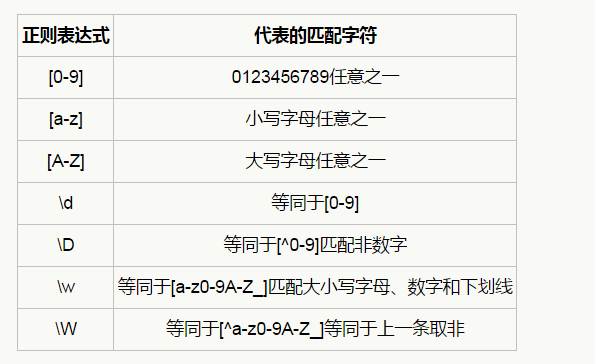
cat后面跟文件,输出文件内容

echo 会把后面的东西都当成字符串输出！

正则表达式：

Python : http://blog.csdn.net/djskl/article/details/44357389

|  |
| --- |
| 字符匹配：  一个点（.）代表一个字符。  比如：.a. 代表3个字符，中间一个字符是a，两边各一个任意字符。  比如：... 代表3个任意字符。  若要匹配点（.）本身，则使用转义字符。  比如：my\... 代表my.后接两个字符。    [...] 代表字符串中的任意一个字符。  比如：[abc] 代表a或b或c中的任意一个字符。  比如：[Mm]y 代表My或者my。    ^出现在[]里，代表“非”  比如：[^a]代表不是a,  [^a-zA-Z0-9]代表不是字母和数字。  位置限定：  行首匹配^：例如：^a，^ab，^a[0-9],分别代表以a,ab,a+数字开头的行。  行尾匹配$：例如：ab$,[abcd]$，以ab，以a|b|c|d结尾。  grep ^$ --color a.sh 匹配空行  grep ';$' --color a.sh 匹配;结尾的行   数量限定：  \*代表前面的字符有0个或者0个以上  比如：a\*b代表ab或aab或aaab等。   \{...\}，制定字符出现个数，代表前边的字符满足制定的个数。  比如：\{1,5\}代表前面的字符有1-5个，\{5\}代表前面的字符有5个。  grep 'w\{3\}' -ni --color a.sh 匹配三个w  6:echo "www\\www\tcom\n"  7:echo -e "www\\www\tcom\n”  {n,}，{,m}，{n,m} 注意：{}需要转义为\{\} |



[\s]表示，只要出现空白就匹配  
[\S]表示，非空白就匹配  
那么它们的组合，表示所有的都匹配，与它相对应的，有[\w\W]等，意义完全相同、  
  
另外要说的一点是，为什么有"."这个通配符了，还要这样的用法。  
  
其实，[\s\S] [\w\W]这样的用法，比较"."所匹配的还要多，因为"."是不会匹配换行的，所有出现有换行匹配的时候，人们就习惯 使用[\s\S]或者[\w\W]这样的完全通配模式。

grep命令：

-c : 输出匹配了多少行  
-i : 不区分大小写

-n : 显示匹配的文件名:行号:行内容 例如：a.sh:3:a=ABcd  
-h : grep多文件时不显示文件名  
-l(L的小写): grep多文件时, 输出包含匹配字符的文件名  
-v : 显示不包含匹配文本的所有行(我经常用除去grep本身)

--color :得到  
基本工作方式: grep 要匹配的内容 文件名, 例如:  
grep 'test' d\* 显示所有以d开头的文件中包含test的行  
grep 'test' aa bb cc 显示在 aa bb cc 文件中包含test的行  
grep '[a-z]\{5}\' aa 显示所有包含字符串至少有5个连续小写字母的串

|  |  |
| --- | --- |
| 脚本a.sh内容：  #!/bin/sh  #  a=ABcd  a1=abcd  a2=ABCD  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n | 脚本b.sh内容：  #!/bin/sh  #  a1=abcd  a2=ABCD |

|  |
| --- |
| grep -c “abcd” \*.sh  结合正则表达式使用grep  grep ‘[a-z]\{4\}’ -n a.sh b.sh #包含至少有5个连续小写字母的字符串  结果：  a.sh:4:a1=abcd  a.sh:6:echo "www\\www\tcom\n"  a.sh:7:echo -e "www\\www\tcom\n"  b.sh:3:a1=abcd  grep '[^a-z0-9=]' -in --color a.sh  结果：  1:#!/bin/sh  2:#  6:echo "www\\www\tcom\n"  7:echo -e "www\\www\tcom\n"  [root@tipdm sh]# grep '^a' -n --color a.sh  3:a=ABcd  4:a1=abcd  5:a2=ABCD  [root@tipdm sh]# grep '^ae' -n --color a.sh  [root@tipdm sh]# grep '^e' -n --color a.sh  6:echo "www\\www\tcom\n"  7:echo -e "www\\www\tcom\n"  [root@tipdm sh]# grep '^a[0-9]' -n --color a.sh  4:a1=abcd  5:a2=ABCD  [root@tipdm sh]# grep '^a[^0-9]' -n --color a.sh  3:a=ABcd  grep 'w\{3\}' -ni --color a.sh  6:echo "www\\www\tcom\n"  7:echo -e "www\\www\tcom\n"  [root@tipdm sh]# grep -n "\-e" a.sh # ”\”对-进行转义  7:echo -e "www\\www\tcom\n"  [root@tipdm sh]# grep abcd a.sh  a1=abcd  [root@tipdm sh]# grep -c abcd a.sh  1  [root@tipdm sh]# grep -c -i abCD a.sh  3  [root@tipdm sh]# grep -i abCD a.sh  a=ABcd  a1=abcd  a2=ABCD  [root@tipdm sh]# grep -i abCD a.sh  a=ABcd  a1=abcd  a2=ABCD  [root@tipdm sh]# grep -in abCD a.sh  3:a=ABcd  4:a1=abcd  5:a2=ABCD  [root@tipdm sh]# grep -i abCD a.sh b.sh  a.sh:a=ABcd  a.sh:a1=abcd  a.sh:a2=ABCD  b.sh:a1=abcd  b.sh:a2=ABCD  [root@tipdm sh]# grep -in abCD a.sh b.sh  a.sh:3:a=ABcd  a.sh:4:a1=abcd  a.sh:5:a2=ABCD  b.sh:3:a1=abcd  b.sh:4:a2=ABCD  [root@tipdm sh]# grep -inh abCD a.sh b.sh  3:a=ABcd  4:a1=abcd  5:a2=ABCD  3:a1=abcd  4:a2=ABCD  [root@tipdm sh]# grep -inhl abCD a.sh b.sh  a.sh  b.sh  [root@tipdm sh]# grep -ci abCD a.sh b.sh  a.sh:3  b.sh:2  [root@tipdm sh]# grep -cih abCD a.sh b.sh  3  2  [root@tipdm sh]# grep -iv abCD a.sh b.sh  a.sh:#!/bin/sh  a.sh:#  a.sh:echo "www\\www\tcom\n"  a.sh:echo -e "www\\www\tcom\n"  b.sh:#!/bin/sh  b.sh:# |

SED:

|  |
| --- |
| **sed(关键字: 编辑)** 以行为单位的文本编辑工具， sed -i可以直接修改文件,主要用于分析文件内容。  使用格式: sed [-nef] '[动作]' [输入文本] -n :一般sed用法中, 来自stdin的数据一般会被列出到屏幕上, 如果使用-n参数后, 只有经过 sed处理的那一行被列出来. -e : 多次操作, 比如你同时又想删除某行, 又想改变其他行, 那么可以用 sed -e '1,5d' -e 's/abc/xxx/g' filename -f : 首先将 sed的动作写在一个档案内, 然后通过 sed -f scriptfile 就可以直接执行 scriptfile 内的sed动作 (没有实验成功, 不推荐使用) -i : 直接编辑, 这回就是真的改变文件中的内容了, 别的都只是改变显示. (不推荐使用) 动作: a 新增, a 后面可以接字符串, 而这个字符串会在新的一行出现. (下一行) c 取代, c 后面的字符串, 这些字符串可以取代 n1,n2之间的行 d 删除, 后面不接任何东西 i 插入, 后面的字符串, 会在上一行出现 p 打印, 将选择的资料列出, 通常和 sed -n 一起运作 sed -n '3p' 只打印第3行 s 取代, 类似vi中的取代, 1,20s/old/new/g  q 退出, 匹配到某行退出, 提高效率  r 匹配到行时读取某文件内容 例如: sed '1r qqq' abc , 将文件qqq中的内容写到第二行。  w file, 将匹配到的行写入文件  例如: sed -n '/m/w qqq' abc , 从abc中读取带m的行写到qqq文件中, 注意, 这个写入带有覆盖性.  Sed、Awk实例参考：  <http://www.cnblogs.com/jankie/archive/2012/01/06/2314478.html>  sed '1,4d' file 指的是把第一到第四行删除。  sed '/aaa/d' file 指的是把还有aaa的那行删除。  sed '/[0-9]\{3\}/d' file 指的是把还有3个数字的行删除。  sed '/^$/d' file 指的是把空白行删除。  sed '/aaa/!d' file 指的是把不含有aaa行的删除。  sed '/aaa/p' file 指的是把含有aaa的行显示出来。  sed -n '/aaa/p' file 指的是把含有aaa的行显示出来，同时显示行号。    使用sed取代字符  例子：  sed 's/aaa/bbb/p' file 将bbb取代aaa，只取代一次。  sed 's/aaa/bbb/pg' file 将bbb取代aaa，全部取代。  sed 's/aaa//p' file 将aaa删除，只删除一次。  sed 's/^...//' file 将每行的前3个字符删除。  sed 's/...$//' file 将每行的后3个字符删除。  sed -n 's/\(a\)/\1b/p' file 找到第一个a然后替换成ab。   * 1. sh脚本内容如下：     删除:d  [root@tipdm sh]# sed '$d' a.sh  #!/bin/sh  #  a=ABcd  a1=abcd  a2=ABCD  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  [root@tipdm sh]# sed -n '$d' a.sh  使用了-n，经过sed处理的那一行被列出来，因为经过sed处理的行, 是个删除操作, 所以不显示。  [root@tipdm sh]# sed -i -e '2d' -e '$d' a.sh #-i模式真实操作了原文件，而非仅仅分析显示  [root@tipdm sh]# sed '/AB/d' a.sh #删除带有AB的行  #!/bin/sh  a1=abcd  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  增加：a  [root@tipdm sh]# sed '/AB/a RRRRRRRRR' a.sh #在所有匹配到AB的行的下面增加 RRRRRRRRRRRRR。  #!/bin/sh  a=ABcd  RRRRRRRRR  a1=abcd  a2=ABCD  RRRRRRRRR  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  [root@tipdm sh]# sed '1,3a RRRRRRR' a.sh #1到3行的每行下面增加RRRRRRRRR  #!/bin/sh  RRRRRRR  #  RRRRRRR  a=ABcd  RRRRRRR  a1=abcd  a2=ABCD  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  替换：c  [root@tipdm sh]# sed '/AB/c RRRRRRRRR' a.sh #将匹配到AB的行替换为RRRRRRRR  #!/bin/sh  RRRRRRRRR  a1=abcd  RRRRRRRRR  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  [root@tipdm sh]# sed '1,3c RRRRRRRRR' a.sh #将第1到第3行的全部内容替换为字 符串RRRRRRRR  RRRRRRRRR  a2=ABCD  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  字符串替换：s  sed '1,$s/n/N/g' a.sh  #!/biN/sh  #  a=ABcd  a1=abcd  a2=ABCD  echo "www\\www\tcom\N"  echo -e "www\\www\tcom\N"  插入：i  sed '/AB/i RRRRRRRRRRR' a.sh #在匹配到AB的行上边插入RRRRRRRRRRRR  #!/bin/sh  #  RRRRRRRRRRR  a=ABcd  a1=abcd  RRRRRRRRRRR  a2=ABCD  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  打印：  Sed -n ‘1p’ a.sh 打印第一行 sed –n ‘1,3’ a.sh 打印1到3行  匹配到的行输出到指定文件：w file  [root@tipdm sh]# sed -n '/AB/w qq' a.sh #将匹配到AB的所有行写入qq文件  [root@tipdm sh]# cat qq  a=ABcd  a2=ABCD  [root@tipdm sh]# sed -n '1,3w qq2' a.sh  [root@tipdm sh]# cat qq2  #!/bin/sh  #  a=ABcd  匹配到的行读取指定文件：r file   * + - 1. t内容如下：   11111  22222  [root@tipdm sh]# sed '/AB/r 1.t' a.sh #匹配到带有AB的行时，读取1.t文件。  #!/bin/sh  #  a=ABcd  11111  22222  a1=abcd  a2=ABCD  11111  22222  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  [root@tipdm sh]# sed '1,3r 1.t' a.sh #读取到1到3行时，读取文件1.t。  #!/bin/sh  11111  22222  #  11111  22222  a=ABcd  11111  22222  a1=abcd  a2=ABCD  echo "www\\www\tcom\n"  echo -e "www\\www\tcom\n"  打印：p  [root@tipdm sh]# sed -n '1,3p' a.sh  #!/bin/sh  #  a=ABcd  [root@tipdm sh]# sed -n '3p' a.sh  a=ABcd  sed -n '/aaa/p' file 指的是把含有aaa的行显示出来，同时显示行号。 |

AWK:

|  |
| --- |
| **参考：<http://www.cnblogs.com/moveofgod/p/3540575.html>**  **awk(关键字:分析&处理)** 一行一行的分析处理  格式：awk '条件类型1{动作1}条件类型2{动作2}' filename,  awk 也可以读取来自前一个指令的 standard input，相对于sed常常用于一整行处理, awk则比较倾向于一行当中分成数个"字段"(区域)来处理, 默认的分隔符是空格键或tab键。 例如: last -n 5 | awk '{print $1 "\t" $3}' 这里大括号内$1"\t"$3 之间不加空格也可以, 不过最好还是加上个空格, 另外注意"\t"是有双引号的, 因为本身这些内容都在单引号内 $0 代表整行 $1代表第一个区域, 依此类推 awk的处理流程是: 1. 读第一行, 将第一行资料填入变量 $0, $1... 等变量中 2. 依据条件限制, 执行动作 3. 接下来执行下一行 所以, AWK一次处理是一行, 而一次中处理的最小单位是一个区域 另外还有3个变量, NF 每一行处理的字段数, NR 目前处理到第几行 FS 目前的分隔符 逻辑判断 > < >= <= == !== , 赋值直接使用= cat /etc/passwd | awk '{FS=":"} $3<10 {print $1 "\t" $3}' 首先定义分隔符为:, 然后判断, 注意看, 判断没有写在{}中, 然后执行动作, FS=":"这是一个动作, 赋值动作, 不是一个判断, 所以写在{}中。 BEGIN END , 给程序员一个初始化和收尾的工作, BEGIN之后列出的操作在{}内将在awk开始扫描输入之前执行, 而END{}内的操作, 将在扫描完输入文件后执行. awk '/test/ {print NR}' abc 将带有test的行的行号打印出来, 注意//之间可以使用正则表达式 awk {}内, 可以使用 if else ,for(i=0;i<10;i++), i=1 while(i<NF) 可见, awk的很多用法都等同于C语言, 比如"\t" 分隔符, print的格式, if, while, for 等等  awk 是相当复杂的工具, 真正使用时, 再补充吧. (有关工具的picture)  例子：  awk '/aa/' file 显示含有aa的行。  awk '{print $1, $2}' file 显示第一第二字段。  awk '/aa/{print $1, $2}' file 将含有aa的行的第一和第二个字段显示出来。  awk -F: '/^root/{print $1, $2}' /etc/passwd 指定:为分隔符，打印第一和第二字段。  awk -F: 'BEGIN{OFS="+++"}'/^root/{print $1, $2} /etc/passwd 以:为分隔符，打印第一和第二段，而且彼此用+++分开。 |

34、SparkSQL总结

HiveContext继承SQLContext,sqlContext.sql(可以执行数据导入导出创建表等等).show()

sqlContext.sql()返回DataFrame,.rdd可转化为RDD[Row]。

RDD与DataFrame之间的转换：dataFrame.rdd

rdd到dataFrame:case class Person (first:String,second:String)

1. .map(row=>Person(row(0),row(1))).toDF()

2. schema = StructType(Seq(StructField(“name”,StringType,true),

StructField(“age”,IntegerType,true)))

sqlContext.createDataFrame(rdd\_row,schema)

sqlContext读取json文件，得到DataFrame

读取关系型数据库Mysql，只不过是得到的数据如何分区的区别！

# 34. 自然语言处理（NLP）

歧义现象的广泛存在使得消除它们需要大量的知识和推理，这就给基于语言学的方法、基于知识的方法带来了巨大的困难，因而以这些方法为主流的自然语言处理研究几十年来一方面在理论和方法方面取得了很多成就，但在能处理大规模真实文本的系统研制方面，成绩并不显著。研制的一些系统大多数是小规模的、研究性的演示系统。

目前存在的问题有两个方面：一方面，迄今为止的语法都限于分析一个孤立的句子，上下文关系和谈话环境对本句的约束和影响还缺乏系统的研究，因此分析歧义、词语省略、代词所指、同一句话在不同场合或由不同的人说出来所具有的不同含义等问题，尚无明确规律可循，需要加强语用学的研究才能逐步解决。另一方面，人理解一个句子不是单凭语法，还运用了大量的有关知识，包括生活知识和专门知识，这些知识无法全部贮存在计算机里。因此一个书面理解系统只能建立在有限的词汇、句型和特定的主题范围内；计算机的贮存量和运转速度大大提高之后，才有可能适当扩大范围.

以上存在的问题成为自然语言理解在[机器翻译](http://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E7%BF%BB%E8%AF%91" \t "_blank)应用中的主要难题，这也就是当今机器翻译系统的译文质量离理想目标仍相差甚远的原因之一；而译文质量是机译系统成败的关键。中国数学家、语言学家[周海中](http://baike.baidu.com/item/%E5%91%A8%E6%B5%B7%E4%B8%AD" \t "_blank)教授曾在经典论文《机器翻译五十年》中指出：要提高机译的质量，首先要解决的是[语言](http://baike.baidu.com/item/%E8%AF%AD%E8%A8%80" \t "_blank)本身问题而不是[程序](http://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F" \t "_blank)设计问题；单靠若干程序来做机译系统，肯定是无法提高机译质量的；另外在人类尚未明了大脑是如何进行语言的模糊识别和[逻辑判断](http://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E5%88%A4%E6%96%AD" \t "_blank)的情况下，机译要想达到“信、达、雅”的程度是不可能的。

瓶颈

目前自然语言处理技术的两大瓶颈就是大规模语料数据的建设，以及语义分析的进一步完善。

主流的自然语言处理技术是以统计机器学习为基础的，这就需要大规模的语料库。在很多任务中，这些语料库是需要人工构建的，这是非常费力的工作。因此，数据共享是一个可以促进研究发展的必不可少的因素。可以说，自然语言处理的快速发展离不开一些开源的语料库，比如WordNet、PennTreebank等。第二，任何语料库无论大小类型，都难以囊括某个领域的全部案例；而且，语料库的标注体系往往难以把握，类别划分过粗，则无法全面、细致地描述语言，类别划分过细，则标注信息过于庞大、降低标注效率，统计数据的稀疏问题严重，训练出来的模型健壮性差。第三，因为人工标注的语料库毕竟是费时费力的工作，因此还需要从模型和算法方面去研究如何利用大量的无人工标注或部分标注的数据，也就是半监督学习，但这方面的研究还不是特别成熟。

自然语言处理技术的

另一大瓶颈就是如何精确地表现自然语言的语义，比如在人机交互过程中，首先就要理解用户的意图，而这里“用户的意图”就是语义。目前业界常用的方法有两种：基于知识或语义学规则的语义分析方法和基于统计学的语义分析方法。尽管两类方法都能在一定程度上进行自然语言语义的推导以及信息之间关联的判别，但是基于知识与语义学规则的方法无法覆盖全部语言现象、推理过程复杂，无法处理不确定性事件，规则间的相容性和适用层次范围存在缺陷和限制，知识和语义规则的建立是瓶颈问题；而基于统计学的方法则过多地依赖于大规模语料库的支持，性能依赖语料库的优劣，易受数据稀疏和数据噪声的干扰，正如之前提到，大规模语料库的建立和语料质量的保证仍是瓶颈问题。

**挑战**

随着智能硬件技术与移动技术的蓬勃爆发，自然语言处理技术的应用趋势也发生了变化。一方面用户要求自然语言处理技术可以精准地理解自己的需求，而且直接给出最匹配的答案，而非简单地给出Url让用户自己去找答案（起码目前代表业内较高水平的小度机器人还是这样做的）。另一方面是需要自然语言处理技术可以与用户进行对话式搜索与智能交互，例如针对“我到哪里可以买到漂亮衣服？”互联网针对衣服款式的定位、价钱的定位等条件与用户进行对话与交互，通过基于自然语言处理技术的搜索引擎来步步引导用户，帮助用户逐渐发现自己的真实需求，进而给出最优答案。第三方面，需要自然语言处理技术对用户进行“画像”，提供“主动推荐、不问即得”的个性化推荐服务。由于每个人各个方面的生活需求（尤其是娱乐方面的需求）不尽相同，自然语言处理技术可以根据用户的浏览历史来挖掘用户的喜好，进而针对用户的喜好进行精准式推荐。

在这样的需求下面，对于自然语言处理技术的未来发展提出了很大的挑战。它要求未来的自然语言处理技术能够做到：

需求识别。通过用户提出了多种多样的、复杂的、基于情感式的、语意模糊的需求进行深刻分析，精确地理解用户的需求。

知识挖掘。经过海量的网络数据与知识的挖掘分析，将各种结构化、非结构化、半结构化的知识进行组织与梳理，最终以结构化、清晰化的知识形式完整地呈现给用户。

用户引导。这与对话式智能交互相关，不仅根据用户的需求来提供“建议”，还能“猜测”用户可能会有什么未想到、未提出的需求，从而“先人一步”为用户提供相关的扩展信息。

结果组织和展现。由于用户更加青睐直接的答案，答案的形式可以是唯一答案、聚合答案、图片、多媒体的形式，这就要求自然语言处理技术能够将挖掘出的信息进行有效地组织与整理，以条理化、简洁化、直接化的形式呈现给用户。

=============================================

百度：王海峰问答 <http://www.iyiou.com/p/40845>

5、未来十年里搜索引擎会有怎样的发展？

现在一说到搜索引擎，大多数人会想到搜索框和搜索结果。那么未来的搜索引擎会是什么样子呢？现在还不好说。但是我们都愿意看到更加强大的搜索引擎，能够让我们在任意场景和产品中利用视觉、听觉甚至触觉等交互形式搜索。搜索将会无处不在。

首先，对用户意图和内容更加深入的理解，以及更加精准的对意图与内容的匹配，都将大大增强搜索引擎的使用体验。理解用户意图不再仅依赖一次查询，而是包括更加复杂的搜索上下文、时间、位置、设备和用户的个性特征。而对内容的理解也将大大深入，不仅能更好的理解语义、语境、观点，还能理解内容的其他各个方面，从内容中抽取出知识。意图与内容的匹配也将考虑上述所有因素，提供更加贴合个人特征与具体语境的搜索结果。此外，搜索引擎也会更像“应答引擎”和“执行引擎”，能够直接回答问题或执行用户指令。

其次，搜索引擎与用户的交互界面也将有更多创新变化。除键盘输入外，还将更加广泛地使用语音和图片等输入方式。随着语音、图像等技术更加成熟和实用，用户将可以使用更加高效、便捷的多模搜索。自然语言交互将成为搜索引擎的主流交互模式。用户可以与搜索引擎对话，直接告诉搜索引擎他们的需求，这比打开键盘输入要方便实用得多。用户可以通过多轮对话与搜索引擎交互，百度是最早尝试运用这种全新的交互方法提升用户搜索体验的公司之一。

第三，搜索将不仅仅是使用搜索引擎，还会融入到各种产品中。比如说，搜索将成为智能硬件产品的核心功能。未来，搜索将无处不在，随处可见。而且，我们还将重新定义可搜索的内容，不仅包括当前搜索引擎已经索引的内容，还将纳入更多服务、物体、设备和数据。

搜索引擎已是大家日常生活中必不可少的工具。用户需求决定搜索引擎的发展方向，而技术进步决定了搜索引擎的发展高度。

7、你如何看待自己从科学家到互联网公司业务副总裁的职业转变？

我热爱技术，并且喜欢投入到技术研究中去。我相信技术可以改变世界。百度为我提供了一个理想的平台，在这里，我的技术工作得以快速、直接地惠及用户。这就是我加入百度的首要原因。我在百度的前几年负责包括NLP、语音、图像、数据挖掘、知识图谱、机器学习和深度学习等在内的技术团队。后来，我意识到伟大的产品是连接技术和广大用户的桥梁，并能推动技术进步。伟大的产品不仅需要先进的技术，更需要出色的产品设计、运营和高效管理。因此，我逐步从只带技术团队转变为全面的业务管理者。我现在的团队超过3000人，包括技术、产品、运营等各方面人才，他们年轻、充满活力、富有激情。我们同心协力，致力为用户提供更好的技术和产品，进而使人们的生活更美好。

在管理大型综合业务团队时，首先要设定战略和目标，然后组建合适的执行团队。良好的制度和文化对于一个大团队来说非常重要，这是业务健康发展的支持和保证。与此同时，我会始终关注技术突破，用户需求的演变以及整个社会环境的发展变化。

8、“百度”这个词的意思是什么？

我们的名字“百度”，其灵感来源于800年前中国宋代的一首词。“....。.众里寻他千百度，蓦然回首，那人却在灯火阑珊处。”百度的字面意思是“数百次”，代表对理想的不懈追求探索。

9、在未来5到10年，我们将在自然语言处理领域看到哪些进步？

在机器翻译、语义理解、问答和对话技术方面将会有重大突破。而且这些技术将被广泛应用，使得人与计算机之间、人与各种智能硬件之间以及人与人之间的沟通和交互方式都会发生改变。

上述技术进步将受益于以下四个方面的技术进展：大数据、学习机制、知识图谱、推理和规划。

大数据——随着互联网的持续蓬勃发展，数据的数量和类型都在高速增长。即便是那些传统行业也开始把他们的数据迁移到线上，一切都在线上被连接起来。数据的价值随着物联网发展会越来越大。

学习机制——学习机制也将持续进步，使我们能从更大规模数据中学习到更多知识。

知识图谱——有了更大规模数据和更强大的学习机制，我们可以构建更完善更庞大的知识图谱，对整个现实世界进行建模。

推理和规划——有了更完善更庞大的知识图谱，我们可以在推理和规划方面取得突破，推理和规划的能力将使自然语言处理系统变得更加智能。

在应用深度学习之前，基于统计的、规则的、实例的方法我们都用了。比如规则方法，擅长抽象语言知识并显式地表示出来，比如语法知识、局部的规则等。

从一种语言到另一种语言并不是完全依靠规则的，如果有限的语法能覆盖所有语言现象，翻译这件事就会变得非常简单。现实中语言是非常复杂的，表示很灵活，很多时候并不是从语法演绎出来，而是约定俗成就这么说，这时候基于实例的方法就会效果更好、效率更高。就像我们学英语时，很多时候不需要去分析，一听到中文，相应的英文就会脱口而出。

机器相对人来说有很多更擅长的能力，但也有一些远不如人的方面。例如，让搜索匹配到合适的网页，但深层的基于背景知识进行深层次的理解及联想则比较困难。还有，比较个大小长短的，对机器来说易如反掌，但要真正去推理则很困难。再比如，机器可以模仿人来写诗，但让机器真正有感而发去搞艺术创作则很难。总结一下，机器很善于匹配、比较、模仿，但要具备像人一样的理解、推理、创造能力，则还有很长的路要走。

知识很重要，所以我们现在很重视建设知识图谱。知识图谱的建设已经是非常浩大的工作，而如何利用这些知识进行理解、推理，是更复杂的事。

语言的复杂性在于，语言不仅仅是表面的字符串，语言的内涵太丰富了，语言实际上是人对整个客观及主观世界的认知、描述和表达。

*机器之心：那 NLP 领域，接下来需要着重解决的是哪些问题？*

**王海峰**：根本问题还是语言的分析理解，语言的生成，以及知识的掌握和运用。

*机器之心：从您开始研究机器翻译，到现在机器翻译都有哪些比较重要的变化？*

**王海峰**：*之前说过的四种方法，基于规则的、实例的、统计的、神经网络的，每种方法我都经历过，每个方法都是一个很大的变化。*

从根本上，我认为还是我们所拥有的基础在变。比如数据的基础，我记得刚来百度的时候，那时候特别开心，因为原来我们用统计方法找一些语料非常困难，几十万句对语料就觉得很好了。然而在百度，通过互联网挖掘到的语料要远远比这个数字大，所以百度翻译效果迅速地就上去了。

您知道有什么方面AI（人工智能）实际上能帮助扩展或者训练人类的智能吗？例如谷歌的alphaGO，专业的围棋选手是否可能通过学习它的推理过程获得提升？  
【吴恩达】We saw this happen after Gary Kasparov lost to Deep Blue. Human chess players are now far better through learning from and also partnering with computer chess players. I've heard of this starting in Go as well, but that feels like it's in an earlier stage. But more generally, I see a lot of opportunities for computers to supplement the human brain. I am especially excited about online education. I think MOOCs like Coursera and open.163.com have been a great start. I hope that online education becomes more adaptive and flexible over time, and that computers can really help customize our learning experiences, the way a personal tutor might.其实从Gary Kasparov输给深蓝之后，这种情况就发生了。现在的象棋选手可以通过向电脑棋手学习或者与其对战而提高自己的水平。我也听说了关于围棋这方面的消息，但是这么说还为时尚早。更一般地说，我看到了许多计算机可以来弥补人脑的机会。我对在线教育很感兴趣，我认为像Coursera和网易公开课都是一个很好的开始；我希望随着时间的推移，在线教育将来可以更加灵活、更能适应需求，可能会成为大家的私人导师。

人工智能成为私人教师！对话交互，传道，循循善诱，老师是最需要智慧的！十年树木，百年树人！

我的理想：

***自然语言处理， 到反观思维，到静坐禅，一念既生，当即寂灭，修正思维，弘扬传统文化！***

《科学日报》最新报导：一个为期8周的正念禅修专业培训，可以在很大程度上改善人的大脑记忆、增加自我认知敏感度、让人能够更好体会他人的情绪和想法，并可有效缓解压力。这项研究由美国麻萨诸塞州总医院的一个研究小组引领进行，其研究结果在本月月底将以论文形式发表于美国《神经影像心理研究》学术期刊上。据称，本次研究首次以书面形式公开发表冥想对人脑灰质所产生的改变和影响。  
  
来自麻省总医院的Sara Lazar博士发表意见说：“长久以来，冥想练习被公认可以带给人内心的平和宁静和生理的放松，但长期进行冥想禅修者还指出，冥想对人的内心感受和认知均有裨益，而且这种良好感觉往往可以持续一整天。” Lazar博士的研究小组最新的研究指出，冥想给人们各方面带来的改善和其大脑结构的改变有关。有些人虽然练习冥想但并未感受到明显改善，往往是因为他们花费了过多的时间来享受放松的感觉，而非努力让自己上升到更高境界。  
  
Lazar的研究小组和其他更多的研究者先前就发现，有经验的禅修者比起新手，其大脑内负责注意力和综合情绪的那部分皮层更厚。但当时他们并没有充足的证据证明，这些大脑结构的变化是因为练习冥想而引起的。  
  
在本次研究中，共有16位研究对象参加了在麻省大学正念禅修中心进行的一个为期8周的培训。培训的主题是基于正念禅修理论的减压训练。麻省大学的正念禅修中心致力于训练人的非主观意识、感知力、情感及心理状态。受训者在特定的音乐中练习冥想，并需每日严格在某个固定的时间点练习一段固定的时长达到预期效果。研究小组分别在培训开始前后两周为受训者进行了脑部的磁共振成像扫描，同时也为对照组的人员进行大脑扫描，以在过后进行比对研究。  
  
练习冥想的组员在受训期间每天花27分钟进行禅修，并在期间回答了研究小组编制的问卷。在培训开始之前他们也回答了同样一份问卷，但在受训后其结果明显比先前大有改善。而从磁共振扫描的影像照片来看，除了之前就已发现的特定区大脑皮层增厚外，还发现海马区的脑灰质也变厚了，该区的脑灰质被公认和人的学习、记忆能力有关；此外，一些与意识、激情和内省相关的结构区也有微妙变化。受训者提出压力感减小，研究人员认为这应该归功于他们大脑杏仁核区脑灰质的变薄，科学家认为该区的脑灰质可致焦虑和压力。在先前研究中曾发现，长期冥想还会引起与自我意识有关的脑区的改变，但本次研究中，研究人员并未发现相关改变。他们认为，该区的结构改变应该和禅修的时间长短有关。所有以上在受训组所见的大脑结构变化，在对照组均未被发现。  
  
Britta H?lze博士来自德国吉森大学，是本次研究论文的第一作者。他惊叹道：“没想到人脑的可塑性如此之强，看来完全可以通过积极努力来改变我们的大脑结构，以增加幸福感，提升生活质量。其实。之前很多针对病人的研究均显示，冥想可有效帮助患者减轻症状感受，而本次研究结果正好向我们揭示了这些改善的根源所在。”  
  
迈阿密大学的神经系统科学家Amishi Jha博士一直从事类似研究，主要研究高压环境下正念训练对个体的影响。对于本次研究成果，他给予了高度评价，也提出了自己的看法：“本次研究结果向我们揭示了正念训练的作用机制，它证明，压力感受不但可以通过为期八周的正念禅修训练得以缓解，同时还将使受训者部分大脑结构发生变化，如大脑杏仁核区脑灰质的变薄等，这对科学家后续进行正念减压疗程以及压力相关的情绪紊乱等相关研究提供了无限契机和可能。”

NLP任职要求：

1.对自然语言处理有完整系统的认识：包括分词、新词发现、词性标注、命名实体识别、句法语法分析、依存分析、文本分类聚类、机器翻译、query改写、摘要抽取、情感分析、语义理解、知识图谱；

2.有自然语言处理实际项目经验, 能够有效的进行任务拆解和映射到具体的理论和方法上，精通模型的建立、调优和评估方案；

3.精通C/C++/Python语言中的一门或多门，能够对算法进行验证和实现，建造、使用过一种以上实用的语义网络，熟悉linux系统和脚本编程，在算法领域有扎实的理论基础和丰富的研发经验，对算法原理及应用有深入研究和独到见解。

岗位职责：

1. 负责自然语言处理相关的工作, 包括但不限于文本分类、命名实体识别、句法分析、信息抽取、知识库构建、主题词识别、自动摘要、智能问答、语义理解、对话生成、话题推荐等;    
2. 负责腾讯海量长短文本的情感分类，意见挖掘和舆情分析；  
3. 创建数亿用户级别的领域画像，并结合多种业务场景变现。

1.           熟悉意图分析、相似问题匹配、知识库挖掘、问答系统设计等NLP相关算法；

1、计算机相关专业硕士，包括：自然语言处理、计算语言学、机器翻译、信息检索等；

2、熟悉自然语言处理、机器学习、模式识别等常用算法，熟悉NLP领域当前热点和前沿技术，对NLP有完整的、系统的认识；

3、有相关项目经历，包含但不仅限于以下方向：智能客服、文本分类、信息抽取、知识图谱、机器学习、自动摘要等,有深度学习背景最佳；

4、熟练掌握C/C++编程语言；熟练使用Python，Shell等脚本语言；

5、良好的团队合作精神，较强的沟通能力；

6、优秀的分析和解决问题的能力，能够从海量数据中发现有价值的规律，对挑战性问题充满激情。

那么你就是我们想要的那一个！

在我们公司，你可能会得到以下工作内容：

1、基于机器学习, 并结合现有的自然语言处理技术，研发智能聊天机器人等的解决方案;

2、实现产品解决方案，进行效果调优;

3、发布相关产品，不断迭代产品效果。

岗位职责：

1. 参与问答和对话系统的算法设计与开发，包括但不限于自然语言处理，信息抽取，语义理解，知识图谱等NLP前沿技术；

2. 追踪和落地相关前沿技术和学术论文(深度学习等)，对模型进行优化和性能提升；

3. 参与和构建相关算法的API和工具包。

岗位要求：

1. 计算机或相关专业的硕士或者博士；

2. 熟悉自然语言处理、机器学习、深度学习等相关算法，对自然语言处理方向有较深和全面的认识；

3. 了解深度学习在自然语言处理方向的相关算法和框架，有Tensorflow, Theano, Keras项目实战经验的优先；

4. 有智能问答、语义理解、知识图谱等相关方向实战经验的优先；

5. 至少精通C、C++、Python、Java或者其他语言中的一种，有较强的编程能力；

6. 具有科学探索精神和团队协作能力。

# 35. CDH安装配置