# Spark+hadoop+mllib及相关概念与操作笔记

|  |  |
| --- | --- |
| 作者： | 刘炜 |
| 版本： | 0.1 |
| 时间： | 2016-07-18 |

# 调研相关注意事项

## 理解调研

调研的意义在于了解当前情况，挖掘潜在的问题，解决存在的疑问，并得到相应的方案。

## 调研流程

首先明确和梳理现有的疑问是什么，要通过调研解决什么问题，然后再去做调研，发现问题，再解决问题。

## 调研成果

最终需要得到结论与方案，以及详尽的论证理由，让别人信服。

## 书写格式

版本与作者以及时间可以以表格的形式，整齐明了。

结论简洁明了，论证理由详尽写在后面。

# Linux常见命令

## Locate 参数

Locate 可能查找不到最新添加的文件，因为它从数据库中找，有的时候没有更新就找不到。

locate指令和find找寻档案的功能类似，但locate是透过update程序将硬盘中的所有档案和目录资料先建立一个索引数据库，在执行loacte时直接找该索引，查询速度会较快，索引数据库一般是由操作系统管理，但也可以直接下达update强迫系统立即修改索引数据库。

## 查看系统版本

Uname -a：显示电脑及操作系统的相关信息

cat /proc/version：说明正在运行的内核版本

cat /etc/issue：显示的是发行版本的信息

lsb\_release -a：(适用于所有的linux，包括Redhat、SuSE、Debian等发行版，但是在debian下要安装lsb)

# Linux环境变量的设置

Linux中环境变量包括系统级和用户级，系统级的环境变量是每个登录到系统的用户都要读取的系统变量，而用户级的环境变量则是该用户使用系统时加载的环境变量。

## 系统级：

/etc/profile：用于交互的Login shell，交互式shell的环境变量。执行bashrc此文件为系统的每个用户设置环境信息,当第一个用户登录时,该文件被执行.并从/etc/profile.d目录的配置文件中搜集shell的设置.

/etc/bashrc：非交互式shell的环境变量。为每一个运行bash shell的用户执行此文件.当bash shell被打开时,该文件被读取。有些linux版本中的/etc目录下已经没有了bashrc文件。

/etc/environment:在登录时操作系统使用的第二个文件,系统在读取你自己的profile前,设置环境文件的环境变量。

## 用户级（这些文件处于家目录下）

~/.profile:每个用户都可使用该文件输入专用于自己使用的shell信息,当用户登录时,该文件仅仅执行一次!默认情况下,他设置一些环境变量,执行用户的.bashrc文件。这里是推荐放置个人设置的地方

~/.bashrc:该文件包含专用于你的bash shell的bash信息,当登录时以及每次打开新的shell时,该文件被读取。

~/.bash\_profile 、~./bash\_login：如果有则不读取.profile( ~/.bash\_profile or ~./bash\_login - If one of these file exist, bash executes it rather then "~/.profile" when it is started as a login shell. (Bash will prefer "~/.bash\_profile" to "~/.bash\_login"). However, these files won't influence a graphical session by default.)

~/.pam\_environment:用户级的环境变量设置文件，没有做测试，不知道管不管用。

想对所有的用户生效，那就需要设置系统级的环境变量。反之，需要修改用户级的文件（最好是修改.profile文件，理由上面已经讲了）。

## 生效

Source ~/.profile

# Hadoop基本命令

Hadoop文件系统命令：hadoop fs -linux命令 参数

Eg: [root@master ~]# hadoop fs -ls /

Eg: [root@master ~]# hadoop fs -cat /lwtest/lw.txt

其中fs是fileSystem，然后 - 后面接一般的linux指令就行，即hadoop文件系统的操作和linux文件系统的操作基本上一致，这样不用浪费时间去记很多命令了。

# 相关概念

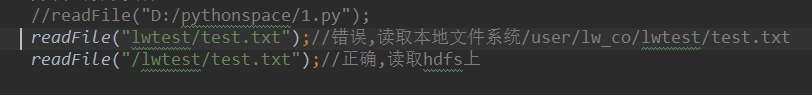
Hbase：分布式、面向列的开源数据库

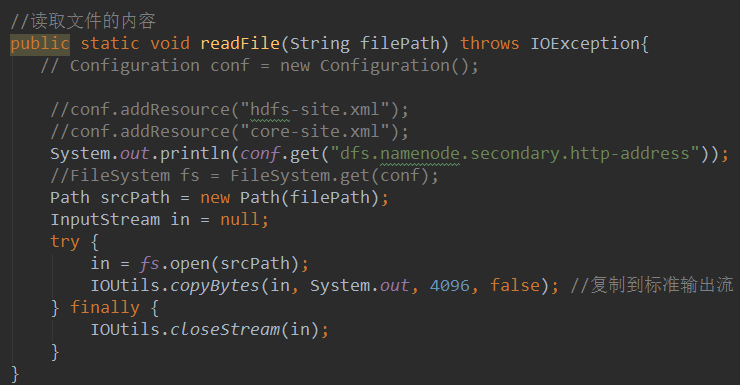
Yarn：Hadoop资源管理器，可以为上层应用提供统一的资源管理和调度，提高资源利用率及方便数据共享

# Hadoop，Hdfs连接操作

## 问题访问的是本地的文件系统而不是HDFS

原因：由于路径写错了





## 建立连接



默认读取Hdfs-site.xml,core-site.xml,mapred-site.xml等配置来建立连接

# Spark集群

## Spark web UI 控制端

<http://10.3.9.135:8180/>

可以在./sbin/start-master.sh下查看

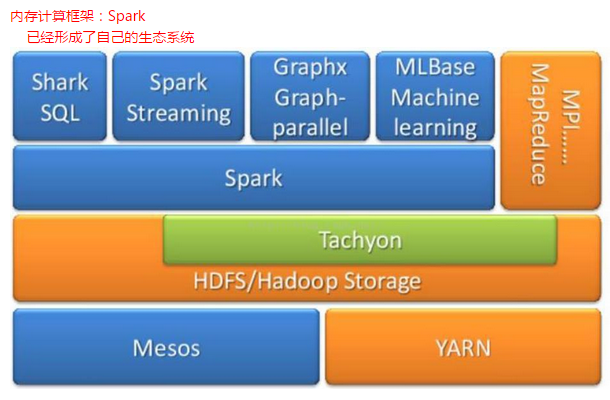
## 命令：clush -a -b jps 显示当前运行的服务进程

## Spark集群部署方式

参考：

<http://www.cnblogs.com/liuyifeng/p/5690627.html>

<http://www.aboutyun.com/forum.php?mod=viewthread&tid=7115>



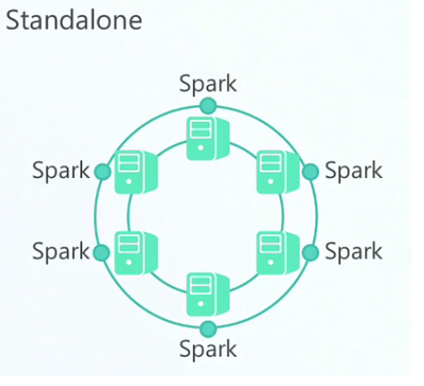
MR：离线计算框架

Storm：实时计算框架

Spark：内存计算框架

### Standalone

分别在集群的每台机器安装Spark,再启动相应的master和slave



### Spark On Mesos

### Spark On Yarn

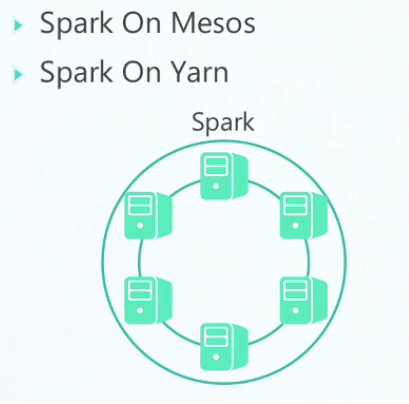
使用Spark客户端向yarn提交任务运行。

部署方式：

1.将spark部署包放到yarn集群某个节点上面

2.yarn客户端读取yarn集群配置文件，在此过程中，spark的master和slave节点不需要启动

Ps:Yarn(淘宝团队) Mesos(豆瓣)



## Hadoop web控制台端口

nameNodeIP:端口

Eg:

10.3.9.135:8088 Hadoop资源管理

10.3.9.135:50070 HDFS文件管理，节点信息

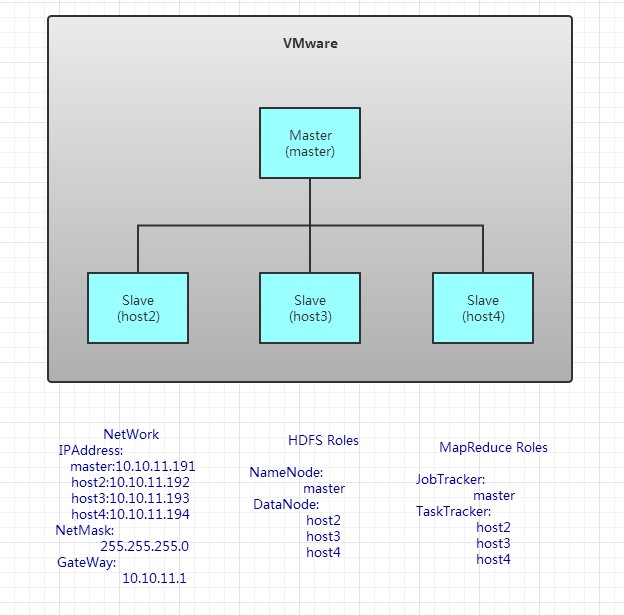
<http://10.3.9.135:50070/explorer.html#/> 可以查看文件

参见：

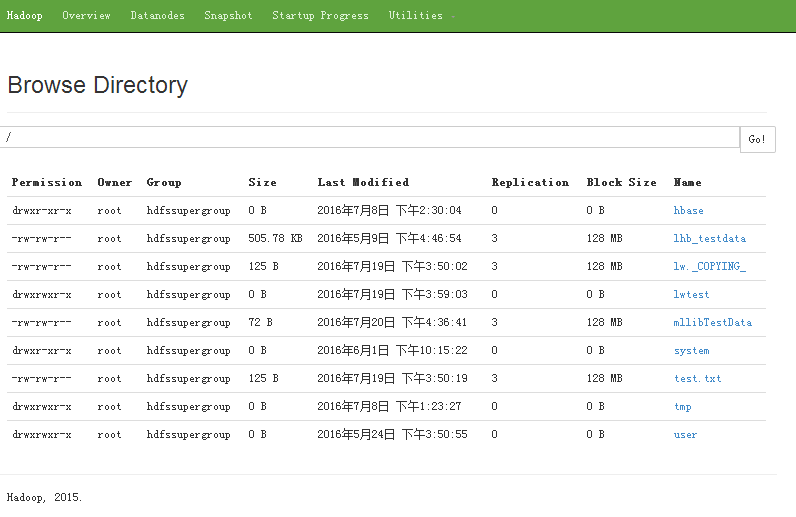
<http://www.cnblogs.com/laov/p/3433994.html>

<http://www.cnblogs.com/ggjucheng/archive/2012/04/17/2454590.html>

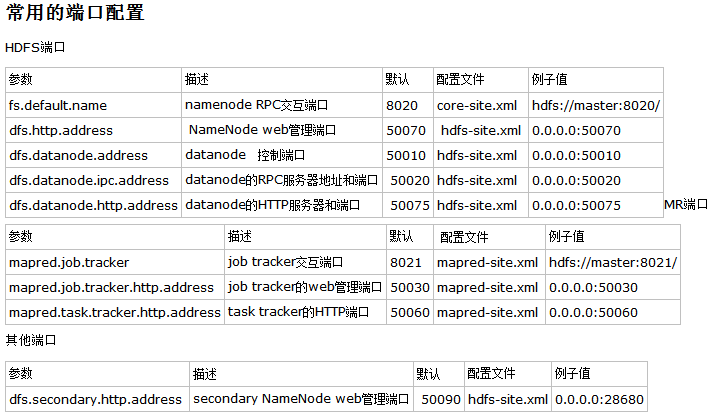
帮助理解：



图：Hadoop分布图



图；Hadoop HDFS文件浏览



图：Hadoop常用端口

## Spark例子运行

./bin/run-example SparkPi 10

./bin/spark-shell --master local[2]

The --master option specifies the master URL for a distributed cluster, or local to run locally with one thread, or local[N] to run locally with N threads. You should start by using local for testing. For a full list of options, run Spark shell with the --help option.

这--master选项指定master url 为一个分布式集群还是本地单线程的，或者local[N]本地N线程的。你应该开始使用本地测试，运行Spark shell --help选项。

## Mllib例子运行

./bin/run-example mllib.JavaKMeans mllibTestData/kmeans\_data.txt 3 100

mllib.JavaKMeans为相应的机器学习程序，是mllib文件夹下的JavaKMeans，后面三个是数据集 及相应参数。这里是K均值算法，后面3表示K的大小，100表示迭代次数。

## Job提交

参考：<http://spark.apache.org/docs/latest/submitting-applications.html>

# 本地8线程运行应用

# Run application locally on 8 cores

./bin/spark-submit \

--class org.apache.spark.examples.SparkPi \

--master local[8] \

/path/to/examples.jar \

100

**Eg:**./bin/spark-submit --class org.apache.spark.examples.SparkPi --master local[2] ./lib/spark-examples-1.6.1-hadoop2.7.1.jar 10

# 单集群

# Run on a Spark standalone cluster in client deploy mode

./bin/spark-submit \

--class org.apache.spark.examples.SparkPi \

--master spark://207.184.161.138:7077 \

--executor-memory 20G \

--total-executor-cores 100 \

/path/to/examples.jar \

1000

# Run on a Spark standalone cluster in cluster deploy mode with supervise

./bin/spark-submit \

--class org.apache.spark.examples.SparkPi \

--master spark://207.184.161.138:7077 \

--deploy-mode cluster \

--supervise \

--executor-memory 20G \

--total-executor-cores 100 \

/path/to/examples.jar \

1000

#在Spark on Yarn的集群上提交应用 （目前的集群）

#Run on a YARN cluster

export HADOOP\_CONF\_DIR=XXX

./bin/spark-submit \

--class org.apache.spark.examples.SparkPi \

--master yarn \

--deploy-mode cluster \ # can be client for client mode

--executor-memory 20G \

--num-executors 50 \

/path/to/examples.jar \

1000

**Eg:例子**

[root@master spark-1.6.1]#

./bin/spark-submit \

--class org.apache.spark.examples.SparkPi \

--master yarn \

--deploy-mode cluster \

./lib/spark-examples-1.6.1-hadoop2.7.1.jar \

10

其中SparkPi是通过蒙特卡罗思想就Pi的算法，10为其参数，大概指的是在单位为10的一个正方形里，正方形里包含一个半径为5的圆，然后生成随机数，均匀分布，数点落在圆里的数及全部的点数，通过面积的比例等于随机数分布的比例来求得Pi.

--deploy-mode cluster 在Yarn集群中需要指定。

**Eg:例子**

root@master spark-1.6.1]#

./bin/spark-submit \

--class org.apache.spark.examples.mllib.JavaLR \

--master yarn \

--deploy-mode cluster \

./lib/spark-examples-1.6.1-hadoop2.7.1.jar \

mllibTestData/lr-data/random.data 1 10

注意最后资源也应该是在集群hdfs上的文件

## 结果查看

参考：

<http://www.360doc.com/content/14/0810/12/7853380_400765442.shtml>

<http://www.iteblog.com/archives/1028>

System.out.println("Pi is roughly " + 4.0 \* count / n);

在yarn上运行，程序默认输出直接输出到logs里面。

Yarn logs -applicationId xxx 可以查看运行结束的Application日志

**Eg:**

yarn logs -applicationId application\_1467954918322\_0037

输出：

…省略…

LogType:stdout

Log Upload Time:星期一 七月 25 11:18:59 +0800 2016

LogLength:23

Log Contents:

Pi is roughly 3.141328

End of LogType:stdout

…省略…

当然可以将结果输出到文本里面，代码：

counts.saveAsTextFile("/home/wyp/result");

或者：

counts.saveAsHadoopFile("/home/wyp/result",

Text.class,

IntWritable.class,

TextOutputFormat.class);

将结果存储到HDFS上的/home/wyp/result文件夹里，第一种输出格式是（key,value）,第二种输出格式为key value.

我们可以根据自己的需要定义一个自己的输出格式，而且我们在输出的时候如果文件比较大，还可以指定输出文件的压缩方式。

## 日志

hdfs dfs -ls /tmp/logs/root/logs/tmp/logs/root/logs/application\_1467954918322\_0059

Hadoop下spark应用日志存在/tmp/logs/root/logs/tmp/logs/root/logs/下面。每个应用id对应一个目录，里面有在不同节点上跑的日志的相应记录。同Yarn logs -applicationId xxx 所看到的信息。

# Maven与eclipse

## eclipse常用快捷键

Alt+/补全

Ctrl+/增加注释

Ctrl+D删除当前行

## 什么是maven

Maven项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具。

Maven这个单词来自于意第绪语（犹太语），意为知识的积累，最初在Jakata Turbine项目中用来简化构建过程。当时有一些项目（有各自Ant build文件），仅有细微的差别，而JAR文件都由CVS来维护。于是希望有一种标准化的方式构建项目，一个清晰的方式定义项目的组成，一个容易的方式发布项目的信息，以及一种简单的方式在多个项目中共享JARs。

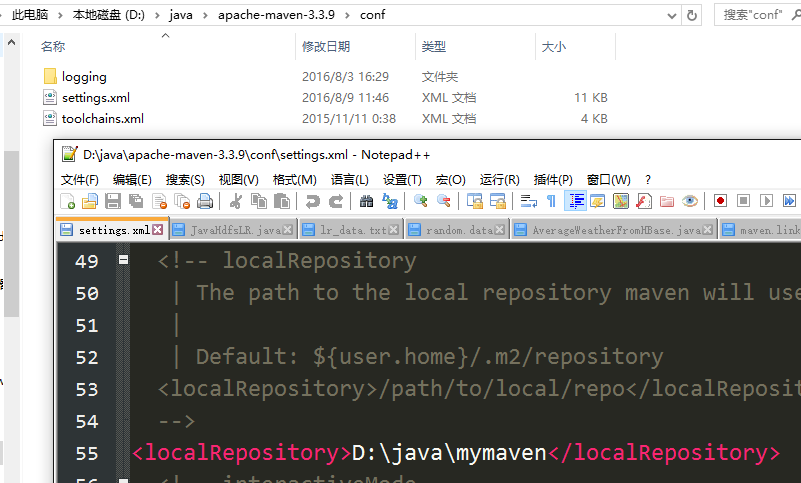
## 安装Maven

<http://dead-knight.iteye.com/blog/1841658>

* + 1. 安装maven，并设置环境变量（网上有的是）
    2. 查看配置是否完成。mvn -v 出现版本信息，表示配置成功



* + 1. 设置仓库位置



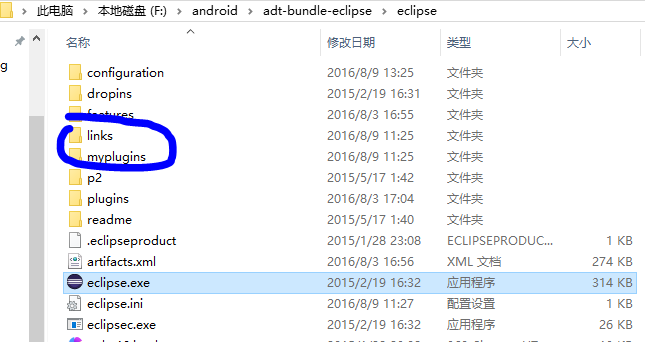
默认的maven仓库位置为：C:\Users\Administrator\.m2（其中Administrator为当前账号）

仓库是用来存Maven运行时的一此插件的。比说archetyper的插件。

## Eclicpse安装maven插件

Maven插件下载：<http://pan.baidu.com/s/1i5weBZZ>

解压后，把links、myplugins文件夹放到eclipse安装目录下，如下图所示：

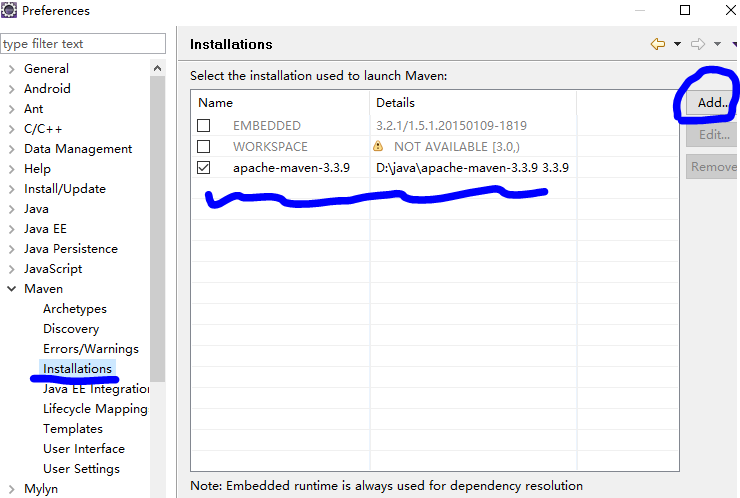


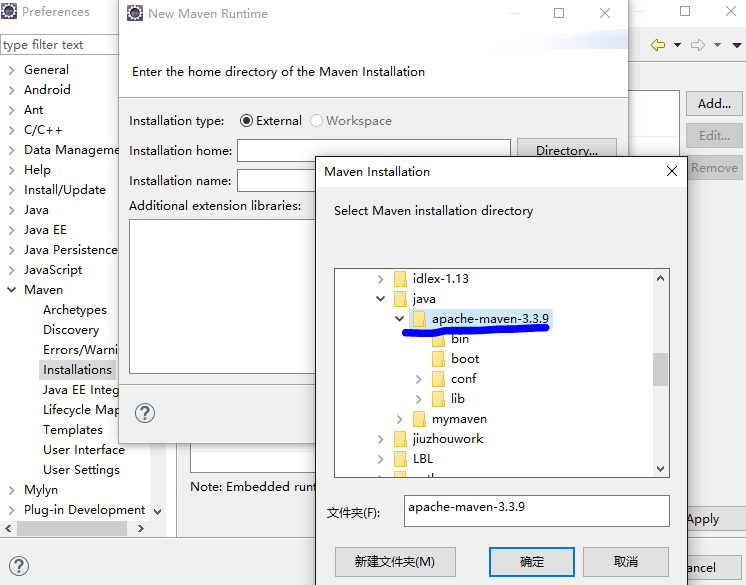
并且修改links下的maven.link文件。指向myplugins目录即可。我的配置为：

path=F:/android/adt-bundle-eclipse/eclipse/myplugins/maven

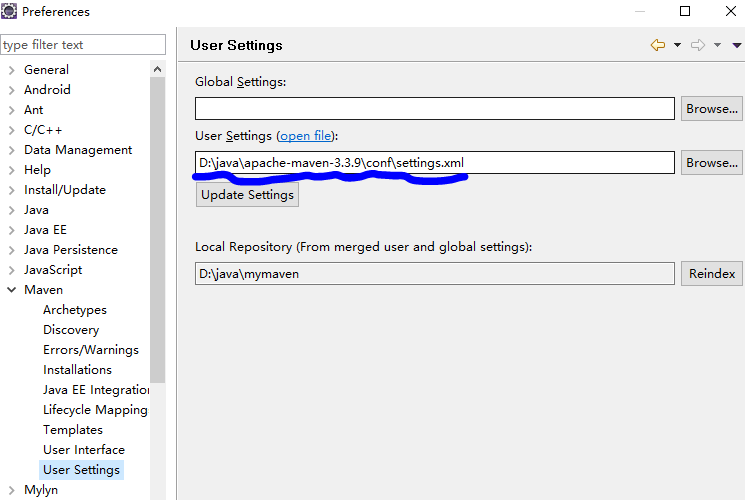
重启eclipse

选择window->preferences->maven，如下图所示：



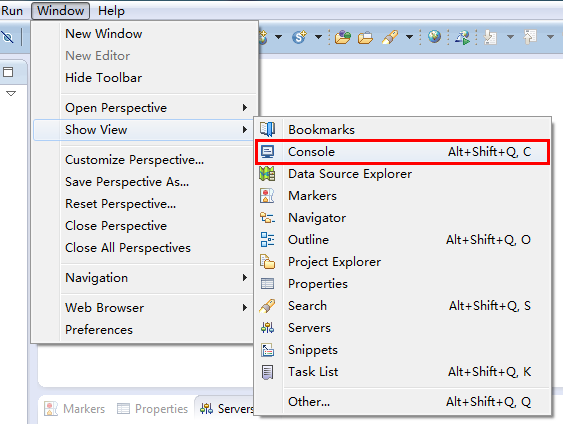


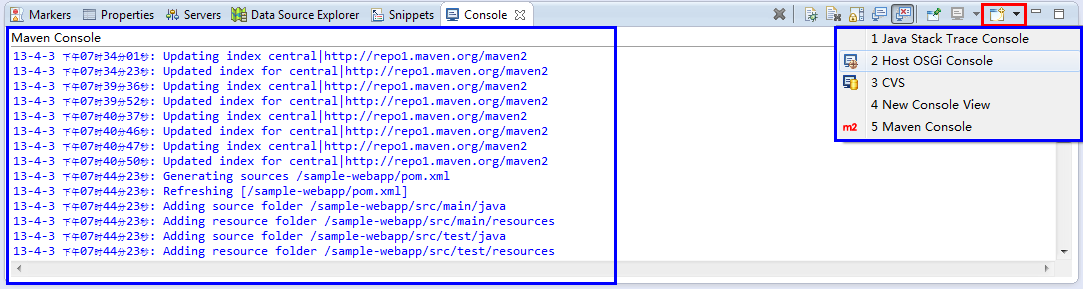
继续选择“User Settings”，配置maven的setting文件，如下图所示：



好，此时maven插件安装完毕。

## 查看maven的console





## 注意事项

Maven-archetype-quickstart与maven-archetype-profiles结构是一样的，即profiles即为quickstart.

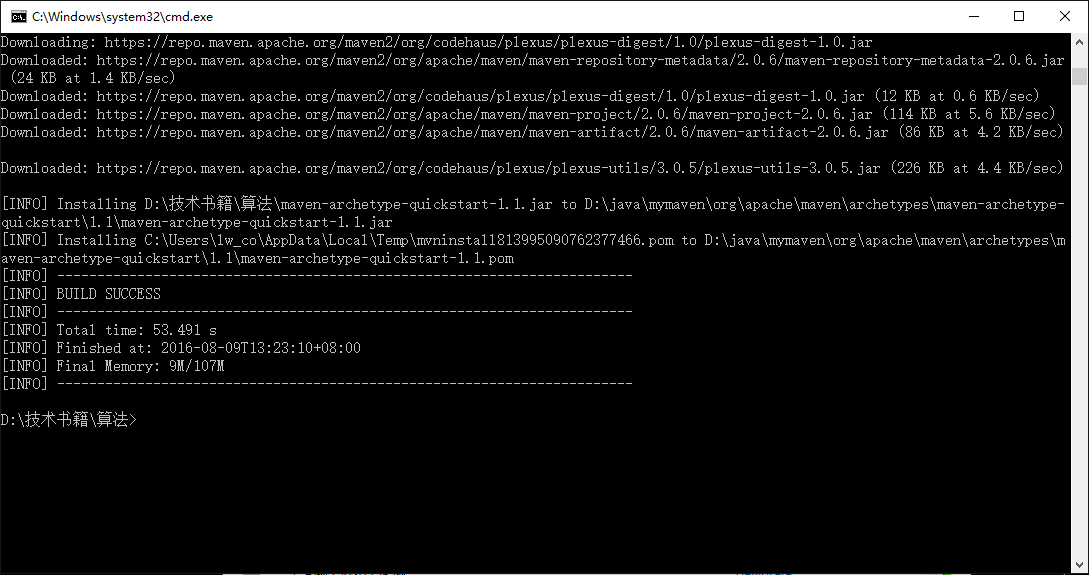
## 错误处理

### Could not resolve archetype org.apache.maven.archetypes:maven-archetype-quickstart

解决方式：

* + - 1. 从http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.1/maven-archetype-quickstart-1.1.jar下载最新版maven-archetype-quickstart-1.1.jar
      2. 命令行到下载目录下执行mvn install:install-file -DgroupId=org.apache.maven.archetypes -DartifactId=maven-archetype-quickstart -Dversion=1.1 -Dpackaging=jar -Dfile=maven-archetype-quickstart-1.1.jar

出现BUILD SUCCESS则表示安装成功了，就可以通过eclispe建quickstart结构的工程了。如下图所示：



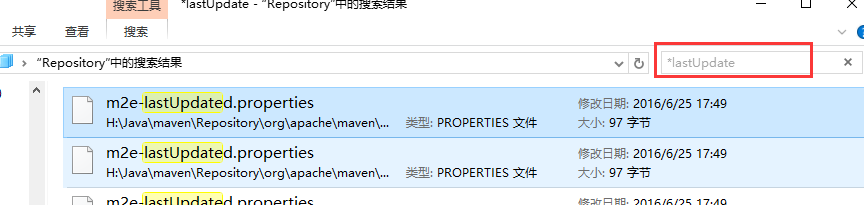
## Pom.xml红叉

<http://www.cnblogs.com/mymelody/p/5616685.html>

* + 1. dependency格式不对，或者jar包没有，使得对应的包没有下载下来
    2. eclipse没有自己下载，出现missing情况

解决方法：

* + - 1. 找到我们的本地maven仓库目录比如我的D:\java\mymaven
      2. 搜索出该目录下的\*lastUpdated.properties文件并删除，如下图所示，可以通过模糊搜索匹配出这样的文件



* + - 1. Maven 更新当前项目，maven就会继续下载缺失的依赖jar包，直至缺失jar包下载完成，上述问题就解决了。

## 远程仓库

参考：<http://www.cnblogs.com/dingyingsi/p/3856456.html>

仓库：

<http://repo2.maven.org/maven2/>

<http://uk.maven.org/maven2/>

[http://repository.apache.org](http://maven.apache.org)

<https://maven.java.net/content/groups/public/>

Other：

<https://repository.jboss.org/nexus/content/repositories/releases/>

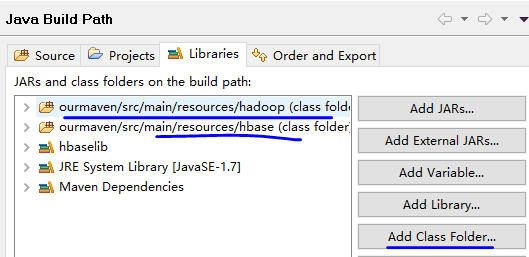
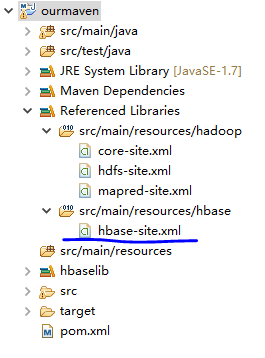
<http://repository.jboss.org/nexus/content/groups/public/>

# Eclipse与HBASE开发环境搭建

## 首先

将Hbase的配置文件hbase-site.xml下载到本地，建立工程后将其引入到工程里。这里将其放到main下的resouces文件夹下。

Ps：为什么要加入到build path，因为在编译时会通过path去查找和检查文件，然后在运行时候利用。

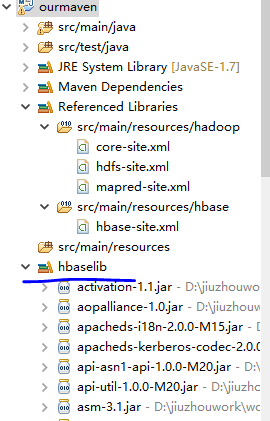


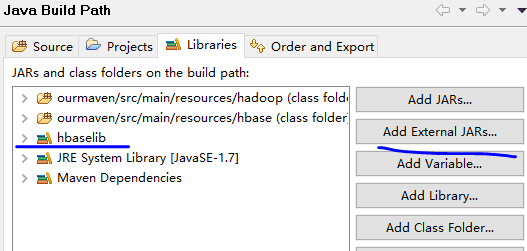
如果要用到MapReduce的话还应用把hadoop相应的配置文件放进来

如果要用到Spark的话当然要用到spark的配置文件

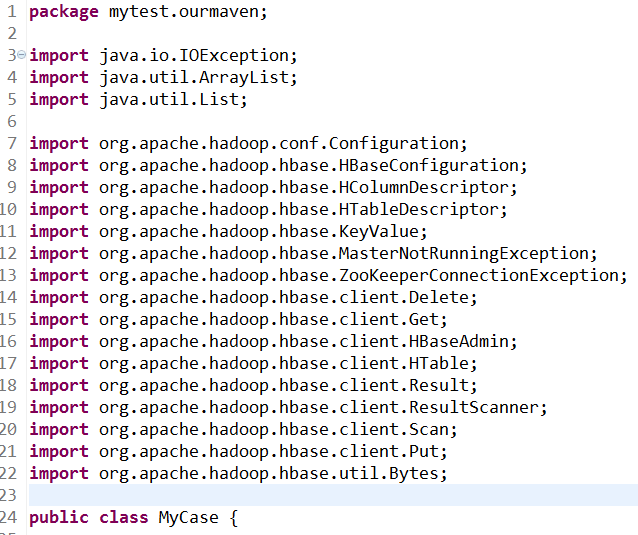
## 不用maven

将Hbase安装包下lib下的jar包拷贝下来，然后放到本地，将jar包加入到build path里面。





这样就可以了，import这些就可以正确的找到了。



Ps：在依赖包里没有的但却能正确的运用的，说明在java的本身包里面，或者在classpath里面，或者在jre/lib/，jre/lib/ext里面，这里面的包将被自动搜索。《Java核心技术第八版》第4.8类路径

## 用maven

* + 1. Maven工程
       1. GroupId：为组织名倒写如com.arvidlw
       2. Artifactid：这个一般为工程名，如bigdata
       3. Project：项目名
    2. Maven注意
       1. 打包时默认不加入依赖包的 <http://blog.csdn.net/jbgtwang/article/details/38226459>
       2. 所以想打出一个独立的可运行jar包的话直接mvn clean install package是不行的。需要略改动下pom文件，加入如下plugin
    3. 依赖包

<dependencies>

<dependency>

<groupId>org.apache.hbase</groupId>

<artifactId>hbase-client</artifactId>

<version>0.98.8-hadoop2</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

远程仓库比如说是：<http://uk.maven.org/maven2/>

则第一个意思是：在这个仓库下的org/apache/hbase/hbase-client/0.98.8-hadoop2/下的jar文件中去找hbase-client-0.98.8-hadoop2.jar

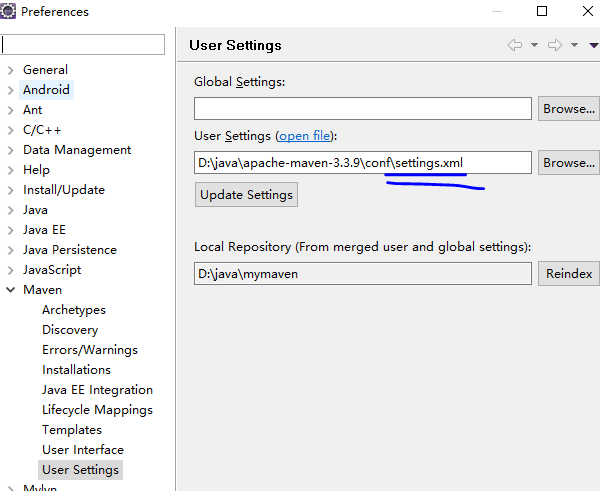
* + 1. Pom.xml出错出现红叉

要么格式不对，相应的文件在远程仓库里没有，要么就是没有把对应的文件下载下来，要么就是联网缓慢下载不下来。

解决办法（三种）：

<http://blog.csdn.net/dmlcq/article/details/51865887>

* + - 1. 右键项目，点击maven，然后update project，可以选择force update。
      2. 找到项目pom.xml的目录，然后用mvn clean与mvn install重新下载编译库。
      3. 换一个远程仓库，在user setting中找到对仓库的配置，然后，编辑设置文件，更换远程仓库地址。



<https://maven2-repository.java.net/>

Eg1:在settings中

<profiles>

<profile>

<id>maven-repository</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<repositories>

<repository>

<id>java.net-Public</id>

<name>Maven Java Net Snapshots and Releases</name> <url>https://maven.java.net/content/groups/public/</url>

</repository>

</repositories>

<pluginRepositories>

…

</pluginRepositories>

…

</profile>

</profiles>

Eg2:在pom.xml中

<repositories>

<repository>

<id>java.net-Public</id>

<name>Maven Java Net Snapshots and Releases</name>

<url>https://maven.java.net/content/groups/public/</url>

<updatePolicy>never</updatePolicy>

</repository>

</repositories>



当用新的仓库时，再次更新的时候就会从新的远程仓库上下载，上面的库类。

## Log4j放在哪

<http://blog.csdn.net/lifuxiangcaohui/article/details/11042375>

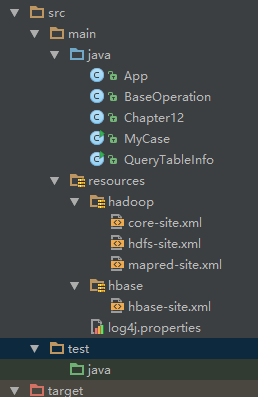
放在build\_path包含的路径里面，即它开始会去找的地方。

可以放在资源文件中，因为开始会去找到然后读

PropertyConfigurator.configure("../log4j.properties");

括号中为路径，只要能找到就ok.

在这时我放在resoures中



出现WARN Please initialize the log4j system properly

由于项目下没有对log4j进行配置

<http://javapub.iteye.com/blog/866664>

<http://www.cnblogs.com/jbelial/archive/2012/06/05/2536814.html>

解决：

建立log4j.properties文件，写入如下：

log4j.rootLogger=DEBUG, stdout

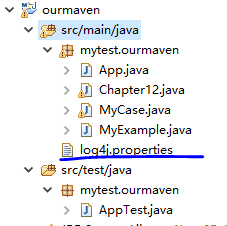
log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%c{1} - %m%n

log4j.logger.java.sql.PreparedStatement=DEBUG

这个文件这里放在src/main/java的根目录中

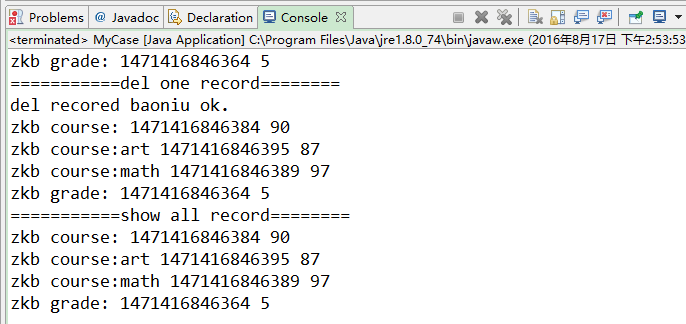


## 出现not locate executable null\bin\winutils.exe

<http://www.tuicool.com/articles/iABZJj>

程序需要找到winutils.exe，具体做什么的还不清楚，就是设置在windows运行的启动环境（不管他程序还是可以正常运行吧，2.X）。

Winutils.exe，是通过它来执行我们编写的程序来连接hbase吧



Java:

java.exe用于启动window console 控制台程序

javaw.exe用于启动 GUI程序

javaws.exe用于web程序。

jvm.dll就是java虚拟机规范在windows平台上的一种实现

<http://blog.csdn.net/topwqp/article/details/8595936>

作用：

1.读取hbase数据你首先需要一个client，jar包里不集成windows环境client的。

2.你需要一个可以运行的client——windows环境下就是exe(hbase默认是安装在linux下的)。

3.hbase是完全依赖hadoop的，hadoop为了满足windows用户提供了启动环境，在hadoopX.X/bin/下边的winutils.exe。

4.悲伤的是，从hadoop2.2开始，此文件莫名其妙地不打包了！任性！

5.此路径的引用是用HADOOP\_HOME变量或者是hadoop.home.dir配置都可以读取的

6.不需要完整安装hadoop2.2，只需要winutils.exe，并指定位置就好

7.System.setProperty("hadoop.home.dir", "X:/yyy");java设置系统变量，并在X:/yyy/bin/下放好winutils.exe就可以了，完全不需要安装，配置环境变量什么的 - -

解决方法：

* + 1. 下载winutils的windows版本

GitHub上，有人提供了winutils的windows的版本，项目地址是： https://github.com/ArvidLW/hadoop-common-2.2.0-bin ,直接下载此项目的zip包，下载后是文件名是hadoop-common-2.2.0-bin-master.zip,随便解压到一个目录

* + 1. 配置环境变量

增加用户变量HADOOP\_HOME，值是下载的zip包解压的目录，然后在系统变量path里增加$HADOOP\_HOME\bin 即可。

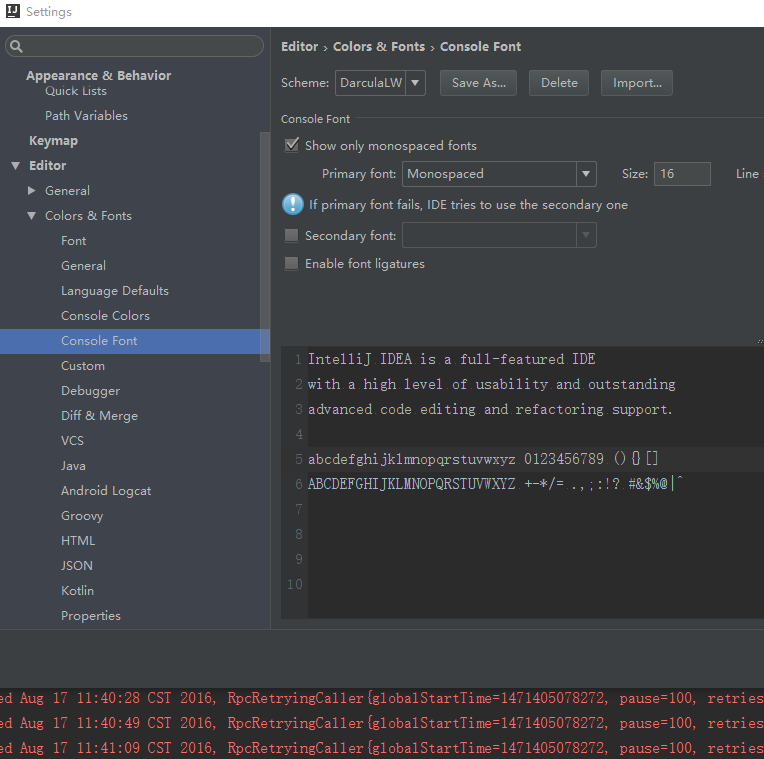
再次运行程序，正常执行。

# Intellij idea 与HBASE开发环境配置

## intellij设置字体

编辑器字体：settings->editor

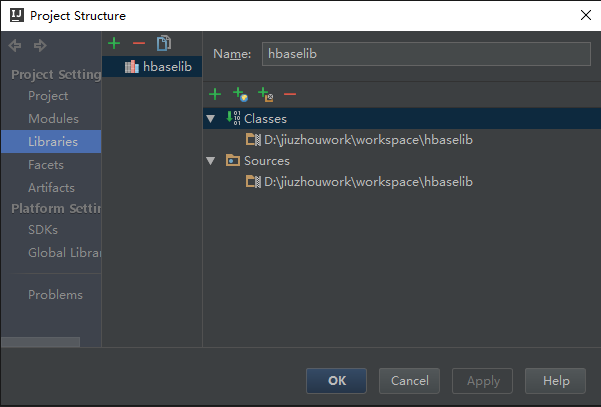
运行窗口字体：settings->editor->clors&Fonts->console Font



## 方法

与eclispe相似，intellj idea设计的更加人性化，要开发HBASE应用就在project Structure将hbaselib下的所有包加入到libraries，这里hbaselib是我把Hbase lib下的所有包都复制过来了然后建了个hbaselib文件夹放在下面。

加入libraries设置这个就相当于把依赖包路径加入到了编译命令中，这些依赖包在服务器上会在环境变量path中找，再在程序指定的依赖包中找，所以只要找到了就能愉快的运行了。

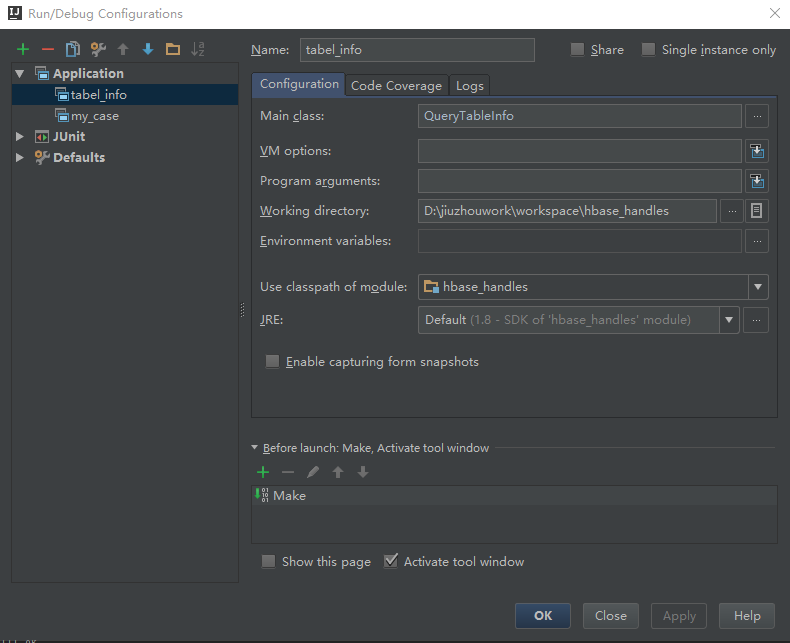


## 遇到错误

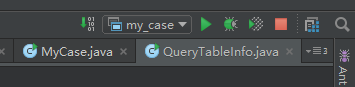
Error1:0 test class found in package ‘<default package>’

解决：Run->configuration,主要设置入口函数,不过eclipse不用设置啊，点到哪个运行哪个，好神奇，idea可以这样吗？

不过可以配置多个应用入口，点击+号，application。



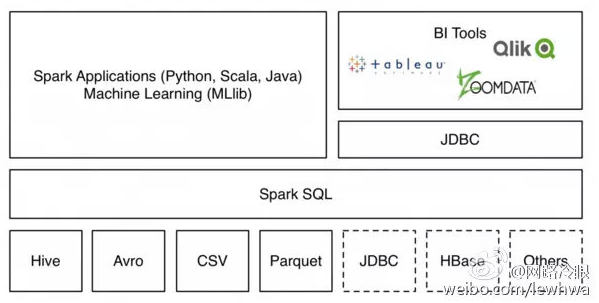
然后，我们在运行时可以选择相应的以不同class为入口的应用



# 数据接入

<http://blog.csdn.net/woshiwanxin102213/article/details/17584043>

<http://www.csdn.net/article/2015-02-13/2823955?ref=myread>



Hive：是建立在Hadoop上的数据仓库基础构架。所有Hive数据都存储在Hadoop兼容的文件系统中（例：Amazon S3、HDFS）中。Hive在加载数据过程中不会对数据进行任何修改，只是将数据移动到HDFS中Hive设定目录下，因此，Hive不支持对数据的改写和添加。

HBase：分布式面向列的开源数据库。不同于一般的关系数据库，它适用于非结构化存储数据库。

JDBC：关系型数据库方访问API，可以访问多种数据库

Spark SQL：摆脱了对Hive的依赖，兼容Hive，可以从RDD、PARQUET文件、JSON文件中获取数据。

<http://www.cnblogs.com/shishanyuan/p/4723604.html?utm_source=tuicool>

<http://www.csdn.net/article/2015-04-03/2824407>

<http://www.cnblogs.com/gaopeng527/p/4315808.html>



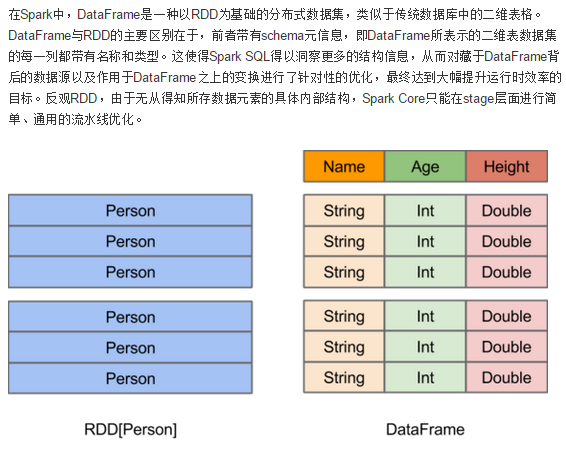
内部：

<http://developer.51cto.com/art/201603/507668.htm>

RDD：(Resilient Distributed Datasets)弹性分布式数据集

DataFrame：是一种以RDD为基础的分布式数据集，类似于传统数据库中的二维表格，被用于SQLContext相关操作。

DataSet：Dataset可以认为是DataFrame的一个特例，主要区别是Dataset每一个record存储的是一个强类型值而不是一个Row。



## Spark sql连接mysql

* + 1. 下载mysql-connector-java-5.1.39-bin.jar <http://www.itcast.cn/news/20151229/16012088060.shtml>

# clush

顾名思义：cluster shell, 用于集群上服务及状态的查询

clush -a -b -c /etc/yum.repos.d/\*

-a：所有节点all

-b：相同输出结果合并combine

-w：指定节点which

执行结构：

Clush+参数+linux命令+参数

意义：

将linux命令发送到集群的各个主机，得到相应信息返回到clush服务整理输出，各主机通过ssh相关联

Eg：clush -a -b -c jps

Jps：查看java进程，当前用户的java进程

查看本地集群信息，可以看到有这么多的应用用务。

HBase的服务应该是在namenode上为HRegionserver,HRegionServer主要负责响应用户I/O请求，向HDFS文件系统中读写数据，是HBase中最核心的模块。

在Master上应用为HMaster

HMaster的作用：

为Region server分配region

负责Region server的负载均衡

发现失效的Region server并重新分配其上的region

HDFS上的垃圾文件回收

处理schema更新请求

HRegionServer作用：

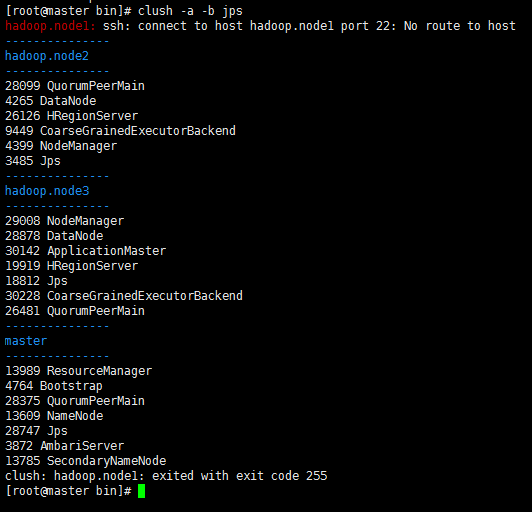
维护master分配给他的region，处理对这些region的io请求

负责切分正在运行过程中变的过大的region

可以看到，client访问hbase上的数据并不需要master参与（寻址访问zookeeper和region server，数据读写访问region server），master仅仅维护table和region的元数据信息（table的元数据信息保存在zookeeper上），负载很低。

HRegionServer存取一个子表时，会创建一个HRegion对象，然后对表的每个列族创建一个Store实例，每个Store都会有一个MemStore和0个或多个StoreFile与之对应，每个StoreFile都会对应一个HFile， HFile就是实际的存储文件。因此，一个HRegion有多少个列族就有多少个Store。

一个HRegionServer会有多个HRegion和一个HLog。



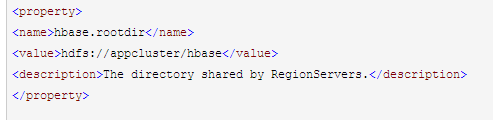
# HBase

## HBase web控制端

默认端口16010：<http://10.3.9.135:16010/master-status>

也可以自己设，eg：

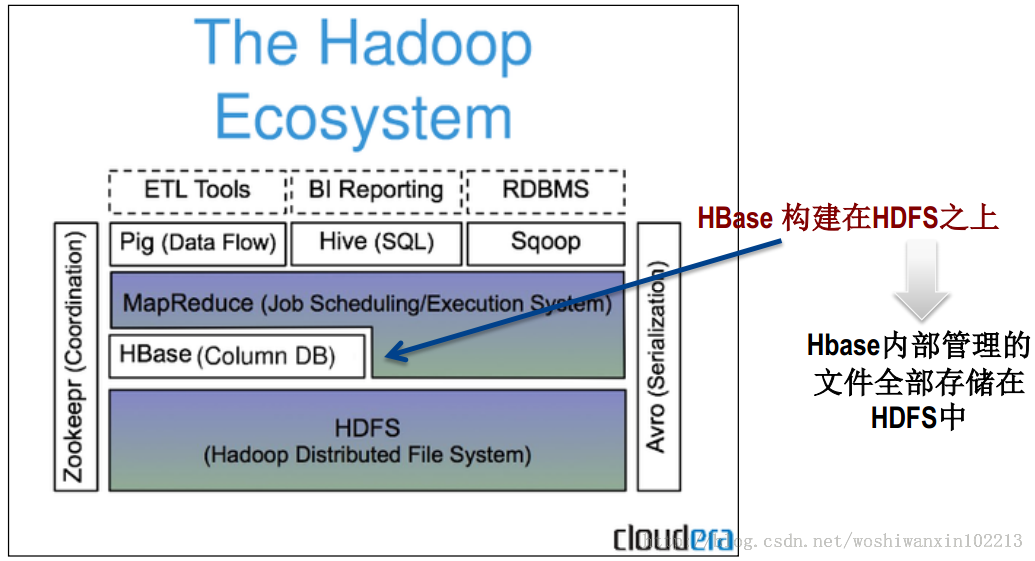
<http://www.cnblogs.com/captainlucky/p/4710642.html>



## 简介

<http://blog.csdn.net/woshiwanxin102213/article/details/17584043>

HBase是种大数据数据库，类似于google, bigTable。其使用和用其实数据库一样，可以用过其提供的端口进行操作，可以在eclipse中开发应用连接HBase并进行相关操作。



## Hbase表的特点

**大**：一个表可以有数十亿行，上百万列；

**无模式**：每行都有一个可排序的主键和任意多的列，列可以根据需要动态的增加，同一张表中不同的行可以有截然不同的列；

**面向列**：面向列（族）的存储和权限控制，列（族）独立检索；

**稀疏**：空（null）列并不占用存储空间，表可以设计的非常稀疏；

**数据多版本**：每个单元中的数据可以有多个版本，默认情况下版本号自动分配，是单元格插入时的时间戳；

**数据类型单一**：Hbase中的数据都是字符串，没有类型。

## Hbase基本概念

RowKey：是Byte array，是表中每条记录的“主键”，方便快速查找，Rowkey的设计非常重要。

Column Family：列族，拥有一个名称(string)，包含一个或者多个相关列

Column：属于某一个columnfamily，familyName:columnName，每条记录可动态添加

Version Number：类型为Long，默认值是系统时间戳，可由用户自定义

Value(Cell)：Byte array

## Hbase物理模型

每个column family存储在HDFS上的一个单独文件中，空值不会被保存。

Key 和 Version number在每个 column family中均有一份；

HBase 为每个值维护了多级索引，即：<key, column family, column name, timestamp>

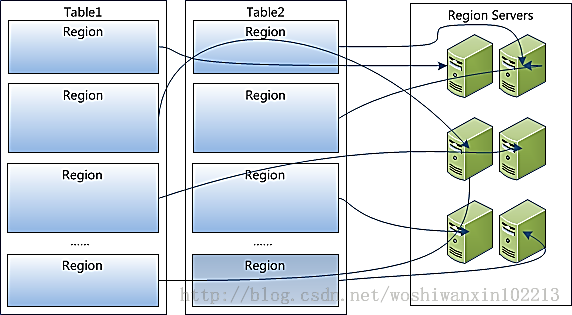
物理存储

1、Table中所有行都按照row key的字典序排列；

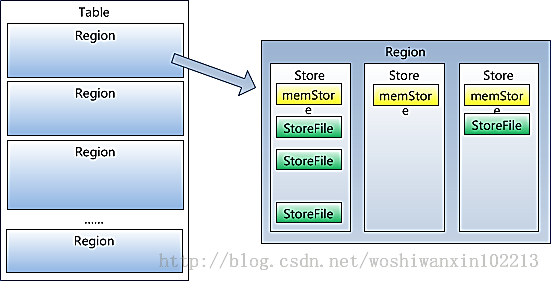
2、Table在行的方向上分割为多个Region；

3、Region按大小分割的，每个表开始只有一个region，随着数据增多，region不断增大，当增大到一个阀值的时候，region就会等分会两个新的region，之后会有越来越多的region；

4、Region是Hbase中分布式存储和负载均衡的最小单元，不同Region分布到不同RegionServer上。



1. Region虽然是分布式存储的最小单元，但并不是存储的最小单元。Region由一个或者多个Store组成，每个store保存一个columns family；每个Strore又由一个memStore和0至多个StoreFile组成，StoreFile包含HFile；memStore存储在内存中，StoreFile存储在HDFS上。



## Hbase shell

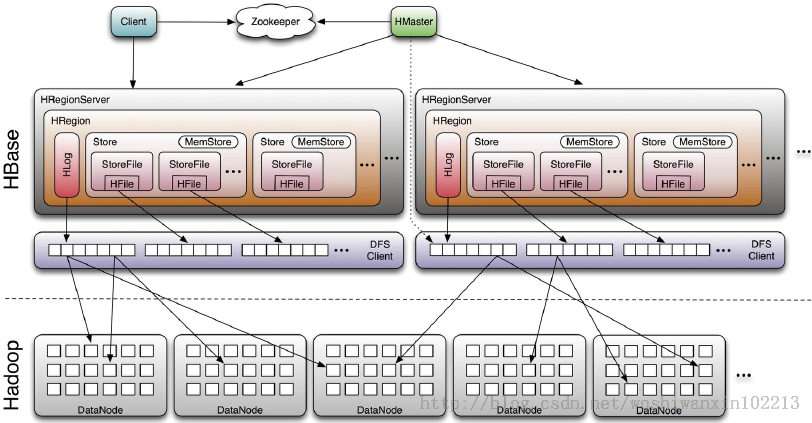
./hbase shell 进入 hbase shell.

<http://www.cnblogs.com/heyCoding/archive/2012/11/09/2762334.html>

scan ‘CityWeather’ , {VERSION => 10}

注意VERSION与VERSIONS的区别，一个是指定版本，一个是列出几个版本。

## HBase架构及基本组件



**Client**

包含访问HBase的接口，并维护cache来加快对HBase的访问，比如region的位置信息

**Master**

为Region server分配region

负责Region server的负载均衡

发现失效的Region server并重新分配其上的region

管理用户对table的增删改查操作

**Region Server**

Regionserver维护region，处理对这些region的IO请求

Regionserver负责切分在运行过程中变得过大的region

**Zookeeper作用**

通过选举，保证任何时候，**集群中只有一个master**，Master与RegionServers 启动时会向ZooKeeper注册

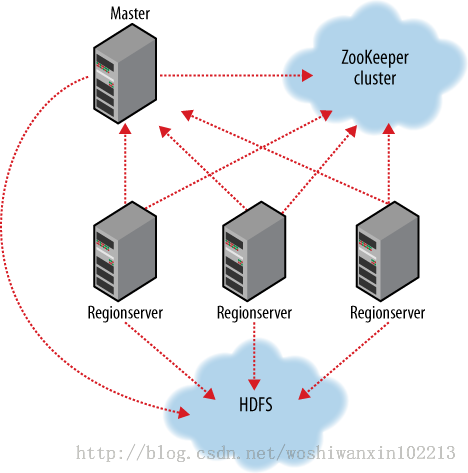
存贮所有Region的寻址入口

实时监控Region server的上线和下线信息。并实时通知给Master

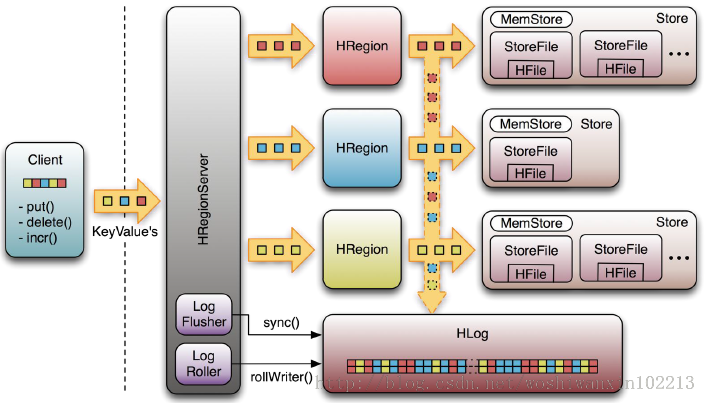
存储HBase的schema和table元数据

默认情况下，HBase 管理ZooKeeper 实例，比如， 启动或者停止ZooKeeper

Zookeeper的引入使得Master不再是单点故障



**Write-Ahead-Log（WAL）**

****

**该机制用于数据的容错和恢复：**

每个HRegionServer中都有一个HLog对象，HLog是一个实现Write Ahead Log的类，在每次用户操作写入MemStore的同时，也会写一份数据到HLog文件中（HLog文件格式见后续），HLog文件定期会滚动出新的，并删除旧的文件（已持久化到StoreFile中的数据）。当HRegionServer意外终止后，HMaster会通过Zookeeper感知到，HMaster首先会处理遗留的 HLog文件，将其中不同Region的Log数据进行拆分，分别放到相应region的目录下，然后再将失效的region重新分配，领取 到这些region的HRegionServer在Load Region的过程中，会发现有历史HLog需要处理，因此会Replay HLog中的数据到MemStore中，然后flush到StoreFiles，完成数据恢复

## HBase容错性

Master容错：Zookeeper重新选择一个新的Master

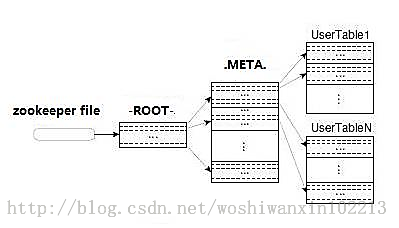
无Master过程中，数据读取仍照常进行；

无master过程中，region切分、负载均衡等无法进行；

RegionServer容错：定时向Zookeeper汇报心跳，如果一旦时间内未出现心跳，Master将该RegionServer上的Region重新分配到其他RegionServer上，失效服务器上“预写”日志由主服务器进行分割并派送给新的RegionServer

Zookeeper容错：Zookeeper是一个可靠地服务，一般配置3或5个Zookeeper实例

Region定位流程：



找RegionServer

ZooKeeper--> -ROOT-(单Region)--> .META.--> 用户表

-ROOT-

表包含.META.表所在的region列表，该表只会有一个Region；

Zookeeper中记录了-ROOT-表的location。

.META.

表包含所有的用户空间region列表，以及RegionServer的服务器地址。

## Hbase使用场景

storing large amounts of data(100s of TBs)

need high write throughput

need efficient random access(key lookups) within large data sets

need to scale gracefully with data

for structured and semi-structured data

don't need fullRDMS capabilities(cross row/cross table transaction, joins,etc.)

大数据量存储，大数据量高并发操作

需要对数据随机读写操作

读写访问均是非常简单的操作

Hbase与HDFS对比

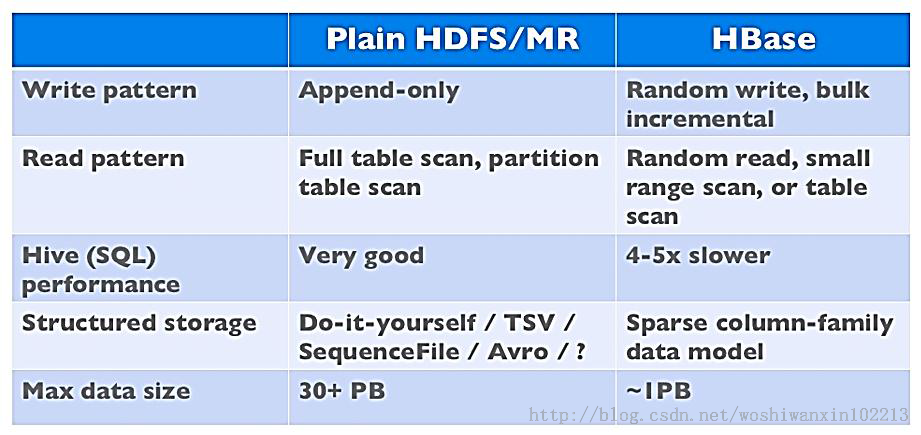
两者都具有良好的容错性和扩展性，都可以扩展到成百上千个节点；

HDFS适合批处理场景

不支持数据随机查找

不适合增量数据处理

不支持数据更新



## HBase高可用性配置

一个分布式运行的Hbase依赖一个zookeeper集群，所有的节点和客户端都必须能够访问zookeeper。默认的情况下Hbase会管理一个zookeep集群。这个集群会随着Hbase的启动而启动。当然，你也可以自己管理一个zookeeper集群，但需要配置Hbase。你需要修改HBASE\_MANAGES\_ZK 来切换。这个值默认是true的，作用是让Hbase启动的时候同时也启动zookeeper.

应该可以不用管HBASE\_MANAGES\_ZK ，按上面所说，这个为true就是为了当重启的时候也重启zookeeper。

<http://www.tuicool.com/articles/Af6vYb>

<http://www.cnblogs.com/captainlucky/p/4710642.html>

HMaster没有单点问题，HBase中可以启动多个HMaster，通过Zookeeper的Master Election机制保证总有一个Master运行。

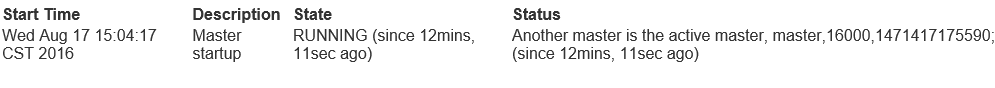
所以这里要配置HBase高可用的话，**只需要启动两个HMaster**，让Zookeeper自己去选择一个Master Acitve。

命令：hbase-daemon.sh start master

检测：通过kill现在active Hmaster，当执行程序时zookeeper就会调用另一个standby Hmaster 使它active.

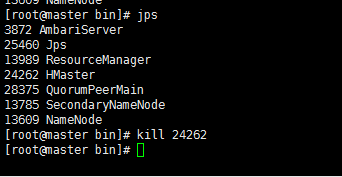
可以通过端口16010查看，每台主机节点都有

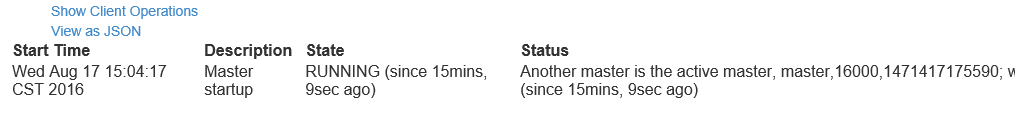
Eg:<http://10.3.9.231:16010/master-status>



表明这台机器上的HMaster是standby的。

当我kill 10.3.9.231的HMaster后





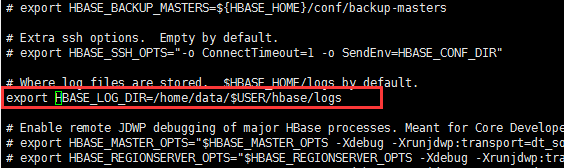
其变成的active HMaster

至于怎么在日志中看到，我还不清楚。

## HBase日志

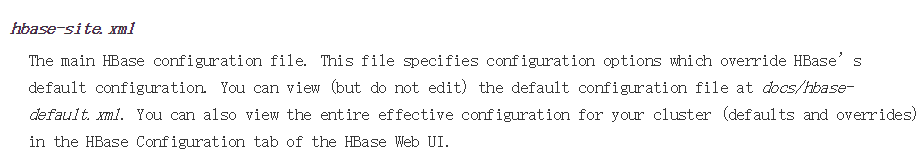
<http://hbase.apache.org/book.html#_configuration_files>

* + 1. 配置logs,在安装目录conf下的 hbase-env.sh



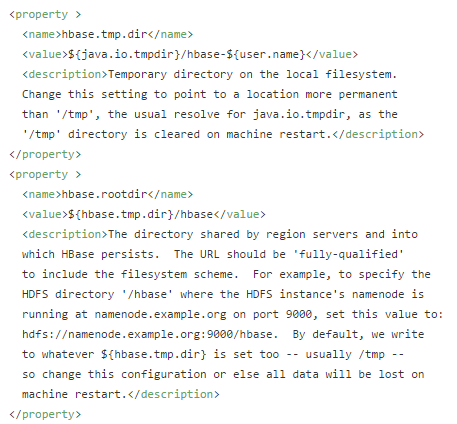
env里面设置的是程序启动前的一些配置，日志是程序没跑起来都需要的。

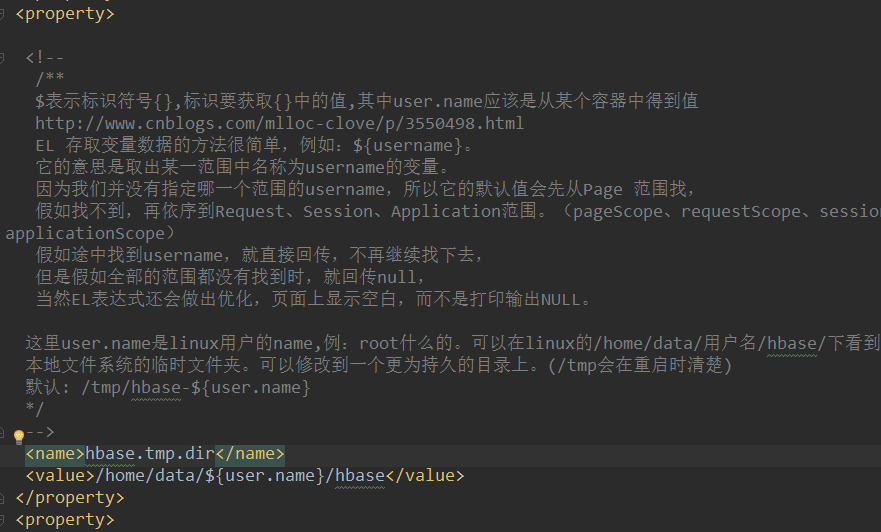
* + 1. 在HBase的安装目录下的conf的hbase-site.xml里面有配置，hbase.tmp.dir，hbase运行临时目录。

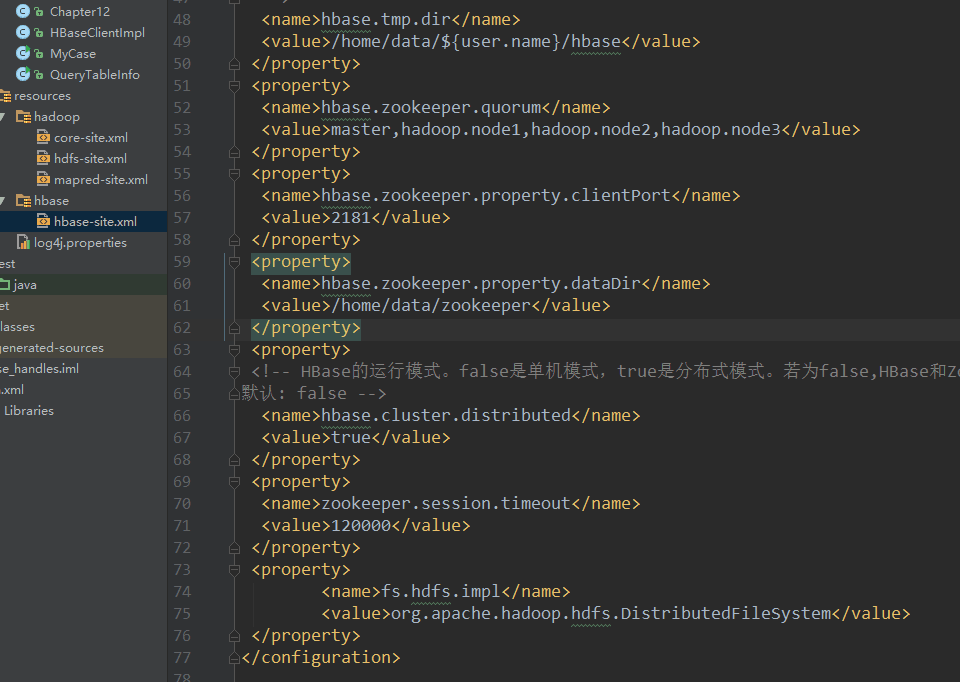


Hbase-default.xml

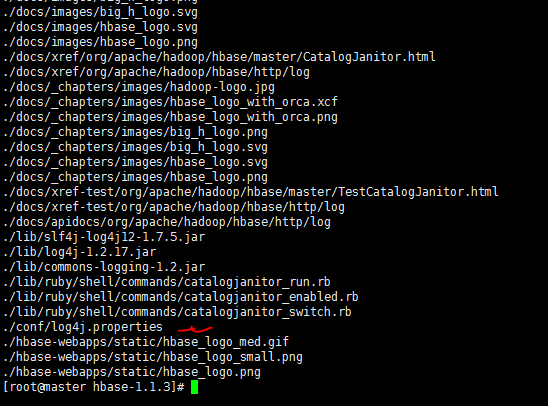
<https://github.com/apache/hbase/blob/master/hbase-common/src/main/resources/hbase-default.xml>





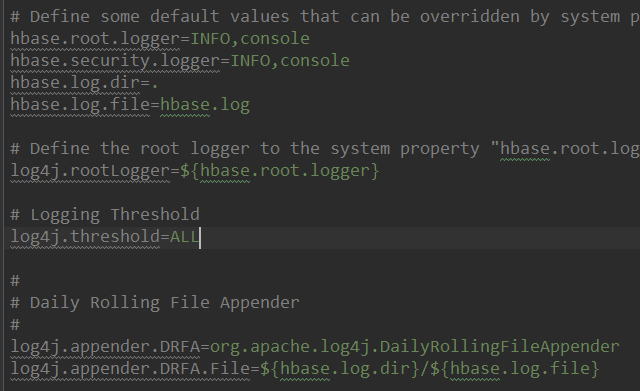


* + 1. 在HBase的安装文件夹下通过find ./ “\*log\*”找到

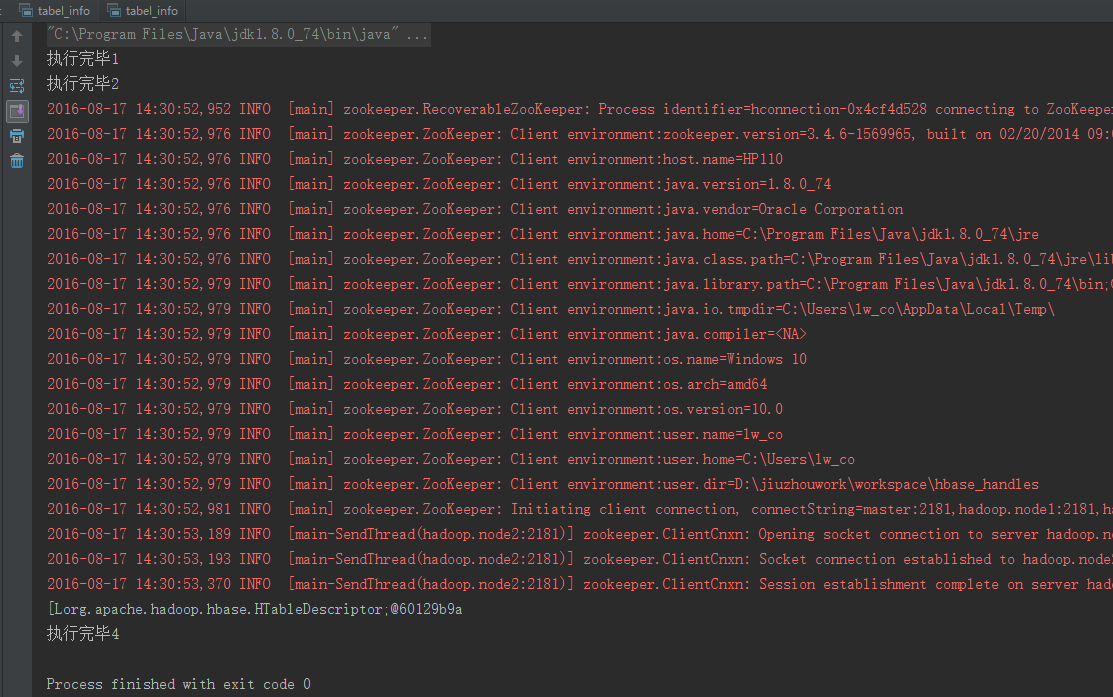


可以看到这里有log4j的配置，也就是我们在运行Hbase相关程序时，它会通过这个配置来打印与操作我们的信息。至于它是否保存在哪或者怎么设置还不太清楚。

这个log4j.properies 可以放在我们的项目的资源文件夹下，这样当我们运行程序时会根据配置打印相关内容，方便我们查看



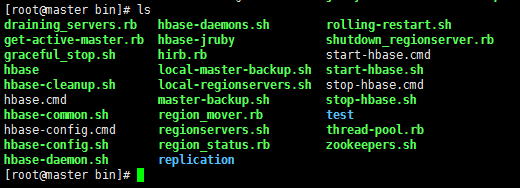
上面是部分配置信息，可以看到日志是通过INFO，控制台输出的



上面是程序运行时输出和Log输出

## ERROR

Bin文件下的命令



ERROR:org.apache.hadoop.hbase.PleaseHoldException

Resolution:Please stop hbase on your cluster first. And restart them with

certain sequences: first regionserver on all nodes, then hmaster.

运行stop-hbase.sh然后运行start-hbase.sh

## 编程

Hbase客户端API中，对HBASE的任何操作都需要首先创建HbaseConfiguration类的实例，为HBaseConfiguration类继承自Configuration类，而Configuration类属于Hadoop核心包中实现的类，该类的主要作用是提供对配置参数的访问途径。

* + 1. Java连接HBase

<http://my.oschina.net/u/160697/blog/516362>

Eg1：

private static HBaseConfiguration hbaseConfig=null;

static {

Configuration config=new Configuration();

config.set("hbase.zookeeper.quorum","192.168.1.98");

config.set("hbase.zookeeper.property.clientPort", "2181");

hbaseConfig = new HBaseConfiguration(config);

}

Eg2:

Configuration configuration = HBaseConfiguration.create();

configuration.set("hbase.zookeeper.property.clientPort", "2181"); //设置zookeeper client端口

configuration.set("hbase.zookeeper.quorum", "192.168.1.19"); // 设置zookeeper quorum

configuration.addResource("/usr/local/hbase-1.0.1.1/conf/hbase-site.xml"); //将hbase的配置加载

configuration.set(TableInputFormat.INPUT\_TABLE, "heartSocket");

在调用HBaseconfiguration.create()方法时，HBASE首先会在classpath下醒找hbase-site.xml文件，将里面的信息解析出来封装到Configuration对象中，如果hbase-site.xml文件不存在，则使用默认的hbase-core.xml文件。

除了将hbase-site.xml放到classpath下，开发人员还可通过config.set(name, value)方法来手工构建Configuration对象：

Configuration.set(String name, String value);

* + 1. 相关操作

<http://blog.csdn.net/javaman_chen/article/details/7220216>

创建表：HBaseAdmin admin = new HBaseAdmin(conf);

HTable封装表格对象，进行增删改查：

HTable table=new HTable(config,tableName);

Get,put,Delete,Scan操作，每个方法对应相应的操作对象

Eg:Delete delete=new Delete();

Table.delete(delete);

Hbase-site.xml参数详解

<http://greatwqs.iteye.com/blog/1837178>

* + 1. 表的结构

<http://blog.csdn.net/woshisap/article/details/43777135>

有多个列族：family

每个列族可以有一个或多个列成员，每个列成员

同一个列族存在同一个目录下，写操作锁行的，每一行是一个原子元素，都可以加锁。

映射：行键、行键+时间戳或行键+列

稀疏存储某些列可以空白

概念视图来看每个表格是由很多行组成的，但是在物理存储上面，它是按照列来保存的。

也就是说一个行关键字对应的信息由多的列族来各自保存。

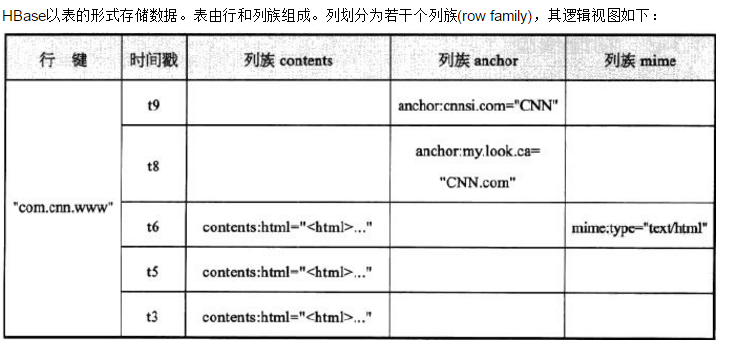
"<family>:<qualifier>"

Family为列族，比如有两列，列的字段和数量是提前指定好的不能改变，qualifier限定修饰符是可以随意加的。

定位某一单元格由行键，列族：限定符，时间戳唯一决定。

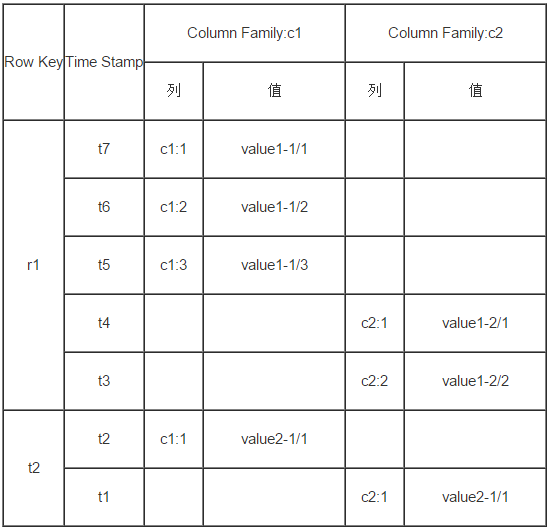
其实就是相当于mysql中的表格结构，row key就相当于主键，Family就是相当于某列，多个family就是多列，family是事先指定的，family就是列簇，因为它可以包含多个qualifier就是限定符。限定符可以任意增加，然后存的时候，考虑的到是稀疏的所以按列Family存放，即如果某个rowkey有多个列family属性，那么每个行的分开存，不存为空的值。

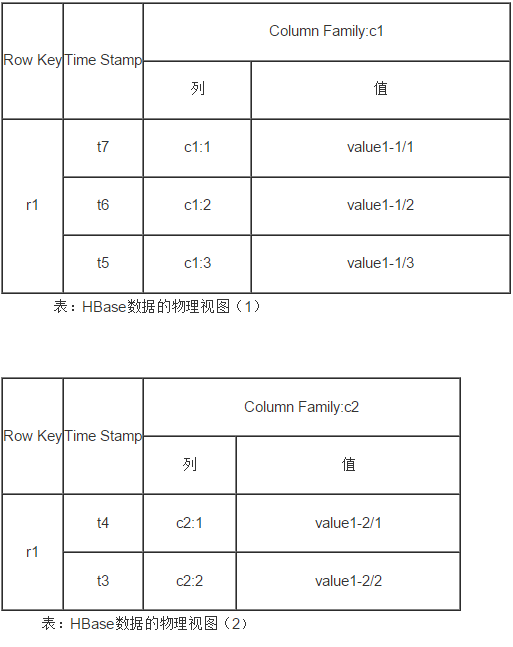
<http://blog.csdn.net/dbanote/article/details/8904003>



单元格由行键，列族：限定符，时间戳唯一决定。

Cell中数据是没有类型的，全部以字节码形式存储





# 算法运行及结果

## 运行样例：

Run-example:

./bin/run-example mllib.JavaKMeans mllibTestData/kmeans\_data.txt 2 100

mllibTestData为HDFS上的文件

如果类名不对，它会not Found，如果参数不够它会提示输入参数

Job:

./bin/spark-submit --class org.apache.spark.examples.mllib.JavaKMeans --master yarn --deploy-mode cluster ./lib/spark-examples-1.6.1-hadoop2.7.1.jar mllibTestData/kmeans\_data.txt 3 20

如果参数不够：exitCode:-1000

如果类名不对：exitCode:10

## linear regression（线性回归）

### JavaHdfsLR <file> <iters>

javaHdfsLR<文件><迭代次数>

本地：

./bin/spark-submit --class org.apache.spark.examples.JavaHdfsLR --master local ./lib/spark-examples-1.6.1-hadoop2.7.1.jar mllibTestData/lr\_data.txt 1

集群：

./bin/spark-submit --class org.apache.spark.examples.JavaHdfsLR --master yarn --deploy-mode cluster ./lib/spark-examples-1.6.1-hadoop2.7.1.jar mllibTestData/lr\_data.txt 1

结果：

LogType:stdout

Log Upload Time:星期一 七月 25 16:46:12 +0800 2016

LogLength:442

Log Contents:

Initial w: [0.4551273600657362, 0.36644694351969087, -0.38256108933468047, -0.4458430198517267, 0.33109790358914726, 0.8067445293443565, -0.2624341731773887, -0.44850386111659524, -0.07269284838169332, 0.5658035575800715]

On iteration 1

Final w: [246.25860765580322, 270.70869288178557, 371.354007739464, 357.4478152969409, 261.9494718512335, 210.01734831542458, 366.7061915626602, 381.34754796597383, 335.20416843810943, 240.24079035615807]

End of LogType:stdout

### JavaLR <input\_dir> <step\_size> <niters>

JavaLR<输入文件><步长大小><迭代次数>

./bin/spark-submit \

--class org.apache.spark.examples.mllib.JavaLR \

--master yarn \

--deploy-mode cluster \

./lib/spark-examples-1.6.1-hadoop2.7.1.jar \

mllibTestData/lr-data/random.data 1 10

结果：

LogType:stdout

Log Upload Time:星期一 七月 25 17:03:14 +0800 2016

LogLength:47

Log Contents:

Final w: [0.6262094779767464,0.535113798652265]End of LogType:stdout