[**hive2.1.1安装部署**](http://www.cnblogs.com/hmy-blog/p/6506417.html)

**一、Hive 运行模式**

与 [Hadoop](http://lib.csdn.net/base/hadoop) 类似，[Hive](http://lib.csdn.net/base/hive) 也有 3 种运行模式：

1. 内嵌模式

将元数据保存在本地内嵌的 Derby [数据库](http://lib.csdn.net/base/mysql)中，这是使用 [Hive](http://lib.csdn.net/base/hive) 最简单的方式。但是这种方式缺点也比较明显，因为一个内嵌的 Derby [数据库](http://lib.csdn.net/base/mysql)每次只能访问一个数据文件，这也就意味着它不支持多会话连接。

2. 本地模式

这种模式是将元数据保存在本地独立的数据库中（一般是 [MySQL](http://lib.csdn.net/base/mysql)），这用就可以支持多会话和多用户连接了。

3. 远程模式

此模式应用于 Hive 客户端较多的情况。把 [MySQL](http://lib.csdn.net/base/mysql) 数据库独立出来，将元数据保存在远端独立的 MySQL 服务中，避免了在每个客户端都安装 MySQL 服务从而造成冗余浪费的情况。

二、下载安装 Hive

http://hive.apache.org/downloads.html

tar -xzvf apache-hive-2.1.1-bin.tar.gz    ##解压

**三、配置系统环境变量**

**修改 /etc/profile 文件  vi /etc/profile 来修改（root用户操作）：**

1. 设置 Hive环境变量
2. # Hive environment
3. export HIVE\_HOME=/home/hadoop/cloud/apache-hive-2.1.1-bin
4. export PATH=$HIVE\_HOME/bin:$HIVE\_HOME/conf:$PATH
5. 使环境变量生效:
6. source /etc/profile

# Hive environment

export HIVE\_HOME=/usr/hive-2.1.1/apache-hive-2.1.1-bin

export HIVE\_CONF\_DIR=$HIVE\_HOME/conf

export CLASSPATH=$CLASSPATH:$HIVE\_HOME/lib

export PATH=$HIVE\_HOME/bin:$HIVE\_HOME/conf:$PATH

**四、内嵌模式**

（1）修改 Hive 配置文件

$HIVE\_HOME/conf 对应的是 Hive 的配置文件路径，类似于之前学习的[Hbase](http://lib.csdn.net/base/hbase), 该路径下的 hive-site.xml 是 Hive 工程的配置文件。默认情况下，该文件并不存在，我们需要拷贝它的模版来实现：

1. cp hive-default.xml.template hive-site.xml

hive-site.xml 的主要配置有：

* hive.metastore.warehouse.dir  
  该参数指定了 Hive 的数据存储目录，默认位置在 HDFS 上面的 /user/hive/warehouse 路径下。
* hive.exec.scratchdir  
  该参数指定了 Hive 的数据临时文件目录，默认位置为 HDFS 上面的 /tmp/hive 路径下。

**同时我们还要修改 Hive 目录下 /conf/hive-env.sh 文件（请根据自己的实际路径修改），该文件默认也不存在，同样是拷贝它的模版来修改：**

**cp hive-env.sh.template  hive-env.sh**

1. # Set HADOOP\_HOME to point to a specific hadoop install directory
2. HADOOP\_HOME=/home/hadoop/cloud/hadoop-2.7.3
4. # Hive Configuration Directory can be controlled by:
5. export HIVE\_CONF\_DIR=/home/hadoop/cloud/apache-hive-2.1.1-bin/conf
7. # Folder containing extra ibraries required for hive compilation/execution can be controlled by:
8. export HIVE\_AUX\_JARS\_PATH=/home/hadoop/cloud/apache-hive-2.1.1-bin/lib

JAVA\_HOME=/usr/java/jdk1.8.0\_131

HADOOP\_HOME=/usr/hadoop/hadoop-2.8.0

HIVE\_HOME=/usr/hive-2.1.1/apache-hive-2.1.1-bin

export HIVE\_CONF\_DIR=/usr/hive-2.1.1/apache-hive-2.1.1-bin/conf

export HIVE\_AUX\_JARS\_PATH=/usr/hive-2.1.1/apache-hive-2.1.1-bin/lib  
（2）创建必要目录  
  
  
前面我们看到 hive-site.xml 文件中有两个重要的路径，切换到 hadoop 用户下查看 HDFS 是否有这些路径：

1. hadoop fs -ls /

**没有发现上面提到的路径，因此我们需要自己新建这些目录，并且给它们赋予用户写（W）权限。**

1. $HADOOP\_HOME/bin/hadoop fs -mkdir -p /user/hive/warehouse
2. $HADOOP\_HOME/bin/hadoop fs -mkdir -p /tmp/hive/
3. hadoop fs -chmod 777 /user/hive/warehouse
4. hadoop fs -chmod 777 /tmp/hive

hadoop fs -mkdir -p /hive/warehouse

hadoop fs -mkdir -p /tmp/hive/

hadoop fs -chmod 777 /hive/warehouse

hadoop fs -chmod 777 /tmp/hive

**检查是否新建成功 hadoop fs -ls / 以及 hadoop fs -ls /user/hive/ ：**

（3）修改 io.tmpdir 路径

同时，要修改 hive-site.xml 中所有包含 ${system:j[ava](http://lib.csdn.net/base/javase).io.tmpdir} 字段的 value 即路径（vim下 / 表示搜索，后面跟你的关键词，比如搜索 hello，则为 /hello , 再回车即可），你可以自己新建一个目录来替换它，例如 /home/[Hadoop](http://lib.csdn.net/base/hadoop)/cloud/apache-hive-2.1.1-bin/iotmp

1. mkdir /home/hadoop/cloud/apache-hive-2.1.1-bin/iotmp
2. chmod 777 /home/hadoop/cloud/apache-hive-2.1.1-bin/iotmp
3. 把hive-site.xml 中所有包含 ${system:Java.io.tmpdir}替换成/home/hadoop/cloud/apache-hive-2.1.1-bin/iotmp

**全局替换命令 先按Esc键  再同时按shift+:把以下替换命令粘贴按回车即可全局替换**

1. %s#${system:java.io.tmpdir}#/home/hadoop/cloud/apache-hive-2.1.1-bin/iotmp#g

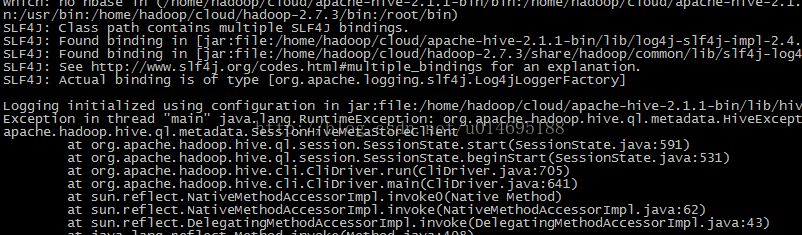
/usr/hive-2.1.1/iotmp

将hive-site.xml文件中的${system:java.io.tmpdir}替换为hive的临时目录，例如我替换为/opt/hive/tmp，该目录如果不存在则要自己手工创建，并且赋予读写权限。

将${system:user.name}都替换为root **（4）运行 Hive**

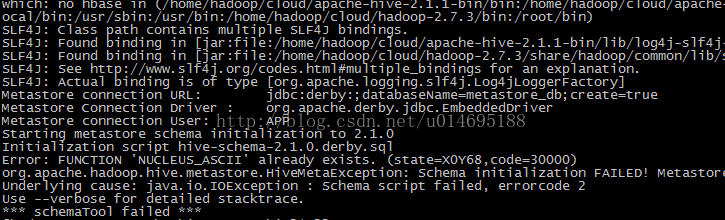
1. ./bin/hive

**报错**

****

解决办法：./bin/schematool -initSchema -dbType derby

**报错**

****

解决方法：删除/home/hadoop/cloud/apache-hive-2.1.1-bin目录下 rm -rf metastore\_db/ 在初始化：./bin/schematool -initSchema -dbType derby  
重新运行

./bin/hive

报错

http://img.blog.csdn.net/20170121180226141?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvdTAxNDY5NTE4OA==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

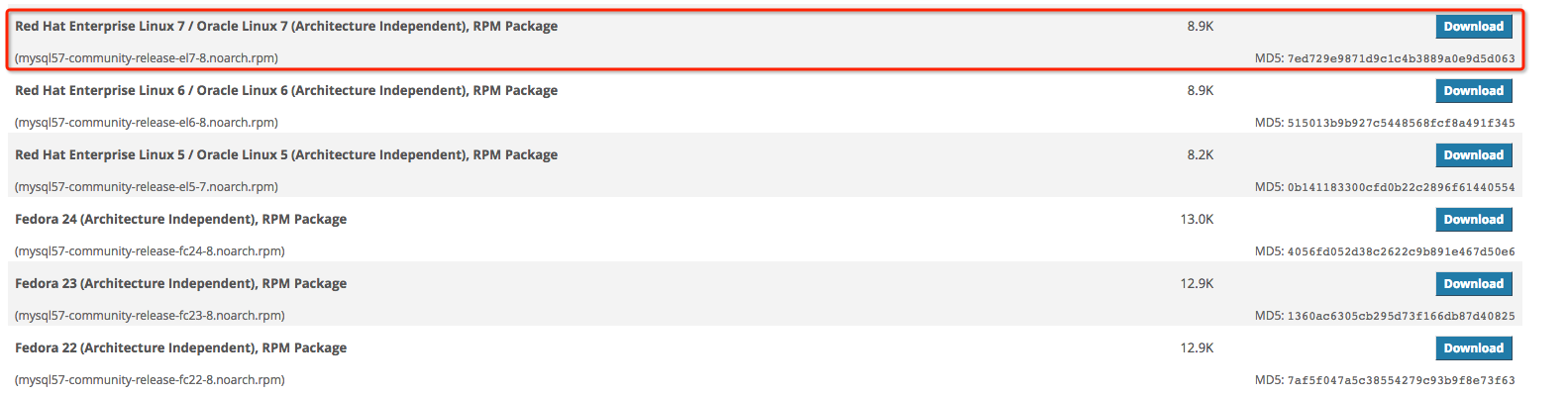
/tem/hive 没写的权限

Hive本身自带一个[数据库](http://lib.csdn.net/base/mysql)，但是有弊端，hive本身数据库，每次只允许一个用户登录

## CentOS7下安装MySQL5.7安装与配置（YUM）

安装环境：CentOS7 64位 MINI版，安装MySQL5.7

## 1、配置YUM源

在[**MySQL**](http://lib.csdn.net/base/14)官网中下载YUM源rpm安装包：<http://dev.mysql.com/downloads/repo/yum/>   


# 下载mysql源安装包

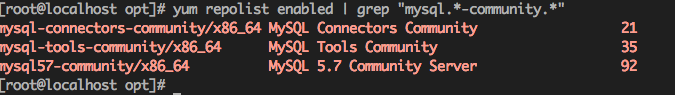
shell> wget http://dev.mysql.com/get/mysql57-community-release-el7-8.noarch.rpm

# 安装mysql源

shell> yum localinstall mysql57-community-release-el7-8.noarch.rpm

检查mysql源是否安装成功

shell> yum repolist enabled | grep "mysql.\*-community.\*"

   
看到上图所示表示安装成功

## 2、安装MySQL

shell> yum install mysql-community-server

## 3、启动MySQL服务

shell> systemctl start mysqld

查看MySQL的启动状态

shell> systemctl status mysqld

● mysqld.service - MySQL Server

Loaded: loaded (/usr/lib/systemd/system/mysqld.service; disabled; vendor preset: disabled)

Active: active (running) since 五 2016-06-24 04:37:37 CST; 35min ago

Main PID: 2888 (mysqld)

CGroup: /system.slice/mysqld.service

└─2888 /usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid

6月 24 04:37:36 localhost.localdomain systemd[1]: Starting MySQL Server...

6月 24 04:37:37 localhost.localdomain systemd[1]: Started MySQL Server.

## 4、开机启动

shell> systemctl enable mysqld

shell> systemctl daemon-reload

## 5、修改root默认密码

mysql安装完成之后，在/var/log/mysqld.log文件中给root生成了一个默认密码。通过下面的方式找到root默认密码，然后登录mysql进行修改：

shell> grep 'temporary password' /var/log/mysqld.log

http://www.centoscn.com/uploads/allimg/160626/1-160626010T0Z8.jpg

shell> mysql -uroot -p

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass4!';

或者

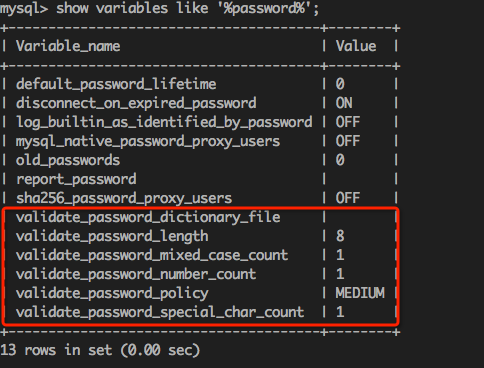
mysql> set password for 'root'@'localhost'=password('MyNewPass4!');

ALTER USER 'root'@'localhost' IDENTIFIED BY 'root123A!'

注意：mysql5.7默认安装了密码安全检查插件（validate\_password），默认密码检查策略要求密码必须包含：大小写字母、数字和特殊符号，并且长度不能少于8位。否则会提示ERROR 1819 (HY000): Your password does not satisfy the current policy requirements错误，如下图所示：   
http://www.centoscn.com/uploads/allimg/160626/1-160626010U5133.jpg

通过msyql环境变量可以查看密码策略的相关信息：

mysql> show variables like '%password%';

  
  validate\_password\_policy：密码策略，默认为MEDIUM策略  validate\_password\_dictionary\_file：密码策略文件，策略为STRONG才需要  validate\_password\_length：密码最少长度  validate\_password\_mixed\_case\_count：大小写字符长度，至少1个  validate\_password\_number\_count ：数字至少1个  validate\_password\_special\_char\_count：特殊字符至少1个  上述参数是默认策略MEDIUM的密码检查规则。

共有以下几种密码策略：

| **策略** | **检查规则** |
| --- | --- |
| 0 or LOW | Length |
| 1 or MEDIUM | Length; numeric, lowercase/uppercase, and special characters |
| 2 or STRONG | Length; numeric, lowercase/uppercase, and special characters; dictionary file |

MySQL官网密码策略详细说明：<http://dev.mysql.com/doc/refman/5.7/en/validate-password-options-variables.html#sysvar_validate_password_policy>

### 修改密码策略

在/etc/my.cnf文件添加validate\_password\_policy配置，指定密码策略

# 选择0（LOW），1（MEDIUM），2（STRONG）其中一种，选择2需要提供密码字典文件

validate\_password\_policy=0

如果不需要密码策略，添加my.cnf文件中添加如下配置禁用即可：

validate\_password = off

重新启动mysql服务使配置生效：

systemctl restart mysqld

## 6、添加远程登录用户

默认只允许root帐户在本地登录，如果要在其它机器上连接mysql，必须修改root允许远程连接，或者添加一个允许远程连接的帐户，为了安全起见，我添加一个新的帐户：

mysql> GRANT ALL PRIVILEGES ON \*.\* TO 'yangxin'@'%' IDENTIFIED BY 'Yangxin0917!' WITH GRANT OPTION;

## 7、配置默认编码为utf8

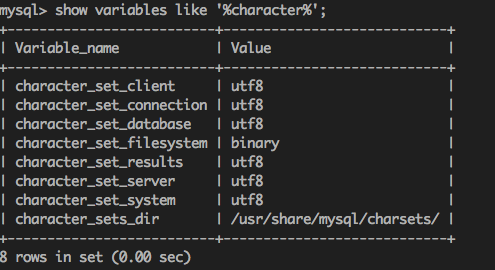
修改/etc/my.cnf配置文件，在[mysqld]下添加编码配置，如下所示：

[mysqld]

character\_set\_server=utf8

init\_connect='SET NAMES utf8'

重新启动mysql服务，查看[**数据库**](http://lib.csdn.net/base/14)默认编码如下所示：



**默认配置文件路径：**  配置文件：/etc/my.cnf  日志文件：/var/log//var/log/mysqld.log  服务启动脚本：/usr/lib/systemd/system/mysqld.service  socket文件：/var/run/mysqld/mysqld.pid

## 设置mysql关联hive

### 修改配置文件

### 创建hive-site.xml文件   
在hive/conf/目录下创建hive-site.xml文件

**搜索javax.jdo.option.ConnectionURL，**将该name对应的value修改为[**MySQL**](http://lib.csdn.net/base/14)的地址，例如我修改后是：

     <name>javax.jdo.option.ConnectionURL</name>

     <value>jdbc:mysql://192.168.27.138:3306/hive?createDatabaseIfNotExist=true</value>

**搜索javax.jdo.option.ConnectionDriverName**，将该name对应的value修改为MySQL驱动类路径，例如我的修改后是：

  <property>

        <name>javax.jdo.option.ConnectionDriverName</name>

        <value>com.mysql.jdbc.Driver</value>

 </property>

**搜索javax.jdo.option.ConnectionUserName**，将对应的value修改为MySQL[**数据库**](http://lib.csdn.net/base/14)登录名：

    <name>javax.jdo.option.ConnectionUserName</name>

    <value>root</value>

**搜索javax.jdo.option.ConnectionPassword，**将对应的value修改为MySQL数据库的登录密码：

     <name>javax.jdo.option.ConnectionPassword</name>

     <value>cj</value>

**搜索hive.metastore.schema.verification，**将对应的value修改为false：

    <name>hive.metastore.schema.verification</name>

     <value>false</value>

1. <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2. <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3. <configuration>
4. <property>
5. <name>javax.jdo.option.ConnectionURL</name>
6. <value>jdbc:mysql://192.168.169.134:3306/hive?createDatabaseIfNotExist=true</value>
7. </property>
8. <property>
9. <name>javax.jdo.option.ConnectionDriverName</name>
10. <value>com.mysql.jdbc.Driver</value>
11. </property>
12. <property>
13. <name>javax.jdo.option.ConnectionUserName</name>
14. <value>root</value>
15. </property>
16. <property>
17. <name>javax.jdo.option.ConnectionPassword</name>
18. <value>123456</value>
19. </property>
20. <property>
21. <name>hive.metastore.schema.verification</name>
22. <value>false</value>
23. <description>
24. Enforce metastore schema version consistency.
25. True: Verify that version information stored in metastore matches with one from Hive jars.  Also disable automatic
26. schema migration attempt. Users are required to manully migrate schema after Hive upgrade which ensures
27. proper metastore schema migration. (Default)
28. False: Warn if the version information stored in metastore doesn't match with one from in Hive jars.
29. </description>
30. </property>
31. </configuration>

报错：Caused by: MetaException(message:Version information not found in metastore. )

解决：hive-site.xml加入

1. <property>
2. <name>hive.metastore.schema.verification</name>
3. <value>false</value>
4. <description>
5. Enforce metastore schema version consistency.
6. True: Verify that version information stored in metastore matches with one from Hive jars.  Also disable automatic
7. schema migration attempt. Users are required to manully migrate schema after Hive upgrade which ensures
8. proper metastore schema migration. (Default)
9. False: Warn if the version information stored in metastore doesn't match with one from in Hive jars.
10. </description>
11. </property>

报错：缺少mysql jar包

解决：将其（如mysql-connector-[Java](http://lib.csdn.net/base/javase)-5.1.15-bin.jar）拷贝到$HIVE\_HOME/lib下即可。

报错：

1. Exception in thread "main" java.lang.RuntimeException: Hive metastore database is not initialized.
2. Please use schematool (e.g. ./schematool -initSchema -dbType ...) to create the schema. If needed,
3. don't forget to include the option to auto-create the underlying database in your JDBC connection string (e.g. ?createDatabaseIfNotExist=true for mysql)

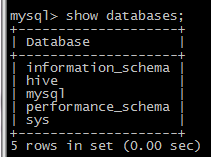
解决：

1. #数据库的初始化。
2. bin/schematool -initSchema -dbType mysql

启动：

1. bin/hive

启动后mysql 多了hive 数据库



[测试](http://lib.csdn.net/base/softwaretest)

### 创建数据库

create database db\_hive\_test;

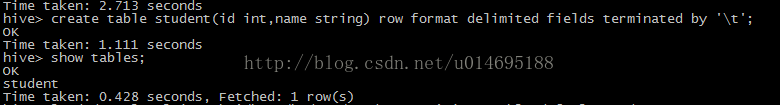
http://img.blog.csdn.net/20170122145641806?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvdTAxNDY5NTE4OA==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

创建测试表

use db\_hive\_test;

create table student(id int,name string) row format delimited fields terminated by '\t';

http://img.blog.csdn.net/20170122151518519?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvdTAxNDY5NTE4OA==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center



#### 加载数据到表中

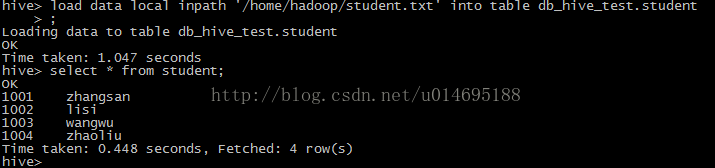
新建student.txt 文件写入数据(id，name 按tab键分隔)

vi student.txt

1. 1001    zhangsan
2. 1002    lisi
3. 1003    wangwu
4. 1004    zhaoli

load data local inpath '/home/hadoop/student.txt' into table  db\_hive\_test.student

 load data local inpath ' /usr/hive-2.1.1/student.txt' into table  db\_hive\_test.student



##### 查询表信息

select \* from student;

查看表的详细信息

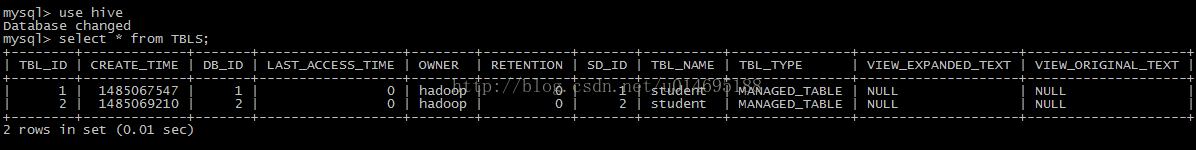
desc formatted student;

通过ui页面查看创建的数据位置

http://192.168.137.130:50070/explorer.html#/user/hive/warehouse/db\_hive\_test.db



通过Mysql查看创建的表

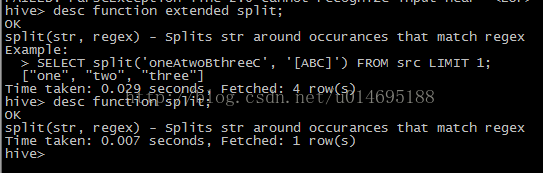


查看hive的函数

show functions;

查看函数详细信息

desc function sum;   
desc function extended



标签: [ubuntu、hive](http://www.cnblogs.com/hmy-blog/tag/ubuntu%E3%80%81hive/)

# [Hive基本命令整理](http://blog.csdn.net/xiaoping8411/article/details/7605039)

创建表：  
hive> CREATE TABLE pokes (foo INT, bar STRING);   
        Creates a table called pokes with two columns, the first being an integer and the other a string

创建一个新表，结构与其他一样  
hive> create table new\_table like records;

创建分区表：  
hive> create table logs(ts bigint,line string) partitioned by (dt String,country String);

加载分区表数据：  
hive> load data local inpath '/home/[**Hadoop**](http://lib.csdn.net/base/20)/input/hive/partitions/file1' into table logs partition (dt='2001-01-01',country='GB');

展示表中有多少分区：  
hive> show partitions logs;

展示所有表：  
hive> SHOW TABLES;  
        lists all the tables  
hive> SHOW TABLES '.\*s';

lists all the table that end with 's'. The pattern matching follows [**Java**](http://lib.csdn.net/base/17) regular  
expressions. Check out this link for documentation <http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html>

显示表的结构信息  
hive> DESCRIBE invites;  
        shows the list of columns

更新表的名称：  
hive> ALTER TABLE source RENAME TO target;

添加新一列  
hive> ALTER TABLE invites ADD COLUMNS (new\_col2 INT COMMENT 'a comment');  
   
删除表：  
hive> DROP TABLE records;  
删除表中数据，但要保持表的结构定义  
hive> dfs -rmr /user/hive/warehouse/records;

从本地文件加载数据：  
hive> LOAD DATA LOCAL INPATH '/home/hadoop/input/ncdc/micro-tab/sample.txt' OVERWRITE INTO TABLE records;

显示所有函数：  
hive> show functions;

查看函数用法：  
hive> describe function substr;

查看数组、map、结构  
hive> select col1[0],col2['b'],col3.c from complex;

内连接：  
hive> SELECT sales.\*, things.\* FROM sales JOIN things ON (sales.id = things.id);

查看hive为某个查询使用多少个MapReduce作业  
hive> Explain SELECT sales.\*, things.\* FROM sales JOIN things ON (sales.id = things.id);

外连接：  
hive> SELECT sales.\*, things.\* FROM sales LEFT OUTER JOIN things ON (sales.id = things.id);  
hive> SELECT sales.\*, things.\* FROM sales RIGHT OUTER JOIN things ON (sales.id = things.id);  
hive> SELECT sales.\*, things.\* FROM sales FULL OUTER JOIN things ON (sales.id = things.id);

in查询：Hive不支持，但可以使用LEFT SEMI JOIN  
hive> SELECT \* FROM things LEFT SEMI JOIN sales ON (sales.id = things.id);

Map连接：Hive可以把较小的表放入每个Mapper的内存来执行连接操作  
hive> SELECT /\*+ MAPJOIN(things) \*/ sales.\*, things.\* FROM sales JOIN things ON (sales.id = things.id);

INSERT OVERWRITE TABLE ..SELECT：新表预先存在  
hive> FROM records2  
    > INSERT OVERWRITE TABLE stations\_by\_year SELECT year, COUNT(DISTINCT station) GROUP BY year   
    > INSERT OVERWRITE TABLE records\_by\_year SELECT year, COUNT(1) GROUP BY year  
    > INSERT OVERWRITE TABLE good\_records\_by\_year SELECT year, COUNT(1) WHERE temperature != 9999 AND (quality = 0 OR quality = 1 OR quality = 4 OR quality = 5 OR quality = 9) GROUP BY year;

CREATE TABLE ... AS SELECT：新表表预先不存在  
hive>CREATE TABLE target AS SELECT col1,col2 FROM source;

创建视图：  
hive> CREATE VIEW valid\_records AS SELECT \* FROM records2 WHERE temperature !=9999;

查看视图详细信息：  
hive> DESCRIBE EXTENDED valid\_records;

// hive常用命令整理---coco

1. 开启行转列功能之后:  
set hive.cli.print.header=true; // 打印列名  
set hive.cli.print.row.to.vertical=true; // 开启行转列功能, 前提必须开启打印列名功能  
set hive.cli.print.row.to.vertical.num=1; // 设置每行显示的列数  
  
  
2.使用过程中出错采用：  
hive -hiveconf hive.root.logger=DEBUG,console   //重启调试。  
  
  
3. hive的三种启动方式区别：  
  1，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  
  2，hive  web界面的启动方式，hive –service hwi    
     用于通过浏览器来访问hive，感觉没多大用途  
  3，hive  远程服务 (端口号10000) 启动方式，nohup hive –service hiveserver  &   
     用java等程序实现通过jdbc等驱动的访问hive就用这种起动方式了，这个是程序员最需要的方式了  
 启动hive service ：$HIVE\_HOME/bin/hive --service hiveserver 10001 >/dev/null 2>/dev/null &  
   
4. hive插入的2中方式：  
基本的插入语法：  
INSERT OVERWRITE TABLE tablename [PARTITON(partcol1=val1,partclo2=val2)]select\_statement FROM from\_statement  
insert overwrite table test\_insert select \* from test\_table;  
对多个表进行插入操作：  
FROM fromstatte  
INSERT OVERWRITE TABLE tablename1 [PARTITON(partcol1=val1,partclo2=val2)]select\_statement1  
INSERT OVERWRITE TABLE tablename2 [PARTITON(partcol1=val1,partclo2=val2)]select\_statement2  
  
  
5.添加metastore启动脚本bin/hive-metastore.sh  
#!/bin/sh  
nohup ./hive --service metastore >> metastore.log 2>&1 &  
echo $! > hive-metastore.pid  
  
  
添加hive server启动脚本bin/hive-server.sh  
nohup ./hive --service hiveserver >> hiveserver.log 2>&1 &  
echo $! > hive-server.pid  
  
  
启动metastore和hive server  
./hive-metastore.sh  
./hive-server.sh

# [Linux安装Hbase(CentOS7+Hbase1.2.5+Hadoop2.8.0)](http://blog.csdn.net/pucao_cug/article/details/72229223)

[1下载和解压缩](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634457)

[1.1下载](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634458)

[1.2解压缩](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634459)

[2 配置](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634460)

[2.1环境变量配置](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634461)

[2.2 conf目录下的配置](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634462)

[2.2.1修改hbase-env.sh文件](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634463)

[2.2.2修改配置文件hbase-site.xml](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634464)

[2.2.3修改regionservers文件](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634465)

[3启动和测试](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634466)

[3.1启动](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634467)

[3.2测试](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634468)

[3.2.1用浏览器访问Hbase状态信息](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634469)

[3.2.2启动hbase的命令行](file:///D:\111\hbase%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Hbase(CentOS7+Hbase1.2.5)-v1.0.0.htm#_Toc482634470)

**关键词：Linux   CentOS**[**Hadoop**](http://lib.csdn.net/base/20)**Hbase  Zookeeper**[**Java**](http://lib.csdn.net/base/17)

**版本号：CentOS7   Hadoop2.8.0     Hbase1.2.5   Zookeeper3.4.10   JDK1.8**

**说明：Hbase可以在Hadoop集群的某些机器(一台也可以，例如选择namenode节点来安装)上安装，也可以全部机器上都安装。Hbase是依赖Hadoop和Zookeeper的，所以在安装Hbase之前，请先安装Hadoop和Zookeeper，Zookeeper可以是单机的也可以是集群，Hadoop的安装请参考博文：**

<http://blog.csdn.net/pucao_cug/article/details/71698903>

      Zookeeper的安装请参考：

<http://blog.csdn.net/pucao_cug/article/details/72228973>

# 1  下载和解压缩

## 1.1  下载

        可以直接在CentOS上用wget下载，也可以在本地下载之后再用工具上载到CetnOS上。

目前稳定的版本是1.2.5，所以这里也是安装1.2.5。

       下载地址：

<http://mirror.bit.edu.cn/apache/hbase/stable/>

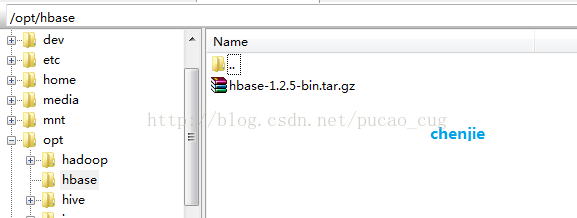
      如图：



## 1.2  解压缩

      使用root账号登录，在opt命令下新建一个没hbase的目录，把压缩包放到该目录下

**如图：**



      进入到该目录内然：

**cd     /opt/hbase**

       执行解压缩命令：

**tar   -xvf    hbase-1.2.5-bin.tar.gz**

# 2  配置

## 2.1  环境变量配置

**在/etc/profile文件中主要增加的内容是：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. ZK\_HOME=/opt/zookeeper/zookeeper-3.4.10
2. HBASE\_HOME=/opt/hbase/hbase-1.2.5

**PATH中增加：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. $HBASE\_HOME/bin:$ZK\_HOME/bin

**我的/etc/profile文件中主要的配置：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. export  JAVA\_HOME=/opt/java/jdk1.8.0\_121
2. export  HADOOP\_HOME=/opt/hadoop/hadoop-2.8.0
3. export  HADOOP\_CONF\_DIR=${HADOOP\_HOME}/etc/hadoop
4. export  HADOOP\_COMMON\_LIB\_NATIVE\_DIR=${HADOOP\_HOME}/lib/native
5. export  HADOOP\_OPTS="-Djava.library.path=${HADOOP\_HOME}/lib"
6. export  HIVE\_HOME=/opt/hive/apache-hive-2.1.1-bin
7. export  HIVE\_CONF\_DIR=${HIVE\_HOME}/conf
8. export  SQOOP\_HOME=/opt/sqoop/sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha
9. export  HBASE\_HOME=/opt/hbase/hbase-1.2.5
10. export  ZK\_HOME=/opt/zookeeper/zookeeper-3.4.10
11. export  CLASS\_PATH=.:${JAVA\_HOME}/lib:${HIVE\_HOME}/lib:$CLASS\_PATH
12. exportPATH=.:${JAVA\_HOME}/bin:${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin:${SPARK\_HOME}/bin:${ZOOKEEPER\_HOME}/bin:${HIVE\_HOME}/bin:${SQOOP\_HOME}/bin:${HBASE\_HOME}:${ZK\_HOME}/bin:$PATH

**说明：环境变量配置完后，要执行source /etc/profile才能生效。**

## 2.2  conf目录下的配置

       对/opt/hbase/hbase-1.2.5/conf目录下的一系列文件做配置。使用命令进入到该目录：

**cd      /opt/hbase/hbase-1.2.5/conf**

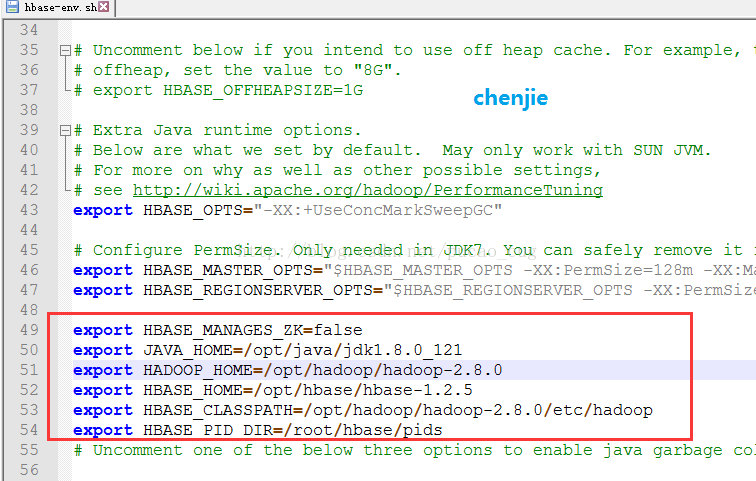
### 2.2.1 修改hbase-env.sh文件

        在/opt/hbase/hbase-1.2.5/conf/hbase-env.sh中增加内容：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. export JAVA\_HOME=/opt/java/jdk1.8.0\_121
2. export HADOOP\_HOME=/opt/hadoop/hadoop-2.8.0
3. export HBASE\_HOME=/opt/hbase/hbase-1.2.5
4. export HBASE\_CLASSPATH=/opt/hadoop/hadoop-2.8.0/etc/hadoop
5. export HBASE\_PID\_DIR=/root/hbase/pids
6. export HBASE\_MANAGES\_ZK=false

如图：



### 2.2.2 修改配置文件hbase-site.xml

      在该配置文件中可以给hbase配置一个临时目录，这里指定为mkdir /root/hbase/tmp，先执行命令创建文件夹。

**执行新建目录的命令：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. mkdir  /root/hbase
2. mkdir  /root/hbase/tmp
3. mkdir  /root/hbase/pids

**在<configuration>节点内增加以下配置：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. <property>
2. <name>hbase.rootdir</name>
3. <value>hdfs://hserver1:9000/hbase</value>
4. <description>The directory shared byregion servers.</description>
5. </property>
6. <property>
7. <name>hbase.zookeeper.property.clientPort</name>
8. <value>2181</value>
9. <description>Property from ZooKeeper'sconfig zoo.cfg. The port at which the clients will connect.
10. </description>
11. </property>
12. <property>
13. <name>zookeeper.session.timeout</name>
14. <value>120000</value>
15. </property>
16. <property>
17. <name>hbase.zookeeper.quorum</name>
18. <value>hserver1,hserver2,hserver3</value>
19. </property>
20. <property>
21. <name>hbase.tmp.dir</name>
22. <value>/root/hbase/tmp</value>
23. </property>
24. <property>
25. <name>hbase.cluster.distributed</name>
26. <value>true</value>
27. </property>

**修改完成后，我的hbase-site.xml内容是：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. <?xmlversion="1.0"?>
2. <?xml-stylesheettype="text/xsl" href="configuration.xsl"?>
3. <!--
4. /\*\*
5. \*
6. \*Licensed to the Apache Software Foundation (ASF) under one
7. \* ormore contributor license agreements.  Seethe NOTICE file
8. \*distributed with this work for additional information
9. \*regarding copyright ownership.  The ASFlicenses this file
10. \* toyou under the Apache License, Version 2.0 (the
11. \*"License"); you may not use this file except in compliance
12. \*with the License.  You may obtain a copyof the License at
13. \*
14. \*    http://www.apache.org/licenses/LICENSE-2.0
15. \*
16. \*Unless required by applicable law or agreed to in writing, software
17. \*distributed under the License is distributed on an "AS IS" BASIS,
18. \*WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
19. \*See the License for the specific language governing permissions and
20. \*limitations under the License.
21. \*/
22. -->
23. <configuration>
24. <property>
25. <name>hbase.rootdir</name>
26. <value>hdfs://hserver1:9000/hbase</value>
27. <description>The directory shared byregion servers.</description>
28. </property>
29. <property>
30. <name>hbase.zookeeper.property.clientPort</name>
31. <value>2181</value>
32. <description>Property from ZooKeeper'sconfig zoo.cfg. The port at which the clients will connect.
33. </description>
34. </property>
35. <property>
36. <name>zookeeper.session.timeout</name>
37. <value>120000</value>
38. </property>
39. <property>
40. <name>hbase.zookeeper.quorum</name>
41. <value>hserver1,hserver2,hserver3</value>
42. </property>
43. <property>
44. <name>hbase.tmp.dir</name>
45. <value>/root/hbase/tmp</value>
46. </property>
47. <property>
48. <name>hbase.cluster.distributed</name>
49. <value>true</value>
50. </property>
52. </configuration>

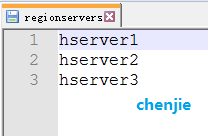
### 2.2.3  修改regionservers文件

将该文件内容修改为：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. hserver1
2. hserver2
3. hserver3

**如图：**

****

# 3  启动和测试

## 3.1 启动

**Hbase是基于hadoop提供的分布式文件系统的，所以启动Hbase之前，先确保hadoop在正常运行，另外Hbase还依赖于zookkeeper，本来我们可以用hbase自带的zookeeper，但是我们上面的配置启用的是我们自己的zookeeper集群，所以在启动hbase前，还要确保zokeeper已经正常运行。**

**Hbase可以只在hadoop的某个namenode节点上安装，也可以在所有的hadoop节点上安装，但是启动的时候只需要在一个节点上启动就行了，本例中，我在hserver1、hserver2、hserver3都安装了Hbase，启动的时候只需要在hserver1上启动就OK。**

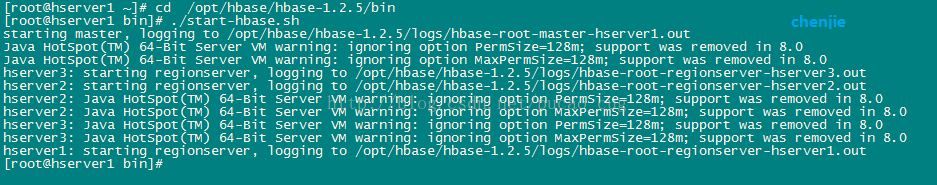
      在hserver1上执行命令，进入到Hbase的bin目录内，命令是：

**cd     /opt/hbase/hbase-1.2.5/bin**

       执行命令启动Hbase，命令是：

**./start-hbase.sh**

**如图：**

****

**输出信息是：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. [root@hserver1~]# cd  /opt/hbase/hbase-1.2.5/bin
2. [root@hserver1bin]# ./start-hbase.sh
3. starting master,logging to /opt/hbase/hbase-1.2.5/logs/hbase-root-master-hserver1.out
4. Java HotSpot(TM)64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in8.0
5. Java HotSpot(TM)64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removedin 8.0
6. hserver3:starting regionserver, logging to /opt/hbase/hbase-1.2.5/logs/hbase-root-regionserver-hserver3.out
7. hserver2:starting regionserver, logging to/opt/hbase/hbase-1.2.5/logs/hbase-root-regionserver-hserver2.out
8. hserver2: JavaHotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; supportwas removed in 8.0
9. hserver2: JavaHotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; supportwas removed in 8.0
10. hserver3: JavaHotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; supportwas removed in 8.0
11. hserver3: Java HotSpot(TM)64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removedin 8.0
12. hserver1:starting regionserver, logging to/opt/hbase/hbase-1.2.5/logs/hbase-root-regionserver-hserver1.out
13. [root@hserver1bin]#

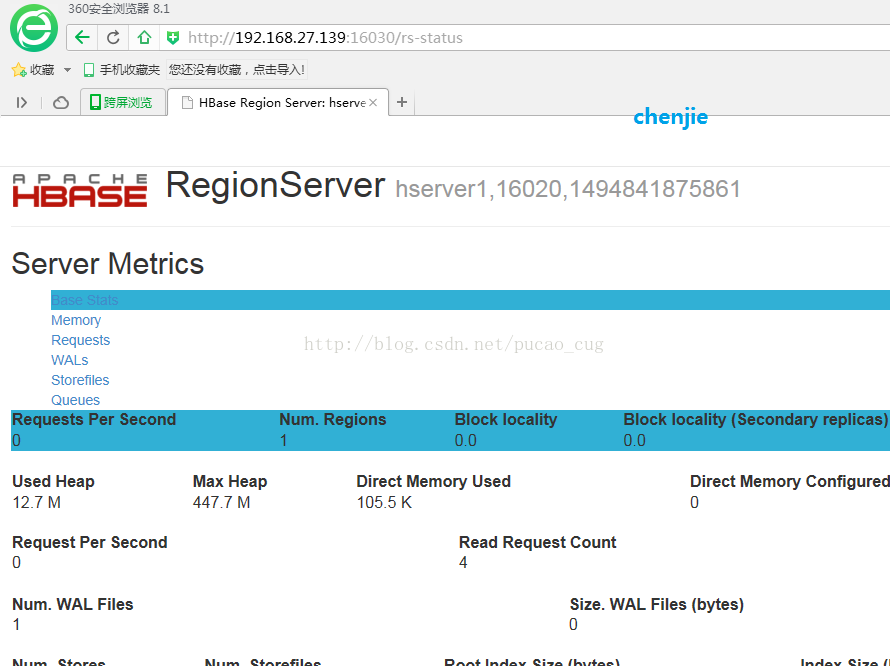
## 3.2  测试

### 3.2.1  用浏览器访问Hbase状态信息

http://192.168.137.130:16030/rs-status

      直接访问地址：<http://192.168.27.139:16030/>

**如图：**

****

### 3.2.2 启动hbase的命令行

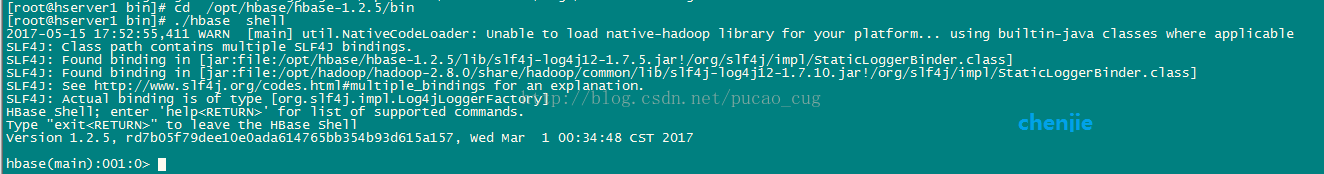
      执行命令，进入到Hbase的bin目录内，命令是：

**cd  /opt/hbase/hbase-1.2.5/bin**

      执行命令启动Hbase命令行窗口，命令是：

**./hbase  shell**

**如图：**

****

**完整的输出是：**

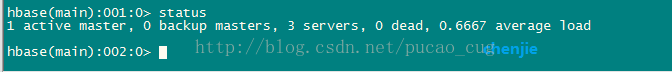
**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. [root@hserver1 bin]# cd  /opt/hbase/hbase-1.2.5/bin
2. [root@hserver1 bin]# ./hbase  shell
3. 2017-05-15 17:52:55,411 WARN  [main] util.NativeCodeLoader: Unable to loadnative-hadoop library for your platform... using builtin-java classes whereapplicable
4. SLF4J: Class path contains multiple SLF4Jbindings.
5. SLF4J: Found binding in[jar:file:/opt/hbase/hbase-1.2.5/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
6. SLF4J: Found binding in[jar:file:/opt/hadoop/hadoop-2.8.0/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
7. SLF4J: Seehttp://www.slf4j.org/codes.html#multiple\_bindings for an explanation.
8. SLF4J: Actual binding is of type[org.slf4j.impl.Log4jLoggerFactory]
9. HBase Shell; enter 'help<RETURN>' forlist of supported commands.
10. Type "exit<RETURN>" toleave the HBase Shell
11. Version 1.2.5,rd7b05f79dee10e0ada614765bb354b93d615a157, Wed Mar  1 00:34:48 CST 2017
13. hbase(main):001:0>

**在hbase命令行模式下，可以输入一系列hbase命令，进行测试**

输入：**status**

**如图：**

****

**完整的输出是：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72229223) [copy](http://blog.csdn.net/pucao_cug/article/details/72229223)

1. hbase(main):001:0> status
2. 1 active master, 0 backup masters, 3servers, 0 dead, 0.6667 average load
4. hbase(main):002:0>

      如果要退出Hbase命令行模式的话，输入：**exit**

**如图：**

**http://img.blog.csdn.net/20170515215715319?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvcHVjYW9fY3Vn/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center**

# [Linux安装zookeeper集群(CentOS7+Zookeeper3.4.10)](http://blog.csdn.net/pucao_cug/article/details/72228973)

[1 对Zookeeper集群进行配置](file:///D:\111\zookeeper%E9%9B%86%E7%BE%A4\Linux%E5%AE%89%E8%A3%85zookeeper%E9%9B%86%E7%BE%A4(CentOS7+Zookeeper3.4.10)-v1.0.0.htm#_Toc482624675)

[1.1创建对应的目录和文件](file:///D:\111\zookeeper%E9%9B%86%E7%BE%A4\Linux%E5%AE%89%E8%A3%85zookeeper%E9%9B%86%E7%BE%A4(CentOS7+Zookeeper3.4.10)-v1.0.0.htm#_Toc482624676)

[1.1.1创建目录](file:///D:\111\zookeeper%E9%9B%86%E7%BE%A4\Linux%E5%AE%89%E8%A3%85zookeeper%E9%9B%86%E7%BE%A4(CentOS7+Zookeeper3.4.10)-v1.0.0.htm#_Toc482624677)

[1.1.2创建myid文件](file:///D:\111\zookeeper%E9%9B%86%E7%BE%A4\Linux%E5%AE%89%E8%A3%85zookeeper%E9%9B%86%E7%BE%A4(CentOS7+Zookeeper3.4.10)-v1.0.0.htm#_Toc482624678)

[1.2修改配置文件zoo.cfg](file:///D:\111\zookeeper%E9%9B%86%E7%BE%A4\Linux%E5%AE%89%E8%A3%85zookeeper%E9%9B%86%E7%BE%A4(CentOS7+Zookeeper3.4.10)-v1.0.0.htm#_Toc482624679)

[2、启动和测试集群](file:///D:\111\zookeeper%E9%9B%86%E7%BE%A4\Linux%E5%AE%89%E8%A3%85zookeeper%E9%9B%86%E7%BE%A4(CentOS7+Zookeeper3.4.10)-v1.0.0.htm#_Toc482624680)

[2.1启动zookeeper集群](file:///D:\111\zookeeper%E9%9B%86%E7%BE%A4\Linux%E5%AE%89%E8%A3%85zookeeper%E9%9B%86%E7%BE%A4(CentOS7+Zookeeper3.4.10)-v1.0.0.htm#_Toc482624681)

[2.2测试](file:///D:\111\zookeeper%E9%9B%86%E7%BE%A4\Linux%E5%AE%89%E8%A3%85zookeeper%E9%9B%86%E7%BE%A4(CentOS7+Zookeeper3.4.10)-v1.0.0.htm#_Toc482624682)

**关键字：Linux  CentOS   Zookeeper**[**Java**](http://lib.csdn.net/base/17)

**版本号：CentOS7  Zookeeper3.4.10   JDK1.8**

**说明：在每个机器上安装zookeeper，然后在稍微修改一下配置，就构成了集群，单机安装zookeeper并且设置为开机启动请参考该博文**

<http://blog.csdn.net/pucao_cug/article/details/71240246>

# 1对Zookeeper集群进行配置

    在hserver1、hserver2、hserver3这3太机器上都安装了zookeeper后(开机启动不是必须的)，按照下面的步骤操作即可。

## 1.1创建对应的目录和文件

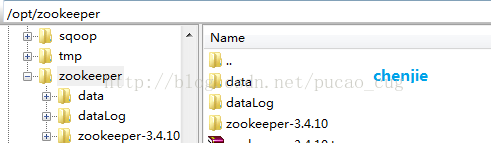
### 1.1.1创建目录

      在hserver1、hserver2、hserver3这3台机器上执行命令创建两个目录：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72228973) [copy](http://blog.csdn.net/pucao_cug/article/details/72228973)

1. mkdir   /opt/zookeeper/data
2. mkdir   /opt/zookeeper/dataLog

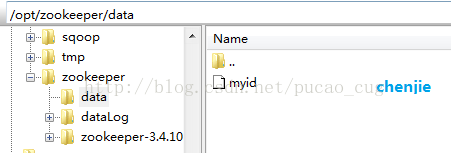
      创建完成后，**如图：**

****

### 1.1.2 创建myid文件

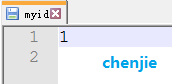
          在hserver1、hserver2、hserver3这3台机器的**/opt/zookeeper/data**目录内都创建一个名为myid的文件

**如图：**

****

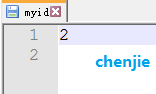
将hserver1机器上的/opt/zookeeper/data/myid文件的内容编辑为1

**如图：**

****

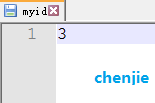
     将hserver2机器上的/opt/zookeeper/data/myid文件的内容编辑为2

**如图：**

****

    将hserver3机器上的/opt/zookeeper/data/myid文件的内容编辑为3

**如图：**

****

**说明：上面新建的目录可以不和我一样，myid中的数字编号也可以不一样，只要和下面1.2中的zoo.cfg的配置对应即可，但是建成一样也无妨。**

## 1.2修改配置文件zoo.cfg

      把集群内的zookeeper的zoo.cfg配置文件都修改成一样的内容，主要是在末尾增加配置：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72228973) [copy](http://blog.csdn.net/pucao_cug/article/details/72228973)

1. dataDir=/opt/zookeeper/data
2. dataLogDir=/opt/zookeeper/dataLog
3. server.1=hserver1:2888:3888
4. server.2=hserver2:2888:3888
5. server.3=hserver3:2888:3888

**我自己的zoo.cfg文件内容是：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72228973) [copy](http://blog.csdn.net/pucao_cug/article/details/72228973)

1. # The number ofmilliseconds of each tick
2. tickTime=2000
3. # The number ofticks that the initial
4. #synchronization phase can take
5. initLimit=10
6. # The number ofticks that can pass between
7. # sending arequest and getting an acknowledgement
8. syncLimit=5
9. # the directorywhere the snapshot is stored.
10. # do not use /tmpfor storage, /tmp here is just
11. # example sakes.
12. dataDir=/tmp/zookeeper
13. # the port atwhich the clients will connect
14. clientPort=2181
15. # the maximumnumber of client connections.
16. # increase thisif you need to handle more clients
17. #maxClientCnxns=60
18. #
19. # Be sure toread the maintenance section of the
20. # administratorguide before turning on autopurge.
21. #
22. #http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance
23. #
24. # The number ofsnapshots to retain in dataDir
25. #autopurge.snapRetainCount=3
26. # Purge taskinterval in hours
27. # Set to"0" to disable auto purge feature
28. #autopurge.purgeInterval=1
30. dataDir=/opt/zookeeper/data
31. dataLogDir=/opt/zookeeper/dataLog
32. server.1=hserver1:2888:3888
33. server.2=hserver2:2888:3888
34. server.3=hserver3:2888:3888

**说明：dataDir和dataLogDir需要自己创建，目录可以自己制定，对应即可。server.1中的这个1需要和hserver1这个机器上的dataDir目录中的myid文件中的数值对应。server.2中的这个2需要和hserver2这个机器上的dataDir目录中的myid文件中的数值对应。server.3中的这个3需要和hserver3这个机器上的dataDir目录中的myid文件中的数值对应。当然，数值你可以随便用，只要对应即可。2888和3888的端口号也可以随便用，因为在不同机器上，用成一样也无所谓。**

# 2  启动和测试集群

## 2.1  启动zookeeper集群

        如果想要开机启动，请参考该博文<http://blog.csdn.net/pucao_cug/article/details/71240246>

第三章的内容( 集群中每台机器都设置为开机启动，那么也就相当于是开机启动了)。

下面的步骤是手动启动zookeeper的步骤。

       分别在3台机器上执行下面的命令，启动机器上安装的zookeeper，hserver1、hserver2、hserver3的启动顺序无关紧要，也没必要说一台启动完成后在去启动另外一台，随意就行，没有关系的。

       执行启动的命令是：

**/opt/zookeeper/zookeeper-3.4.10/bin/zkServer.sh        start**

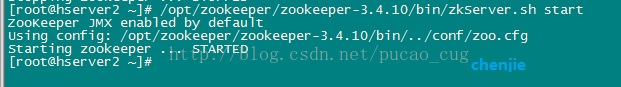
**hserver1机器**

**如图：**



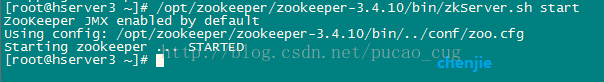
**hserver2机器**

**如图：**



**hserver3机器**

**如图：**

****

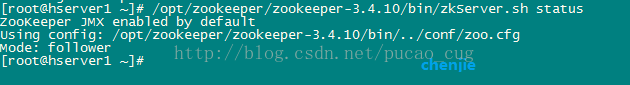
## 2.2  测试

         查看zookeeper集群中的zookeeper节点的状态，会发现其中一个是leader，其余是follower。分别在3台机器上执行命令查看zookkeeper状态。按下面步骤操作：

**在hserver1上执行命令：**

**/opt/zookeeper/zookeeper-3.4.10/bin/zkServer.sh     status**

**如图：**

****

**文本是：**

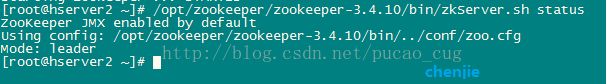
**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72228973) [copy](http://blog.csdn.net/pucao_cug/article/details/72228973)

1. [root@hserver1 ~]#/opt/zookeeper/zookeeper-3.4.10/bin/zkServer.sh status
2. ZooKeeper JMX enabled by default
3. Using config:/opt/zookeeper/zookeeper-3.4.10/bin/../conf/zoo.cfg
4. Mode: follower
5. [root@hserver1 ~]#

**在hserver2上执行命令：**

**/opt/zookeeper/zookeeper-3.4.10/bin/zkServer.sh status**

**如图：**

****

**文本是：**

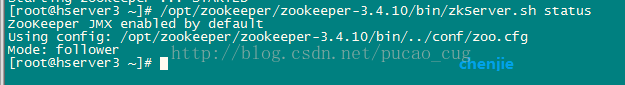
**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72228973) [copy](http://blog.csdn.net/pucao_cug/article/details/72228973)

1. [root@hserver2 ~]#/opt/zookeeper/zookeeper-3.4.10/bin/zkServer.sh status
2. ZooKeeper JMX enabled by default
3. Using config:/opt/zookeeper/zookeeper-3.4.10/bin/../conf/zoo.cfg
4. Mode: leader
5. [root@hserver2 ~]#

**在hserver3上执行命令：**

**/opt/zookeeper/zookeeper-3.4.10/bin/zkServer.sh     status**

**如图：**

****

**文本是：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72228973) [copy](http://blog.csdn.net/pucao_cug/article/details/72228973)

1. [root@hserver3~]# /opt/zookeeper/zookeeper-3.4.10/bin/zkServer.sh status
2. ZooKeeper JMX enabled by default
3. Using config:/opt/zookeeper/zookeeper-3.4.10/bin/../conf/zoo.cfg
4. Mode: follower
5. [root@hserver3 ~]#

# [Linux安装Spark集群(CentOS7+Spark2.1.1+Hadoop2.8.0)](http://blog.csdn.net/pucao_cug/article/details/72353701)

[安装Spark依赖的Scala](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714320)

[1.1下载和解压缩Scala](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714321)

[1.2 配置环境变量](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714322)

[1.3 验证Scala](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714323)

[2下载和解压缩Spark](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714324)

[2.1 下载Spark压缩包](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714325)

[2.2 解压缩Spark](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714326)

[3 Spark相关的配置](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714327)

[3.1 配置环境变量](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714328)

[3.2 配置conf目录下的文件](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714329)

[3.2.1 新建spark-env.h文件](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714330)

[3.2.2 新建slaves文件](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714331)

[4 启动和测试Spark集群](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714332)

[4.1 启动Spark](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714333)

[4.2 测试和使用Spark集群](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714334)

[4.2.1访问Spark集群提供的URL](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714335)

[4.2.2运行Spark提供的计算圆周率的示例程序](file:///D:\111\spark%E5%AE%89%E8%A3%85\Linux%E5%AE%89%E8%A3%85Spark(CentOS7+Spark2.1.1+Hadoop2.8.0)-v1.0.0.htm#_Toc482714336)

关键字：Linux   CentOS  [**Hadoop**](http://lib.csdn.net/base/20)  [**Spark**](http://lib.csdn.net/base/10)   Scala   [**Java**](http://lib.csdn.net/base/17)

版本号：CentOS7  Hadoop2.8.0  Spark2.1.1   Scala2.12.2   JDK1.8

**说明：Spark可以在只安装了JDK、Scala的机器上直接单机安装，但是这样的话只能使用单机模式运行不涉及分布式运算和分布式存储的代码，例如可以单机安装Spark，单机运行计算圆周率的Spark程序。但是我们要运行的是Spark集群，并且需要调用Hadoop的分布式文件系统，所以请你先安装Hadoop，Hadoop集群的安装可以参考该博文：**

<http://blog.csdn.net/pucao_cug/article/details/71698903>

**安装单机版的Spark可以参考该博文：**

[**http://blog.csdn.net/pucao\_cug/article/details/72377219**](http://blog.csdn.net/pucao_cug/article/details/72377219)

**Spark集群的最小化安装只需要安装这些东西：JDK  、Scala  、Hadoop  、Spark**

# 1 安装Spark依赖的Scala

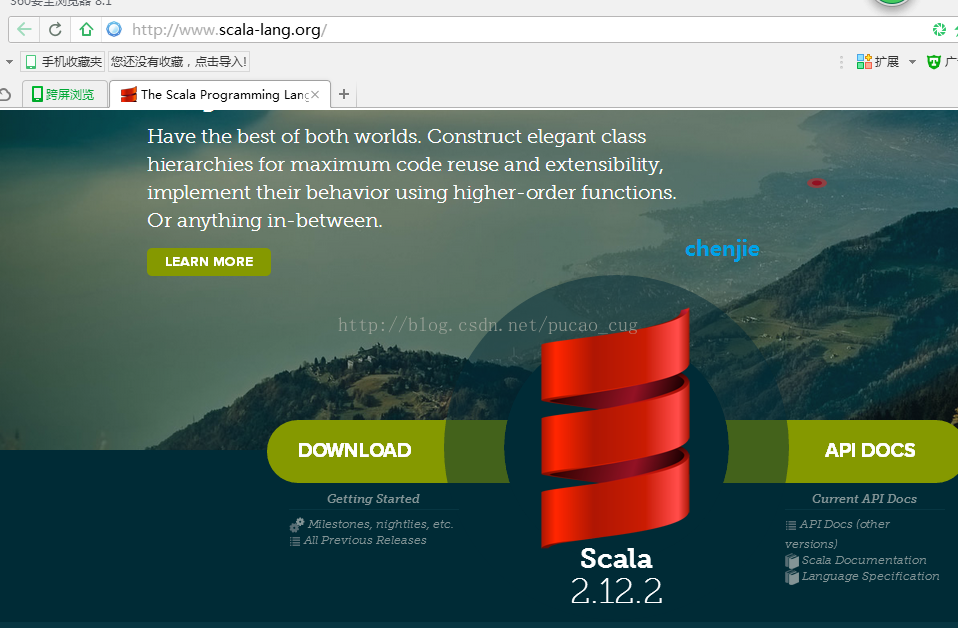
          Hadoop的安装请参考上面提到的博文，因为Spark依赖scala，所以在安装Spark之前，这里要先安装scala。在每个节点上都进行安装。

## 1.1  下载和解压缩Scala

        打开地址：<http://www.scala-lang.org/>

        目前最新版是2.12.2，我就安装该版本

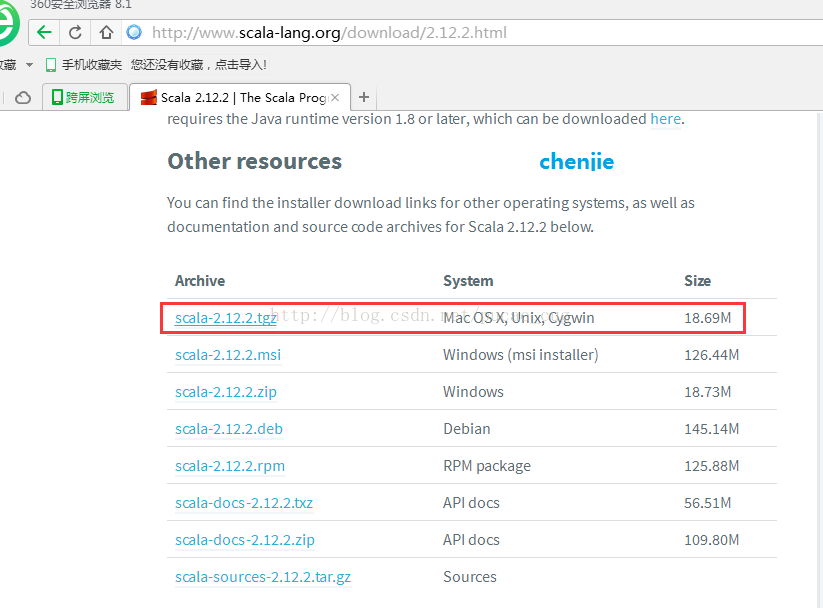
**如图：**

****

直接打开下面的地址也可以：

<http://www.scala-lang.org/download/2.12.2.html>

**如图：**

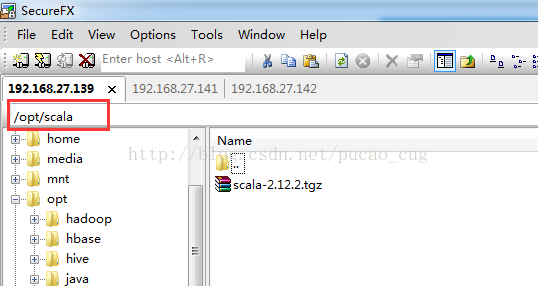
****

直接用下面的地址下载tgz包也可以：

<https://downloads.lightbend.com/scala/2.12.2/scala-2.12.2.tgz>

在Linux服务器的opt目录下新建一个名为scala的文件夹，并将下载的压缩包上载上去

**如图：**



执行命令，进入到该目录：

**cd    /opt/scala**

执行命令进行解压缩：

**tar   -xvf    scala-2.12.2**

## 1.2  配置环境变量

       编辑/etc/profile这个文件，在文件中增加一行配置：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. export    SCALA\_HOME=/opt/scala/scala-2.12.2

      在该文件的PATH变量中增加下面的内容：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. ${SCALA\_HOME}/bin

      添加完成后，我的/etc/profile的配置如下：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. export JAVA\_HOME=/opt/java/jdk1.8.0\_121
2. export HADOOP\_HOME=/opt/hadoop/hadoop-2.8.0
3. export HADOOP\_CONF\_DIR=${HADOOP\_HOME}/etc/hadoop
4. export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=${HADOOP\_HOME}/lib/native
5. export HADOOP\_OPTS="-Djava.library.path=${HADOOP\_HOME}/lib"
6. export HIVE\_HOME=/opt/hive/apache-hive-2.1.1-bin
7. export HIVE\_CONF\_DIR=${HIVE\_HOME}/conf
8. export SQOOP\_HOME=/opt/sqoop/sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha
9. export HBASE\_HOME=/opt/hbase/hbase-1.2.5
10. export ZK\_HOME=/opt/zookeeper/zookeeper-3.4.10
11. export SCALA\_HOME=/opt/scala/scala-2.12.2
12. export CLASS\_PATH=.:${JAVA\_HOME}/lib:${HIVE\_HOME}/lib:$CLASS\_PATH
13. export PATH=.:${JAVA\_HOME}/bin:${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin:${SPARK\_HOME}/bin:${ZK\_HOME}/bin:${HIVE\_HOME}/bin:${SQOOP\_HOME}/bin:${HBASE\_HOME}/bin:${SCALA\_HOME}/bin:$PATH

**说明：你可以只关注开头说的JDK  SCALA   Hadoop  Spark的环境变量，其余的诸如Zookeeper、Hbase、Hive、Sqoop都不用管。**

**如图：**

****

      环境变量配置完成后，执行下面的命令：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. source   /etc/profile

## 1.3 验证Scala

    执行命令：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. scala     -version

**如图：**

http://img.blog.csdn.net/20170516222239196?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvcHVjYW9fY3Vn/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

# 2 下载和解压缩Spark

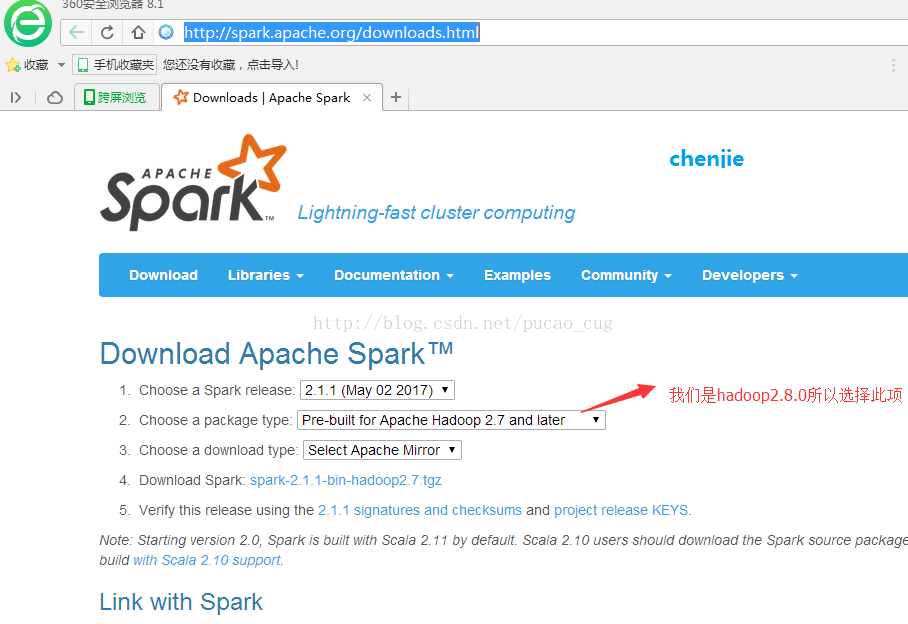
      在每个节点上都安装Spark，也就是重复下面的步骤。

## 2.1 下载Spark压缩包

      打开下载地址：

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

**如图：**



**点击上图的 Download Spark，相当于是直接打开地址：**

<https://www.apache.org/dyn/closer.lua/spark/spark-2.1.1/spark-2.1.1-bin-hadoop2.7.tgz>

      下载后得到了大约200M的文件： spark-2.1.1-bin-hadoop2.7

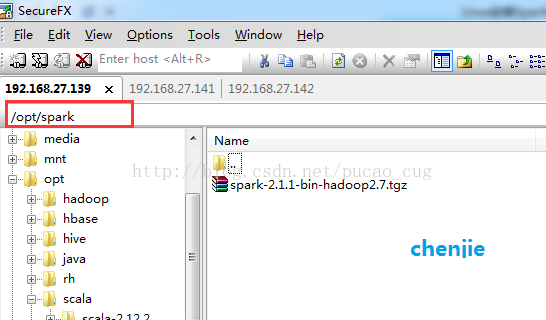
        直接用下面的地址下面也可以：

<http://mirrors.hust.edu.cn/apache/spark/spark-2.1.1/spark-2.1.1-bin-hadoop2.7.tgz>

## 2.2  解压缩Spark

       下载完成后，在Linux服务器的opt目录下新建一个名为spark的文件夹，把刚才下载的压缩包，上载上去。

**如图：**

****

**进入到该目录内，也就是执行下面的命令：**

**cd    /opt/spark**

**执行解压缩命令：**

**tar   -zxvf   spark-2.1.1-bin-hadoop2.7.tgz**

# 3  Spark相关的配置

         说明：因为我们搭建的是基于hadoop集群的Spark集群，所以每个hadoop节点上我都安装了Spark，都需要按照下面的步骤做配置，启动的话只需要在Spark集群的Master机器上启动即可，我这里是在hserver1上启动。

## 3.1  配置环境变量

编辑/etc/profile文件，增加

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. export  SPARK\_HOME=/opt/spark/spark-2.1.1-bin-hadoop2.7

      上面的变量添加完成后编辑该文件中的PATH变量，添加

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. ${SPARK\_HOME}/bin

**注意：因为$SPARK\_HOME/sbin目录下有一些文件名称和$HADOOP\_HOME/sbin目录下的文件同名，为了避免同名文件冲突，这里不在PATH变量里添加$SPARK\_HOME/sbin只添加了$SPARK\_HOME/bin。**

修改完成后，我的/etc/profile文件内容是：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. export JAVA\_HOME=/opt/java/jdk1.8.0\_121
2. export HADOOP\_HOME=/opt/hadoop/hadoop-2.8.0
3. export HADOOP\_CONF\_DIR=${HADOOP\_HOME}/etc/hadoop
4. export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=${HADOOP\_HOME}/lib/native
5. export HADOOP\_OPTS="-Djava.library.path=${HADOOP\_HOME}/lib"
6. export HIVE\_HOME=/opt/hive/apache-hive-2.1.1-bin
7. export HIVE\_CONF\_DIR=${HIVE\_HOME}/conf
8. export SQOOP\_HOME=/opt/sqoop/sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha
9. export HBASE\_HOME=/opt/hbase/hbase-1.2.5
10. export ZK\_HOME=/opt/zookeeper/zookeeper-3.4.10
11. export SCALA\_HOME=/opt/scala/scala-2.12.2
12. export SPARK\_HOME=/opt/spark/spark-2.1.1-bin-hadoop2.7
13. export CLASS\_PATH=.:${JAVA\_HOME}/lib:${HIVE\_HOME}/lib:$CLASS\_PATH
14. export PATH=.:${JAVA\_HOME}/bin:${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin:${SPARK\_HOME}/bin:${ZK\_HOME}/bin:${HIVE\_HOME}/bin:${SQOOP\_HOME}/bin:${HBASE\_HOME}:${SCALA\_HOME}/bin:$PATH

**说明：你可以只关注开头说的JDK  SCALA   Hadoop  Spark的环境变量，其余的诸如Zookeeper、Hbase、Hive、Sqoop都不用管。**

**如图：**

****

编辑完成后，执行命令：

**source    /etc/profile**

## 3.2 配置conf目录下的文件

        对/opt/spark/spark-2.1.1-bin-hadoop2.7/conf目录下的文件进行配置。

### 3.2.1  新建spark-env.h文件

        执行命令，进入到/opt/spark/spark-2.1.1-bin-hadoop2.7/conf目录内：

**cd    /opt/spark/spark-2.1.1-bin-hadoop2.7/conf**

       以spark为我们创建好的模板创建一个spark-env.h文件，命令是：

**cp    spark-env.sh.template   spark-env.sh**

**如图：**

**http://img.blog.csdn.net/20170516222415057?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvcHVjYW9fY3Vn/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center**

    编辑spark-env.h文件，在里面加入配置(具体路径以自己的为准)：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. export SCALA\_HOME=/opt/scala/scala-2.12.2
2. export JAVA\_HOME=/opt/java/jdk1.8.0\_121
3. export HADOOP\_HOME=/opt/hadoop/hadoop-2.8.0
4. export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop
5. export SPARK\_HOME=/opt/spark/spark-2.1.1-bin-hadoop2.7
6. export SPARK\_MASTER\_IP=hserver1
7. export SPARK\_EXECUTOR\_MEMORY=1G

### 3.2.2 新建slaves文件

执行命令，进入到/opt/spark/spark-2.1.1-bin-hadoop2.7/conf目录内：

**cd   /opt/spark/spark-2.1.1-bin-hadoop2.7/conf**

以spark为我们创建好的模板创建一个slaves文件，命令是：

**cp    slaves.template   slaves**

**如图：**

**http://img.blog.csdn.net/20170516222438900?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvcHVjYW9fY3Vn/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center**

编辑slaves文件，里面的内容为：

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. hserver2
2. hserver3

# 4 启动和测试Spark集群

## 4.1 启动Spark

          因为spark是依赖于hadoop提供的分布式文件系统的，所以在启动spark之前，先确保hadoop在正常运行。Hadoop2.8.0的安装和启动，请参考该博文：

<http://blog.csdn.net/pucao_cug/article/details/71698903>

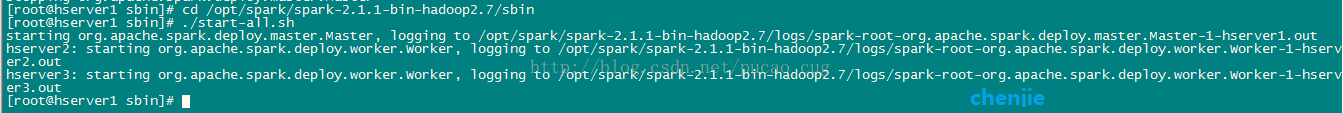
        在hadoop正常运行的情况下，在hserver1（也就是hadoop的namenode，spark的marster节点）上执行命令：

**cd   /opt/spark/spark-2.1.1-bin-hadoop2.7/sbin**

执行启动脚本：

**./start-all.sh**

**如图：**

****

**完整内容是：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. [root@hserver1 sbin]# cd/opt/spark/spark-2.1.1-bin-hadoop2.7/sbin
2. [root@hserver1 sbin]# ./start-all.sh
3. starting org.apache.spark.deploy.master.Master,logging to/opt/spark/spark-2.1.1-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.master.Master-1-hserver1.out
4. hserver2: startingorg.apache.spark.deploy.worker.Worker, logging to/opt/spark/spark-2.1.1-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-hserver2.out
5. hserver3: startingorg.apache.spark.deploy.worker.Worker, logging to/opt/spark/spark-2.1.1-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-hserver3.out
6. [root@hserver1 sbin]#

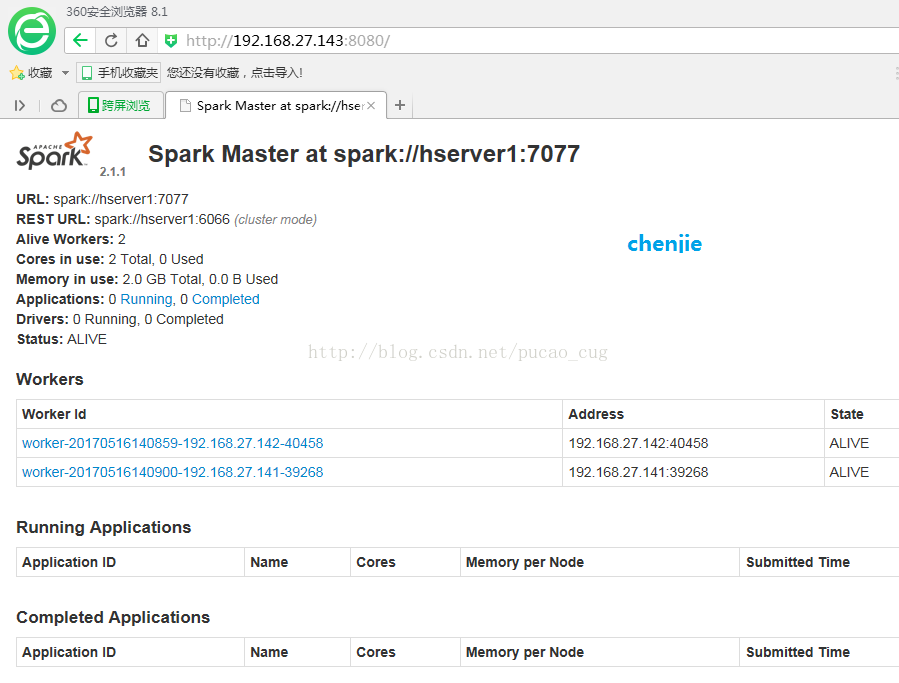
**注意：上面的命令中有./这个不能少，./的意思是执行当前目录下的start-all.sh脚本。**

## 4.2  测试和使用Spark集群

### 4.2.1  访问Spark集群提供的URL

        在浏览器里访问Mster机器，我的Spark集群里Master机器是hserver1，IP地址是192.168.27.143，访问8080端口，URL是：<http://192.168.27.143:8080/>

**如图：**

****

### 4.2.2  运行Spark提供的计算圆周率的示例程序

      这里只是简单的用local模式运行一个计算圆周率的Demo。按照下面的步骤来操作。

**第一步，进入到Spark的根目录，也就是执行下面的脚本：**

**cd     /opt/spark/spark-2.1.1-bin-hadoop2.7**

**如图：**

**http://img.blog.csdn.net/20170516223201021?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvcHVjYW9fY3Vn/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center**

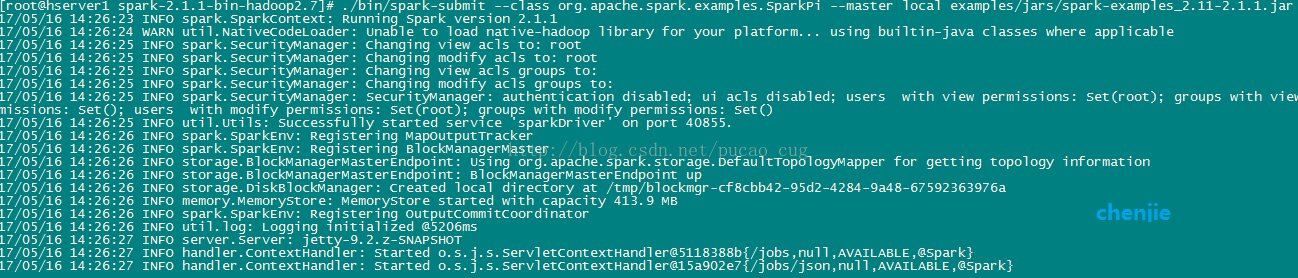
**第二步，调用Spark自带的计算圆周率的Demo，执行下面的命令：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. ./bin/spark-submit  --class  org.apache.spark.examples.SparkPi  --master local   examples/jars/spark-examples\_2.11-2.1.1.jar

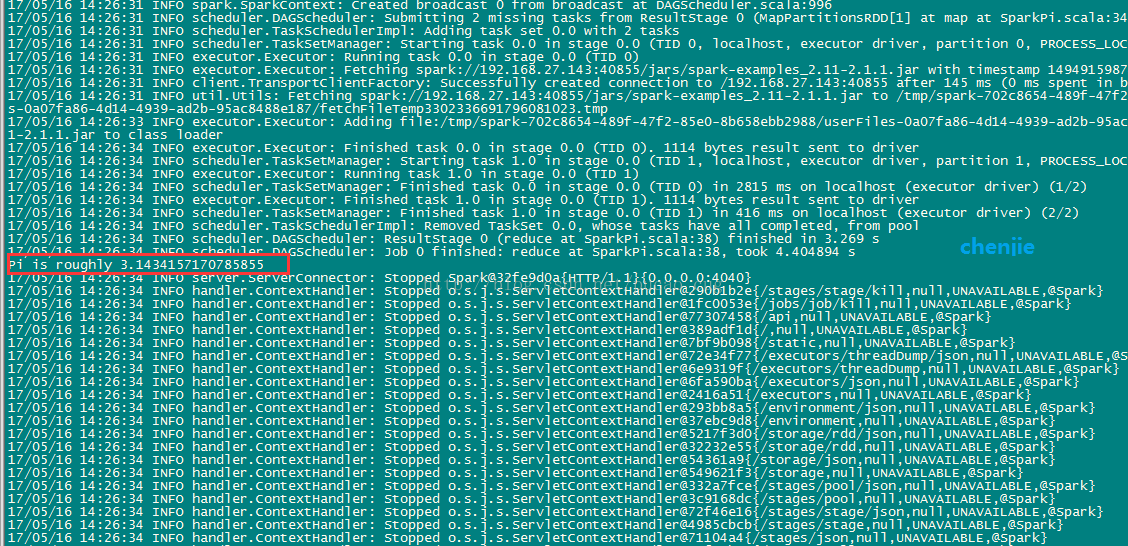
**命令执行后，spark示例程序已经开始执行**

**如图：**

****

        很快执行结果出来了，执行结果我用红框标出来了

**如图：**

****

**完整的控制台输出是：**

**[plain]** [view plain](http://blog.csdn.net/pucao_cug/article/details/72353701) [copy](http://blog.csdn.net/pucao_cug/article/details/72353701)

1. [root@hserver1 bin]# cd  /opt/spark/spark-2.1.1-bin-hadoop2.7
2. [root@hserver1 spark-2.1.1-bin-hadoop2.7]# ./bin/spark-submit--class org.apache.spark.examples.SparkPi --master localexamples/jars/spark-examples\_2.11-2.1.1.jar
3. 17/05/16 14:26:23 INFO spark.SparkContext: Running Spark version2.1.1
4. 17/05/16 14:26:24 WARN util.NativeCodeLoader: Unable to loadnative-hadoop library for your platform... using builtin-java classes whereapplicable
5. 17/05/16 14:26:25 INFO spark.SecurityManager: Changing view acls to:root
6. 17/05/16 14:26:25 INFO spark.SecurityManager: Changing modify aclsto: root
7. 17/05/16 14:26:25 INFO spark.SecurityManager: Changing view aclsgroups to:
8. 17/05/16 14:26:25 INFO spark.SecurityManager: Changing modify aclsgroups to:
9. 17/05/16 14:26:25 INFO spark.SecurityManager: SecurityManager:authentication disabled; ui acls disabled; users  with view permissions: Set(root); groups withview permissions: Set(); users  withmodify permissions: Set(root); groups with modify permissions: Set()
10. 17/05/16 14:26:25 INFO util.Utils: Successfully started service'sparkDriver' on port 40855.
11. 17/05/16 14:26:26 INFO spark.SparkEnv: Registering MapOutputTracker
12. 17/05/16 14:26:26 INFO spark.SparkEnv: RegisteringBlockManagerMaster
13. 17/05/16 14:26:26 INFO storage.BlockManagerMasterEndpoint: Usingorg.apache.spark.storage.DefaultTopologyMapper for getting topology information
14. 17/05/16 14:26:26 INFO storage.BlockManagerMasterEndpoint:BlockManagerMasterEndpoint up
15. 17/05/16 14:26:26 INFO storage.DiskBlockManager: Created localdirectory at /tmp/blockmgr-cf8cbb42-95d2-4284-9a48-67592363976a
16. 17/05/16 14:26:26 INFO memory.MemoryStore: MemoryStore started withcapacity 413.9 MB
17. 17/05/16 14:26:26 INFO spark.SparkEnv: RegisteringOutputCommitCoordinator
18. 17/05/16 14:26:26 INFO util.log: Logging initialized @5206ms
19. 17/05/16 14:26:27 INFO server.Server: jetty-9.2.z-SNAPSHOT
20. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@5118388b{/jobs,null,AVAILABLE,@Spark}
21. 17/05/16 14:26:27 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@15a902e7{/jobs/json,null,AVAILABLE,@Spark}
22. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@7876d598{/jobs/job,null,AVAILABLE,@Spark}
23. 17/05/16 14:26:27 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@4a3e3e8b{/jobs/job/json,null,AVAILABLE,@Spark}
24. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@5af28b27{/stages,null,AVAILABLE,@Spark}
25. 17/05/16 14:26:27 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@71104a4{/stages/json,null,AVAILABLE,@Spark}
26. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@4985cbcb{/stages/stage,null,AVAILABLE,@Spark}
27. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@72f46e16{/stages/stage/json,null,AVAILABLE,@Spark}
28. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@3c9168dc{/stages/pool,null,AVAILABLE,@Spark}
29. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@332a7fce{/stages/pool/json,null,AVAILABLE,@Spark}
30. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@549621f3{/storage,null,AVAILABLE,@Spark}
31. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@54361a9{/storage/json,null,AVAILABLE,@Spark}
32. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@32232e55{/storage/rdd,null,AVAILABLE,@Spark}
33. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@5217f3d0{/storage/rdd/json,null,AVAILABLE,@Spark}
34. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@37ebc9d8{/environment,null,AVAILABLE,@Spark}
35. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@293bb8a5{/environment/json,null,AVAILABLE,@Spark}
36. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@2416a51{/executors,null,AVAILABLE,@Spark}
37. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@6fa590ba{/executors/json,null,AVAILABLE,@Spark}
38. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@6e9319f{/executors/threadDump,null,AVAILABLE,@Spark}
39. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@72e34f77{/executors/threadDump/json,null,AVAILABLE,@Spark}
40. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@7bf9b098{/static,null,AVAILABLE,@Spark}
41. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@389adf1d{/,null,AVAILABLE,@Spark}
42. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@77307458{/api,null,AVAILABLE,@Spark}
43. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@1fc0053e{/jobs/job/kill,null,AVAILABLE,@Spark}
44. 17/05/16 14:26:27 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@290b1b2e{/stages/stage/kill,null,AVAILABLE,@Spark}
45. 17/05/16 14:26:27 INFO server.ServerConnector: StartedSpark@32fe9d0a{HTTP/1.1}{0.0.0.0:4040}
46. 17/05/16 14:26:27 INFO server.Server: Started @5838ms
47. 17/05/16 14:26:27 INFO util.Utils: Successfully started service'SparkUI' on port 4040.
48. 17/05/16 14:26:27 INFO ui.SparkUI: Bound SparkUI to 0.0.0.0, andstarted at http://192.168.27.143:4040
49. 17/05/16 14:26:27 INFO spark.SparkContext: Added JARfile:/opt/spark/spark-2.1.1-bin-hadoop2.7/examples/jars/spark-examples\_2.11-2.1.1.jarat spark://192.168.27.143:40855/jars/spark-examples\_2.11-2.1.1.jar withtimestamp 1494915987472
50. 17/05/16 14:26:27 INFO executor.Executor: Starting executor IDdriver on host localhost
51. 17/05/16 14:26:27 INFO util.Utils: Successfully started service'org.apache.spark.network.netty.NettyBlockTransferService' on port 41104.
52. 17/05/16 14:26:27 INFO netty.NettyBlockTransferService: Servercreated on 192.168.27.143:41104
53. 17/05/16 14:26:27 INFO storage.BlockManager: Usingorg.apache.spark.storage.RandomBlockReplicationPolicy for block replicationpolicy
54. 17/05/16 14:26:27 INFO storage.BlockManagerMaster: RegisteringBlockManager BlockManagerId(driver, 192.168.27.143, 41104, None)
55. 17/05/16 14:26:27 INFO storage.BlockManagerMasterEndpoint:Registering block manager 192.168.27.143:41104 with 413.9 MB RAM,BlockManagerId(driver, 192.168.27.143, 41104, None)
56. 17/05/16 14:26:27 INFO storage.BlockManagerMaster: RegisteredBlockManager BlockManagerId(driver, 192.168.27.143, 41104, None)
57. 17/05/16 14:26:27 INFO storage.BlockManager: InitializedBlockManager: BlockManagerId(driver, 192.168.27.143, 41104, None)
58. 17/05/16 14:26:28 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@4e6d7365{/metrics/json,null,AVAILABLE,@Spark}
59. 17/05/16 14:26:28 INFO internal.SharedState: Warehouse path is'file:/opt/spark/spark-2.1.1-bin-hadoop2.7/spark-warehouse'.
60. 17/05/16 14:26:28 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@705202d1{/SQL,null,AVAILABLE,@Spark}
61. 17/05/16 14:26:28 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@3e58d65e{/SQL/json,null,AVAILABLE,@Spark}
62. 17/05/16 14:26:28 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@6f63c44f{/SQL/execution,null,AVAILABLE,@Spark}
63. 17/05/16 14:26:28 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@62a8fd44{/SQL/execution/json,null,AVAILABLE,@Spark}
64. 17/05/16 14:26:28 INFO handler.ContextHandler: Startedo.s.j.s.ServletContextHandler@1d035be3{/static/sql,null,AVAILABLE,@Spark}
65. 17/05/16 14:26:30 INFO spark.SparkContext: Starting job: reduce atSparkPi.scala:38
66. 17/05/16 14:26:30 INFO scheduler.DAGScheduler: Got job 0 (reduce atSparkPi.scala:38) with 2 output partitions
67. 17/05/16 14:26:30 INFO scheduler.DAGScheduler: Final stage:ResultStage 0 (reduce at SparkPi.scala:38)
68. 17/05/16 14:26:30 INFO scheduler.DAGScheduler: Parents of finalstage: List()
69. 17/05/16 14:26:30 INFO scheduler.DAGScheduler: Missing parents:List()
70. 17/05/16 14:26:30 INFO scheduler.DAGScheduler: SubmittingResultStage 0 (MapPartitionsRDD[1] at map at SparkPi.scala:34), which has nomissing parents
71. 17/05/16 14:26:30 INFO memory.MemoryStore: Block broadcast\_0 storedas values in memory (estimated size 1832.0 B, free 413.9 MB)
72. 17/05/16 14:26:30 INFO memory.MemoryStore: Block broadcast\_0\_piece0stored as bytes in memory (estimated size 1167.0 B, free 413.9 MB)
73. 17/05/16 14:26:31 INFO storage.BlockManagerInfo: Addedbroadcast\_0\_piece0 in memory on 192.168.27.143:41104 (size: 1167.0 B, free:413.9 MB)
74. 17/05/16 14:26:31 INFO spark.SparkContext: Created broadcast 0 frombroadcast at DAGScheduler.scala:996
75. 17/05/16 14:26:31 INFO scheduler.DAGScheduler: Submitting 2 missingtasks from ResultStage 0 (MapPartitionsRDD[1] at map at SparkPi.scala:34)
76. 17/05/16 14:26:31 INFO scheduler.TaskSchedulerImpl: Adding task set0.0 with 2 tasks
77. 17/05/16 14:26:31 INFO scheduler.TaskSetManager: Starting task 0.0in stage 0.0 (TID 0, localhost, executor driver, partition 0, PROCESS\_LOCAL,6026 bytes)
78. 17/05/16 14:26:31 INFO executor.Executor: Running task 0.0 in stage0.0 (TID 0)
79. 17/05/16 14:26:31 INFO executor.Executor: Fetchingspark://192.168.27.143:40855/jars/spark-examples\_2.11-2.1.1.jar with timestamp1494915987472
80. 17/05/16 14:26:31 INFO client.TransportClientFactory: Successfullycreated connection to /192.168.27.143:40855 after 145 ms (0 ms spent inbootstraps)
81. 17/05/16 14:26:31 INFO util.Utils: Fetchingspark://192.168.27.143:40855/jars/spark-examples\_2.11-2.1.1.jar to/tmp/spark-702c8654-489f-47f2-85e0-8b658ebb2988/userFiles-0a07fa86-4d14-4939-ad2b-95ac8488e187/fetchFileTemp3302336691796081023.tmp
82. 17/05/16 14:26:33 INFO executor.Executor: Addingfile:/tmp/spark-702c8654-489f-47f2-85e0-8b658ebb2988/userFiles-0a07fa86-4d14-4939-ad2b-95ac8488e187/spark-examples\_2.11-2.1.1.jarto class loader
83. 17/05/16 14:26:34 INFO executor.Executor: Finished task 0.0 in stage0.0 (TID 0). 1114 bytes result sent to driver
84. 17/05/16 14:26:34 INFO scheduler.TaskSetManager: Starting task 1.0in stage 0.0 (TID 1, localhost, executor driver, partition 1, PROCESS\_LOCAL,6026 bytes)
85. 17/05/16 14:26:34 INFO executor.Executor: Running task 1.0 in stage0.0 (TID 1)
86. 17/05/16 14:26:34 INFO scheduler.TaskSetManager: Finished task 0.0in stage 0.0 (TID 0) in 2815 ms on localhost (executor driver) (1/2)
87. 17/05/16 14:26:34 INFO executor.Executor: Finished task 1.0 in stage0.0 (TID 1). 1114 bytes result sent to driver
88. 17/05/16 14:26:34 INFO scheduler.TaskSetManager: Finished task 1.0in stage 0.0 (TID 1) in 416 ms on localhost (executor driver) (2/2)
89. 17/05/16 14:26:34 INFO scheduler.TaskSchedulerImpl: Removed TaskSet0.0, whose tasks have all completed, from pool
90. 17/05/16 14:26:34 INFO scheduler.DAGScheduler: ResultStage 0 (reduceat SparkPi.scala:38) finished in 3.269 s
91. 17/05/16 14:26:34 INFO scheduler.DAGScheduler: Job 0 finished:reduce at SparkPi.scala:38, took 4.404894 s
92. Pi is roughly 3.1434157170785855
93. 17/05/16 14:26:34 INFO server.ServerConnector: StoppedSpark@32fe9d0a{HTTP/1.1}{0.0.0.0:4040}
94. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@290b1b2e{/stages/stage/kill,null,UNAVAILABLE,@Spark}
95. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@1fc0053e{/jobs/job/kill,null,UNAVAILABLE,@Spark}
96. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@77307458{/api,null,UNAVAILABLE,@Spark}
97. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@389adf1d{/,null,UNAVAILABLE,@Spark}
98. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@7bf9b098{/static,null,UNAVAILABLE,@Spark}
99. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@72e34f77{/executors/threadDump/json,null,UNAVAILABLE,@Spark}
100. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@6e9319f{/executors/threadDump,null,UNAVAILABLE,@Spark}
101. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@6fa590ba{/executors/json,null,UNAVAILABLE,@Spark}
102. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@2416a51{/executors,null,UNAVAILABLE,@Spark}
103. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@293bb8a5{/environment/json,null,UNAVAILABLE,@Spark}
104. 17/05/16 14:26:34 INFO handler.ContextHandler: Stopped o.s.j.s.ServletContextHandler@37ebc9d8{/environment,null,UNAVAILABLE,@Spark}
105. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@5217f3d0{/storage/rdd/json,null,UNAVAILABLE,@Spark}
106. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@32232e55{/storage/rdd,null,UNAVAILABLE,@Spark}
107. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@54361a9{/storage/json,null,UNAVAILABLE,@Spark}
108. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@549621f3{/storage,null,UNAVAILABLE,@Spark}
109. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@332a7fce{/stages/pool/json,null,UNAVAILABLE,@Spark}
110. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@3c9168dc{/stages/pool,null,UNAVAILABLE,@Spark}
111. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@72f46e16{/stages/stage/json,null,UNAVAILABLE,@Spark}
112. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@4985cbcb{/stages/stage,null,UNAVAILABLE,@Spark}
113. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@71104a4{/stages/json,null,UNAVAILABLE,@Spark}
114. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@5af28b27{/stages,null,UNAVAILABLE,@Spark}
115. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@4a3e3e8b{/jobs/job/json,null,UNAVAILABLE,@Spark}
116. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@7876d598{/jobs/job,null,UNAVAILABLE,@Spark}
117. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@15a902e7{/jobs/json,null,UNAVAILABLE,@Spark}
118. 17/05/16 14:26:34 INFO handler.ContextHandler: Stoppedo.s.j.s.ServletContextHandler@5118388b{/jobs,null,UNAVAILABLE,@Spark}
119. 17/05/16 14:26:34 INFO ui.SparkUI: Stopped Spark web UI athttp://192.168.27.143:4040
120. 17/05/16 14:26:34 INFO spark.MapOutputTrackerMasterEndpoint:MapOutputTrackerMasterEndpoint stopped!
121. 17/05/16 14:26:34 INFO memory.MemoryStore: MemoryStore cleared
122. 17/05/16 14:26:34 INFO storage.BlockManager: BlockManager stopped
123. 17/05/16 14:26:34 INFO storage.BlockManagerMaster: BlockManagerMasterstopped
124. 17/05/16 14:26:34 INFOscheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:OutputCommitCoordinator stopped!
125. 17/05/16 14:26:34 INFO spark.SparkContext: Successfully stoppedSparkContext
126. 17/05/16 14:26:34 INFO util.ShutdownHookManager: Shutdown hookcalled
127. 17/05/16 14:26:34 INFO util.ShutdownHookManager: Deleting directory/tmp/spark-702c8654-489f-47f2-85e0-8b658ebb2988
128. [root@hserver1 spark-2.1.1-bin-hadoop2.7]#

**注意：上面只是使用了单机本地模式调用Demo，使用集群模式运行Demo，请参考该博文**

[**http://blog.csdn.net/pucao\_cug/article/details/72453382**](http://blog.csdn.net/pucao_cug/article/details/72453382)