## 准备

Hadoop各个版本下载 : https://archive.apache.org/dist/hadoop/core/

Linux system：Redhat 5u4

Hadoop 2.6.0 hadoop-2.6.0.tar.gz 2014-11-30 23:52 186M

https://archive.apache.org/dist/hadoop/core/hadoop-2.6.0/

Java version 1.7.0\_79 64bit

Scala version 2.10.5

Hadoop 2.6.0

Spark 2.0.0-bin-hadoop2.6

Python 2.7.12

1. **开机启动图形界面与shell文本界面默认修改**

centos6 是通过修改 /ect/inittab 文件将 id:3:initdefault

centos7 开启启动init 3（dos） 5（win）是通过软连接的方式实现的：

ln -sf /lib/systemd/system/runlevel**3**.target /etc/systemd/system/default.target

1. **创建一个非root用户，今后用这个用户user登录并操作相关内容**

adduser user1

passwd 12345678

chmod -v u+w /etc/sudoers //只读的权限，需要先添加w权限

chmod -v u-w /etc/sudoers //权限收回

1. **修改/etc/sudoers 文件内容在：**

root ALL=(ALL) ALL

**master** ALL=(ALL) ALL /添加这行，赋值给你master用户sudo权利

1. **修改网络文件配置项：vi /etc/sysconfig/network-scripts/ifcfg-eth0 ifcfg-ens33.**

**BOOTPROTO=static**

**DNS1=8.8.8.8**

**IPADDR=192.168.1.210**

**PREFIX=24**

**GATEWAY=192.168.1.1**

**ONBOOT=yes //modify :yes**

**如果克隆后的系统，需要重新配置一下默认的网卡NAT，并重新设置mac地址，使之对应的网卡名称，例如eth0保持一致**

1. /etc/udev/rules.d/ 70-persistent-net.rules
2. /etc/sysconfig/network-scripts/ifcfg-eth0

## 配置

### 设置hostname

**$ Vi /etc/sysconfig/network**

NETWORKING=yes

NETWORKING\_IPV6=no

HOSTNAME=master/slave

### 配置主机与IP地址关系,master与slave都需要设置

**$ Vi /etc/hosts**

192.168.2.10 master

192.168.2.11 slave

### 启动FTP协议

**$ /etc/init.d/vsftpd restart**

### 上传并安装JDK文件

通过filezilla 第三方软件上传JDK文件jdk-7-linux-x64.rpm

**$ rpm -ivh jdk-7-linux-x64.rpm**

Preparing... ########################################### [100%]

1:jdk ########################################### [100%]

Unpacking JAR files...

rt.jar...

jsse.jar...

charsets.jar...

tools.jar...

localedata.jar...

**$ javac -version #查看就java的版本信息**

javac 1.7.0

### 创建hadoop用户与密码(hadoop)

**$ useradd Hadoop**

**$ passwd Hadoop**

### 配置ssh

Master 主机上和slave从机上都运行生成key

**$ ssh-keygen -t rsa**

-rw------- 1 hadoop hadoop 1675 03-13 23:37 id\_rsa

-rw-r--r-- 1 hadoop hadoop 394 03-13 23:37 id\_rsa.pub

#### Master主机上运行：

**$ cat id\_rsa.pub > authorized\_keys**

**#将公钥写入authorized\_keys**

**$ scp authorized\_keys slave@slave ~/.ssh //用户名@计算机名/IP**

**#远程发送uthorized\_keys到slave指定路径下**

#### Slave从机上运行：

**$ cat id\_rsa.pub >> authorized\_keys**

**#将从机的公钥写入authorized\_keys**

**$ scp authorized\_keys** [master@master:~/.ssh/](mailto:master@master:~/.ssh/) **//用户名@计算机名/IP**

**#远程发送uthorized\_keys到master指定路径下**

修改authorized\_keys权限

**$ chmod 600 authorized\_keys**

**注意：当不是root用户 远程ssh访问其他机子的时候，**

**格式是：ssh username@ servername**

**如果直接想 ssh username 则需要完成一下方式**

**在.ssh文件夹下创建 config 文件 内容如下**

**$ vi ~/.ssh/config**

Host servername

User username

Hostname serverIP

Host ubuntu

User michael

Hostname 192.168.0.222

**$ chmod 600 config**

**$ ssh ubuntu**

### 上传下载好的hadoop2.6.0并解压

上传hadoop-2.6.0.tar.gz至/home/Hadoop/installer/

**$ tar -zxvf hadoop-2.6.0.tar.gz #解压缩**

**$ mv hadoop-2.6.0 hadoop #重命名文件夹**

### 配置环境变量

**$ vi .bashrc**

export JAVA\_HOME=/usr/java/jdk1.7.0

export HADOOP\_HOME=/home/hadoop/installer/hadoop/

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

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

export CLASSPATH=$CLASSPATH:$HADOOP\_HOME/lib:$JAVA\_HOME/lib

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

在master主机下运行：

**$ . .bashrc #是环境变量生效**

**$ scp .bashrc slave:~ #远程上传.bashrc至slave的hadoop的根目录下**

在slave从机下运行：

**$ . .bashrc #是环境变量生效**

### 安装hadoop与配置

/home/hadoop/installer/hadoop/etc/hadoop/ 路径下

**$ vi hadoop-env.sh**

export JAVA\_HOME=${JAVA\_HOME}

export JAVA\_HOME=/usr/java/jdk1.7.0

export HADOOP\_HEAPSIZE=100

**$ vi yarn-env.sh**

export JAVA\_HOME=/usr/java/jdk1.7.0

JAVA\_HEAP\_MAX=-Xmx300m

YARN\_HEAPSIZE=100

**$ vi mapred-env.sh**

export JAVA\_HOME=/usr/java/jdk1.7.0

export HADOOP\_JOB\_HISTORYSERVER\_HEAPSIZE=100

**$ vi slaves**

slave

**$ vi core-site.xml**

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://master:9000</value>

</property>

<property>

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

<value>/home/hadoop/data</value>

</property>

</ configuration>

创建tmp目录

**[hadoop@master ~]$ mkdir tmp**

**[hadoop@slave ~]$ mkdir tmp**

**$ vi hdfs-site.xml**

<property>

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

<value>master:50090 </value>

</property>

<property>

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

<value>/home/hadoop/data/dfs/name</value>

</property>

<property>

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

<value>/home/hadoop/data/dfs/data</value>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.webhdfs.enabled</name>

<value>true </value>

</property>

创建目录

**[Hadoop@master ~]$ mkdir -p data/dfs/data**

**[Hadoop@master ~]$ mkdir -p data/dfs/name**

**[Hadoop@slave ~]$ mkdir -p data/dfs/data**

**[Hadoop@slave ~]$ mkdir -p data/dfs/name**

**[hadoop@master hadoop]$ cp mapred-site.xml.template mapred-site.xml**

**$ vi mapred-site.xml**

<property>

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

<value>yarn </value>

</property>

<property>

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

<value>master:10020 </value>

</property>

<property>

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

<value>master:19888 </value>

</property>

<property>

<name>mapreduce.map.memory.mb</name>

<value>300 </value>

</property>

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>300 </value>

</property>

<property>

<name>yarn.app.mapreduce.am.resource.mb</name>

<value>100 </value>

</property>

**$ vi yarn-site.xml**

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

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

<value>master</value>

</property>

<property>

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

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

</property>

<property>

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

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

</property>

<property>

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

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

</property>

<property>

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

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

</property>

<property>

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

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

</property>

<property>

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

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

</property>

<property>

<name>yarn.scheduler.maximum-allocation-mb </name>

<value>1024</value>

</property>

<property>

<name>yarn.nodemanager.vmem-pmem-ratio</name>

<value>6.1</value>

</property>

**[hadoop@master installer]$ scp -r hadoop slave:~/installer/**

**#将主机上的hadoop文件夹传给 slave ~/installer/目录**

在master主机上运行：

**[hadoop@master ~]$ . .bashrc**

**[hadoop@master ~]$ javac -version**

javac 1.7.0

在slave从机上运行

**[hadoop@slave ~]$ . .bashrc**

**[hadoop@slave ~]$ javac -version**

javac 1.7.0

**$ hadoop namenode -format #格式化hdfs文集系统**

**$ start-dfs.sh**

**$ start-yarn.sh**

### 安装spark与配置

安装scala环境 下载地址：<http://www.scla-lang.org/download/>

安装Python ：Python-2.7.12.tgz

上传scala-2.10.5.tgz到master和slave机器的hadoop用户installer目录下

两台机器都有做

/home/hadoop/installer/scala

配置环境变量

$ vi .bashrc

export JAVA\_HOME=/usr/java/jdk1.7.0

export HADOOP\_HOME=/home/hadoop/installer/hadoop

export SCALA\_HOME=/home/hadoop/installer/scala

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

export CLASSPATH=$CLASSPATH:$HADOOP\_HOME/lib:$JAVA\_HOME/lib:$SCALA\_HOME/lib

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

安装Python

安装gcc

$ mkdir /RHEL5U4

$ mount /dev/cdrom /media/ #将cdrom挂载到media下

$ cp -r \* /RHEL5U4

$ vi /etc/yum.repos.d/iso.repo

Name=5u4\_server

Baseurl=file:///RHEL5U4/Server

Enable=1

Gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

$ yum clean all

$ yum install gcc

**[hadoop@master installer]$ tar -zxvf Python-2.7.12**

**[hadoop@master Python-2.7.12]$ ./configure—prefix=/usr/local/python27**

**[hadoop@master Python-2.7.12]$ make**

**[hadoop@master Python-2.7.12]$ make install**

**[hadoop@master Python-2.7.12]$ mv /usr/bin/python /usr/bin/python\_old**

**#**替换系统老版本的Python

**[hadoop@master Python-2.7.12]$ In -s /usr/local/python27/bin/python /usr/bin/**

#生成软链接

**[hadoop@master Python-2.7.12]$ python**

安装spark环境

下载地址：http;//spark.apache.org/downloads.html

上传 spark-2.0.0-bin-hadoop2.6.tgz到master的hadoop用户installer目录下解压缩

$ tar -zxvf spark-2.0.0-bin-hadoop2.6.tgz

$ mv spark-2.0.0-bin-hadoop2.6 spark

$ cd spark

$ pwd

/home/hadoop/installer/spark/

$ vi .bashrc

export JAVA\_HOME=/usr/java/jdk

export HADOOP\_HOME=/home/hadoop/installer/hadoop

export SCALA\_HOME=/home/hadoop/installer/scala

export SPARK\_HOME=/home/hadoop/installer/spark

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

export CLASSPATH=$CLASSPATH:$HADOOP\_HOME/lib:$JAVA\_HOME/lib:$SCALA\_HOME/lib:$SPARK\_HOME/jars

export PATH=$PATH:$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin:$SCALA\_HOME/bin:$SPARK\_HOME/bin: $SPARK\_HOME/sbin

进入到：/home/hadoop/installer/spark/conf

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

**$ vi spark-env.sh**

export JAVA\_HOME=/usr/java/jdk1.7.0

export SCALA\_HOME=/home/hadoop/installer/scala

export SPARK\_MASTER\_HOST=master

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

export SPARK\_EXECUTOR\_MOMORY=600M

export SPARK\_DRIVER\_MEMORY=600M

**[hadoop@master conf]$ cp slaves.template slaves**

**$ vi slaves #让主节点和从节点都有worker进程**

master

slave

**[hadoop@master installer]$ scp -r spark slave:~/installer/ #远程发送slave**

**启动spark集群**

**$ start-master.sh #主机启动 master worker**

**$ start-slaves.sh #主机启动 worker**

**$ start-dfs.sh #主机启动 namenode datanode secondarynamenode**

**$ start-yarn.sh #从机启动 datanode**

**$ spark-shell #启动spark**

**eg**

**scala> val file = sc.textFile(“hdfs://master:9000/data/input/wordcount/.1txt”)**

**scala> val count=file.flatmap(line =>line.split(“”)).map(word,1)).reduceBykey(\_+\_)**

**scala> count.collect()**

**http:192.168.2.10:8080 #访问spark网页**

## RDD

RDD 是个抽象累，全称是resilient distributed datasets ,是一个容错的，并行的数据结构，可以让用户显式的将数据存储到磁盘和内存中，并能控制数据的分区。同事，RDD还提供了一组丰富的操作来操作这些数据，诸如map flatmap filter 等转换操作。除此之外，RDD还提供了诸如：join groupBy reduceBykey等更为方便的操作，以支持常见的数据运算。但实际上继承RDD的派生类一般只要实现两个方法：

1. getPartitions（）用来告知这么将input分片；
2. conpute（）用来输入每个partition被函数处理的一个单元；

RDD的特点

1. 它是在集群节点上的不可变的、已分区的集合对象
2. 通过并行转换的方式来创建（map,filiter,join）
3. 失败自动重建。
4. 可以控制存储级别（内存、磁盘等）来进行重用。
5. 必须是可序列话的。

RDD的好处

1. RDD只能从持久存储或者通过transformation操作产生，相比于分布式共享内存（DSM）可以更高效实现容错，对于丢失部分数据分区只需要根据它的lineage就可以重新计算出来，而不需要做特定的checkpoint。（RDD实现了基于lineage的容错机制。RDD的转换关系，构成了compute chain，可以把这个compute chain 认为是RDD之间演化的lineage。在部分计算结果丢失时，只需要根据这个lineage重算即可）
2. RDD的不变性，可以实现类似hadoop mapreduce的推测式执行。
3. RDD的数据分区特性，可以通过数据的本地性来提高性能，这与hadoop mapreduce是一样的。
4. RDD都是可序列化的，在内存不足时可以自动降级为磁盘存储，把RDD存储于磁盘上，这时性能会有大的下降但不会差于现在的mapreduce。

RDD的存储与分区

1. 用户可以选择不同的存储级别存储RDD以便重用。
2. 当前RDD默认是存储于内存，但当内存不足时，RDD会spill到disk
3. RDD在需要时进行分区把数据分布于集群中时根据每条记录key进行分区（如Hash分区），一次保证两个数据集在join是能高效。

RDD内部表示

在RDD的内部实现中每个RDD都可以使用5个方面的特性来表示：

1. 分区列表（数据块列表）
2. 计算每个分片的函数（根据父RDD计算出此RDD）
3. 对于父RDD的依赖列表
4. 对key-value RDD的partition（可选）
5. 每个数据分片的预定义地址列表（如HDFS上的数据块的地址）（可选）

RDD的创建方

1. 从hadoop文件系统（或与hadoop兼容的其他存储系统）输入（例如HDFS）创建。
2. 从父RDD转换得到新RDD
3. 通过parallelize将单机数据创建为分布式RDD

## 安装R语言方案一：

##缺少zlib或者zlib版本过低

##yum install gcc-gfortran gcc gcc-c++ readline-devel libXt-devel

## <http://kuxingseng2016.blog.51cto.com/1374617/1846326>

##用这个 **http://blog.sina.com.cn/s/blog\_6ac25ea00102whkc.html**

### 1、yum install readline-devel gcc\*

### 2、yum install libXt-devel

### 3、tar zxvf R-3.3.3.tar.gz

### 4、安装zlib

tar xvf zlib-1.2.8.tar.gz

cd zlib-1.2.8

./configure --prefix=/opt/zlib-1.2.8

make && make install

### 5、cd /root/R-3.3.3

./configure --prefix=/opt/R-3.3.3 --enable-R-shlib LDFLAGS="-L/opt/zlib-1.2.8/lib" CPPFLAGS="-I/opt/zlib-1.2.8/include"

### 6、安装bzip2

tar zxvf bzip2-1.0.6.tar.gz

cd bzip2-1.0.6

make -f Makefile-libbz2\_so

make clean

make && make install PREFIX=/opt/bzip2-1.0.6

### 7、cd /root/R-3.3.3

./configure --prefix=/opt/R-3.3.3 --enable-R-shlib LDFLAGS="-L/opt/zlib-1.2.8/lib -L/opt/bzip2-1.0.6/lib" CPPFLAGS="-I/opt/zlib-1.2.8/include -I/opt/bzip2-1.0.6/include"

### 8、安装xz

tar zxvf xz-5.2.2.tar.gz

cd xz-5.2.2

./configure --prefix=/opt/xz-5.2.2

make -j3 && make install

### 9、cd /root/R-3.3.3

./configure --prefix=/opt/R-3.3.3 --enable-R-shlib LDFLAGS="-L/opt/zlib-1.2.8/lib -L/opt/bzip2-1.0.6/lib -L/opt/xz-5.2.2/lib" CPPFLAGS="-I/opt/zlib-1.2.8/include -I/opt/bzip2-1.0.6/include -I/opt/xz-5.2.2/include"

### 10、重新安装pcre

rm -rf /opt/pcre-8.39

cd pcre-8.39

./configure --prefix=/opt/pcre-8.39 --enable-utf8

make -j3 && make install

### 11、cd /root/R-3.3.3

./configure --prefix=/opt/R-3.3.3 --enable-R-shlib LDFLAGS="-L/opt/zlib-1.2.8/lib -L/opt/bzip2-1.0.6/lib -L/opt/xz-5.2.2/lib -L/opt/pcre-8.39/lib" CPPFLAGS="-I/opt/zlib-1.2.8/include -I/opt/bzip2-1.0.6/include -I/opt/xz-5.2.2/include -I/opt/pcre-8.39/include/"

### 12、安装curl

yum install openssl\* ##需要支持ssl服务

tar zxvf curl-7.50.1.tar.gz

cd curl-7.50.1

./configure --prefix=/opt/curl-7.50.1

make && make install

### 13、cd /root/R-3.3.3

./configure --prefix=/opt/R-3.3.3 --enable-R-shlib LDFLAGS="-L/opt/zlib-1.2.8/lib -L/opt/bzip2-1.0.6/lib -L/opt/xz-5.2.2/lib -L/opt/pcre-8.39/lib -L/opt/curl-7.50.1/lib" CPPFLAGS="-I/opt/zlib-1.2.8/include -I/opt/bzip2-1.0.6/include -I/opt/xz-5.2.2/include -I/opt/pcre-8.39/include -I/opt/curl-7.50.1/include"

### 14、最终操作

cd /root/R-3.3.3

./configure --prefix=/opt/R --enable-R-shlib LDFLAGS="-L/opt/zlib-1.2.8/lib -L/opt/bzip2-1.0.6/lib -L/opt/xz-5.2.2/lib -L/opt/pcre-8.39/lib -L/opt/curl-7.50.1/lib" CPPFLAGS="-I/opt/zlib-1.2.8/include -I/opt/bzip2-1.0.6/include -I/opt/xz-5.2.2/include -I/opt/pcre-8.39/include -I/opt/curl-7.50.1/include"

vi /etc/profile

export PATH=$PATH:$JAVA\_HOME/bin:/opt/curl-7.50.1/bin

##在这一行加上":/opt/curl-7.50.1/bin"

source /etc/profile

##让环境变量立马生效

验证；

echo $PATH

此处报错是由于没有找到动态库，看第一二行

解决方法：添加动态库

#cat /etc/ld.so.conf

include ld.so.conf.d/\*.conf

/opt/pcre-8.39/lib

/opt/xz-5.2.2/lib

然后执行

#ldconfig

make && make install

vim /etc/profile ##添加R源环境变量

export PATH=$PATH:$JAVA\_HOME/bin:/opt/curl-7.50.1/bin:/opt/R-3.3.3/bin

source /etc/profile ##让环境变量立马生效

## 安装R语言方案二

### 下载和安装bzip2：

bzip2不是标准的GNU包，根据下载的安装文件的说明文档，执行以下命令：

cd ~/src

wget http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz

tar xzvf bzip2-1.0.6.tar.gz

cd bzip2-1.0.6

make -f Makefile-libbz2\_so

make clean && make

make install PREFIX=/HOME/packages

注意：这里下载完bzip2后，需要修改Makefile文件，在CFLAGS这个变量后面添加:-fPIC，否则后面安装R的时候会报错。

### 安装xz包：

cd ~/src

wget http://tukaani.org/xz/xz-5.2.2.tar.gz

tar xzvf xz-5.2.2.tar.gz

cd xz-5.2.2

./configure --prefix=/HOME/packages

make -j3 && make install

### 安装pcre包：

cd ~/src

wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.38.tar.gz

tar xzvf pcre-8.38.tar.gz

cd pcre-8.38

./configure --enable-utf8 --prefix=/HOME/packages

make -j3 && make install

### 安装curl包：

cd ~/src

wget --no-check-certificate https://curl.haxx.se/download/curl-.47.1.tar.gz

tar xzvf curl-7.47.1.tar.gz

cd curl-7.47.1

./configure --prefix=/HOME/packages

make -j3 && make install

### 设置安装好的包的环境变量vi /etc/profile（！这步最重要）：

/etc/.bash\_profile

export PATH=/HOME/packages/bin:$PATH

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$LD\_LIBRARY\_PATH

export CFLAGS="-I/HOME/packages/include"

export LDFLAGS="-L/HOME/packages/lib"

前两个是安装R的“make”过程需要用到的，后两个是“configure”过程需要用到的。

### 最后就是安装R-3.3.0:

cd ~/src/R-3.3.0

mkdir builddir

cd builddir

../configure --prefix=/HOME/packages/R-3.3.0

make && make install

## 安装RStudio Server

1.在编译R的时候，要执行./configure --enable-R-shlib=yes

2.安装RPM时可能会报缺少libR.so的错误，忽略安装即可，

rpm -ivh --nodeps rstudio-server.rpm

Yum安装时需要安装到指定的文件夹，则需要 --installroot

yum install \*\*\*\*\*.rpm --installroot=/usr/src/ vim

ps auxw | grep studio

rpm -ivh names.rpm

1. rpm -qa|grep 软件包名

2. rpm -ql 软件包名 (l是L的小写,不是坚线) #指明具体的安装位置

RPM默认安装路径：rpm 全名是 “RedHat Package Manager"，简称则为RPM。

|  |  |
| --- | --- |
| /etc | 一些设置文件放置的目录如/etc/crontab |
| /usr/bin | 一些可执行文件 |
| /usr/lib | 一些程序使用的动态函数库 |
| /usr/share/doc | 一些基本的软件使用手册与帮助文档 |
| /usr/share/man | 一些man page文件 |

### RStudio Server 的系统配置 /usr/lib/rstudio-server

RStudio Server主要有两个配置文件，默认文件不存在

/etc/rstudio/rserver.conf

/etc/rstudio/rsession.conf

设置端口和ip控制：

~ vi /etc/rstudio/rserver.conf

www-port=8787 # 监听端口

www-address=**127.0.0.1** # 允许访问的IP地址，设置成实际本机的IP地址

rsession-which-r=/opt/R/bin/R

重启Rstudio Server服务器，配置生效：

~ sudo rstudio-server restart

会话配置管理

~ vi /etc/rstudio/rsession.conf

session-timeout-minutes=30

# 会话超时时间

r-cran-repos=http://ftp.ctex.org/mirrors/CRAN/

# 设置CRAN资源库2. RStudio Server 的系统管理

rstudio-server test-config #使参数生效

rstudio-server restart #重启rstudioserver

rstudio-server verify-installation查看安装过程中出现的问题

原因是libRblas.so文件无法加载 .so文件拷到/usr/lib

（发现，原因可能是当初指定安装路径时参数设置的问题，./configure时，不加--prefix= 参数时，没有出现该问题，

rpm -e rstudio-server卸载rstudio-server）

### 启动、停止、重启 RStudio Server服务器的命令如下：

~ sudo rstudio-server start # 启动

~ sudo rstudio-server stop # 停止

~ sudo rstudio-server restart # 重启

rpm -e rstudio-server #卸载rstudio-server

ps aux | grep rstudio-server显示

这种情况，主机ip:8787就可以登录rstudio-server

netstat -ntlp | grep 8787

查看8787端口的监控信息

查看下 /tmp/rstudio-rsession/ 的 owner，如果是 root 的话就改成 rstudio-server。

sudo chown rstudio-server:rstudio-server rstudio-rsession

## Install Rstudio desktop version

I fixed this error following this link: http://jermdemo.blogspot.com.es/2011/08/installing-rstudio-server-on-scientific.html

### You have to install some dependencies:

yum install libcrypto.so.6 -y

yum install libgfortran.so.1 -y

yum install libssl.so.6 -y

yum install openssl098e-0.9.8e -y

yum install gcc41-libgfortran-4.1.2 -y

yum install pango-1.28.1 -y

### and the tricky one:

wget ftp.scientificlinux.org/linux/scientific/6.0/x86\_64/os/Packages/compat-libgfortran-41-4.1.2-39.el6.x86\_64.rpm

rpm -Uvh compat-libgfortran-41-4.1.2-39.el6.x86\_64.rpm

and install Rstudio with no-deps flag:

rpm -Uvh --nodeps rstudio-server-0.97.336-x86\_64.rpm

rstudio-server verify-installation