

# CentOS下Hadoop+Spark集群搭建

## CentOS下Hadoop+Spark集群环境搭建

### 硬件环境

虚拟机\*3

每台虚拟机配置：系统CentOS6.5 64位，内存1g，硬盘20g。

网络地址：

- master : 172.27.35.10
- slave1 : 172.27.35.11
- slave2 : 172.27.35.12

### 软件环境

- java版本：1.8.0\_151
- hadoop版本：2.7.6
- spark版本：2.3.0
- scala版本：2.11.12

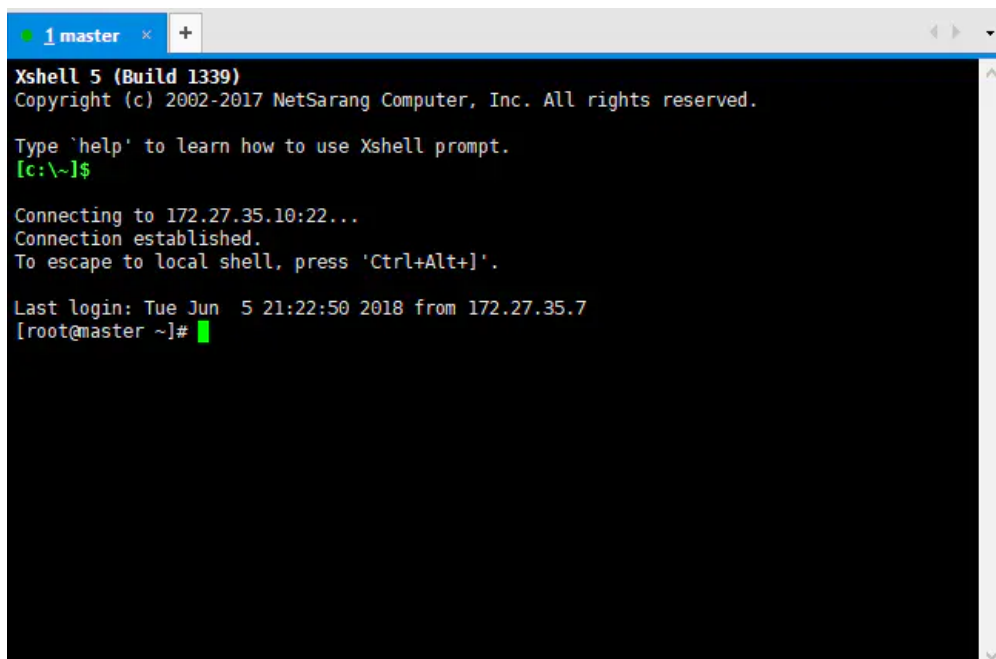
### xshell安装

下载安装[xshell](#)，使用SSH远程登录虚拟机。

### java安装

1、登录master主机

使用xshell远程登录master主机，登录成功后如下图所示：



2、检查虚拟机网络连接是否正常

可以使用ping命令来检查网络问题：

```
1 | ping www.baidu.com
```

如果ping成功，则网络没有问题。

如果ping没有成功，则输入 `ifconfig`，查看网络设置。如果显示如下图：

```
[root@master ~]# ifconfig
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:12 errors:0 dropped:0 overruns:0 frame:0
        TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:848 (848.0 b)  TX bytes:848 (848.0 b)

[root@master ~]# _
```

则说明网卡没有设置启动好，需设置网卡并启动。

修改网卡设置：

```
1 | vim /etc/sysconfig/network-scripts/ifcfg-eth0
```

修改后如图所示：

```
DEVICE=eth0
WADDR=00:0C:29:FB:20:E3
TYPE=Ethernet
UUID=0c8f006c-6830-4a9a-a030-6639234ed2b5
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=static
GATEWAY=172.27.35.1
IPADDR=172.27.35.10
NETMASK=255.255.0.0
~
~
~
~
~
```

如上图配置将master主机IP地址设置为静态地址172.27.35.10，其余主机可以参照上述操作将IP地址设置为相应静态地址。

然后配置DNS：

```
1 | vim /etc/resolv.conf
```

配置后如图所示：

```
gameserver 114.114.114.114
~
~
~
~
~
```

注意：虚拟机中设置静态IP地址时，网关、子网掩码要和宿主机一样，IP地址也要和宿主机在同一个网段，否则连不上网，桥接模式要记得选择网卡。

配置完成后输入 `service network restart` 重启网卡，便可成功连接网络。如果使用的是虚拟机，并且子节点是从其他机器克隆的话，注意修改ifcfg-eth0中的HWADDR硬件地址，并且删除/etc/udev/rules.d/70-persistent-net.rules，这个文件确定了网卡和MAC地址的信息之间的绑定，所以克隆后需删除，待机器重启后重新生成。

### 3、更新软件包

在终端程序输入以下命令来更新软件包：

```
1 | yum upgrade
```

### 4、安装java

在[oracle官网](#)下载对应的jdk，拷贝到master主节点上，这里用的版本为jdk-8u151-linux-x64.tar.gz。

输入解压缩命令：

```
1 | tar -zxvf jdk-8u151-linux-x64.tar.gz
```

将解压后文件夹重命名移动到/usr/local/java中（这里软件包都一律安装到/usr/local文件夹中）：

```
1 | mv jdk1.8.0_151/ /usr/local/java
```

### 5、配置系统变量

输入命令修改系统配置文件：

```
1 | vim /etc/profile
```

在文件末尾输入：

```
1 | export JAVA_HOME=/usr/local/java
2 | export JRE_HOME=$JAVA_HOME/jre
3 | export PATH=$PATH:$JAVA_HOME/bin
4 | export CLASSPATH=.:$JAVA_HOME/lib/tools.jar:$JAVA_HOME/lib/dt.jar:$JRE_HOME/lib
```

之后保存退出，输入 `source /etc/profile` 使配置文件生效。

### 6、查看java版本

```
1 | java -version
```

结果如图所示：

```
[root@master java]# java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
[root@master java]#
```

如上，java环境安装配置成功。

## 7、发送jdk到从节点上

(1) 修改主节点、从节点hosts文件，修改后hosts文件如下图所示：

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

172.27.35.10 master
172.27.35.11 slave1
172.27.35.12 slave2
~
~
~
```

(2) 配置各个节点ssh免密登陆

在master主节点上输入命令 `ssh-keygen -t rsa` 生成公钥，结果如图所示：

```
[root@master ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
9a:8e:bf:46:a0:c8:98:57:7d:61:ec:15:c9:2f:0c:09 root@master
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      Eo o.o      |
|      * +        |
|      . o = .     |
|      o . o o .   |
| oo o . . S .    |
|+.o .o           |
|      .o          |
|      o.          |
|      .o+.        |
+-----+

```

然后输入命令将公钥发送到各个子节点上：

```
1 | ssh-copy-id -i ~/.ssh/id_rsa.pub root@slave1
```

结果如图所示：

```
[root@master ~]# ssh-copy-id -i ~/.ssh/id_rsa.pub root@slave1
The authenticity of host 'slave1 (172.27.35.11)' can't be established.
RSA key fingerprint is 32:4d:38:67:81:62:7c:l0:ec:70:l4:5c:f4:c4:64:3e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave1,172.27.35.11' (RSA) to the list of known hosts.
root@slave1's password:
Now try logging into the machine, with "ssh 'root@slave1'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
[root@master ~]#
```

上图所示只是将公钥从master主节点发送到slave1从节点的authorized\_keys列表，发送到其他从节点只需改变主机名就可以了。

输入 `ssh slave1` 验证是否主节点到从节点免密登陆，结果如图所示：

```
[root@master ~]# ssh slave1
Last login: Tue Jun  5 19:44:58 2018 from 172.27.35.7
[root@slave1 ~]#
```

说明主节点到slave1从节点免密登陆配置成功。

注意：除了配置主节点到各个子节点间免密登陆，我们最好也配置各个子节点到主节点间以及各个子节点间免密登陆，在需要配置到其他节点间免密登陆的主机上按照如上方法操作即可。

(3) 发送jdk到从节点

输入命令 `scp -r /usr/local/java/ root@slave1:/usr/local/java/`，将jdk发送到slave1从节点上，如下图所示：



发送到其他子节点只需修改目标主机名即可。

(4) 配置各个从节点系统变量

参照第5步所示方法。最后输入 `java -version` 验证配置是否成功。

# Hadoop安装

## 1、安装Hadoop

到[Hadoop官网](#)下载Hadoop安装包，拷贝到主节点上，这里用的版本为hadoop-2.7.6.tar.gz。

输入解压缩命令：

```
1 | tar -zxvf hadoop-2.7.6.tar.gz
```

将解压后文件夹重命名移动到/usr/local/hadoop中：

```
1 | mv hadoop-2.7.6 /usr/local/hadoop
```

## 2、配置系统环境变量

输入命令 `vim /etc/profile`，在文件末尾输入如下内容：

```
1 | export HADOOP_HOME=/usr/local/hadoop
2 | export PATH=$PATH:$HADOOP_HOME/bin
3 | export PATH=$PATH:$HADOOP_HOME/sbin
4 | export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

之后保存退出，输入 `source /etc/profile` 使配置文件生效。

## 3、hadoop相关文件配置

hadoop配置文件所在目录为\$HADOOP\_HOME/etc/hadoop，此处HADOOP\_HOME为hadoop安装目录，进入hadoop配置文件所在目录，修改相应配置文件。

#### (1)hadoop-env.sh文件配置

修改JAVA\_HOME为当前jdk安装目录：

```
1 | export JAVA_HOME=/usr/local/java
```

#### (2)core-site.xml文件配置如下

```
1 | <configuration>
2 | <property>
3 | <name>fs.default.name</name>
4 | <value>hdfs://master:9000</value>
5 | </property>
6 | </configuration>
```

#### (3)hdfs-site.xml文件配置如下

```
1 | <configuration>
2 | <property>
3 | <name>dfs.replication</name>
4 | <value>3</value>
5 | </property>
6 |
7 | <property>
8 | <name>dfs.namenode.name.dir</name>
9 | <value>file:/usr/local/hadoop/hdfs/namenode</value>
10 | </property>
11 |
12 | <property>
13 | <name>dfs.datanode.data.dir</name>
14 | <value>file:/usr/local/hadoop/hdfs/datanode</value>
15 | </property>
16 | </configuration>
```

#### (4)slaves文件配置如下

```
1 | slave1
2 | slave2
```

因为我们没有用到hadoop的yarn与mapreduce，所以hadoop相关配置到此结束。

#### 4、发送hadoop安装包到各个从节点

输入命令 `scp -r /usr/local/hadoop/ root@slave1:/usr/local/hadoop`，将hadoop安装包发送到slave1节点，发送的其他节点只需修改相应主机名即可。

然后修改对应从节点系统变量，方法参照第2步。

#### 5、格式化namenode

在master主节点输入命令 `hadoop namenode -format` 格式化namenode，如下图所示：

```
[root@master local]# hadoop namenode -format  
DEPRECATED: Use of this script to execute hdfs command is deprecated.
```

## 6、启动hdfs

在master主节点输入命令 `start-dfs.sh` ,启动hdfs，如下图所示：

```
[root@master ~]# start-dfs.sh  
Starting namenodes on [master]  
root@master's password:  
master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-master.out  
slave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-slave1.out  
slave2: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-slave2.out  
Starting secondary namenodes [0.0.0.0]  
root@0.0.0.0's password:  
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-root-secondarynamenode-master.out  
[root@master ~]#
```

## 7、检查hdfs是否启动成功

在主节点输入 `jps` ,查看已启动的java进程，如下图所示，显示namenode、secondaryNameNode启动成功：

```
[root@master ~]# jps  
2496 Jps  
2370 SecondaryNameNode  
2189 NameNode  
[root@master ~]#
```

分别进入各个从节点，查看datanode是否启动成功，如下图所示，则datanode启动成功：

```
[root@slave1 hadoop]# jps  
13876 Jps  
13803 DataNode  
[root@slave1 hadoop]#
```

## 8、hdfs管理界面进入

在地址栏输入`http://172.27.35.10:50070`，此处172.27.35.10为namenode主机ip，尝试进入hdfs管理界面，如果无法进入，一般是防火墙的问题，可以输入命令 `service iptables stop` 关闭防火墙，也可以进一步输入命令 `chkconfig iptables off` 关闭防火墙开机自启动，为了集群的顺利运行，可以把集群中的机器防火墙都关闭掉。成功进入hdfs管理界面如下图所示：

HadoopOverviewDatanodesDatanode Volume FailuresSnapshotStartup ProgressUtilities

Overview 'master:9000' (active)

Started:	Tue Jun 05 21:35:01 CST 2018
Version:	2.7.6, r085099c6c128be31604560c376fa282e95282b8
Compiled:	2018-04-18T01:33Z by kshvachk from branch-2.7.6
Cluster ID:	CID-32031c8f-fa5d-42f0-ac09-2e4b3caa6f38
Block Pool ID:	BP-1420876827-172.27.35.10-1528205085993

Summary

Security is off.  
Safemode is off.  
1 files and directories, 0 blocks = 1 total filesystem object(s).  
Heap Memory used 39.6 MB of 67.33 MB Heap Memory. Max Heap Memory is 966.69 MB.  
Non Heap Memory used 47.84 MB of 48.67 MB Committed Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	34.46 GB
DFS Used:	48 KB (0%)
Non DFS Used:	3.44 GB
DFS Remaining:	29.26 GB (84.91%)
Block Pool Used:	48 KB (0%)
DataNodes usage% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	2 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0

# scala安装

## 1、安装scala

在安装Spark之前，我们需要先安装scala，到[scala官网](#)下载scala，拷贝到主节点中，此处安装版本为scala-2.11.12.tgz。

输入解压缩命令：

```
1 | tar -zxvf scala-2.11.12.tgz
```

将解压后文件夹重命名移动到/usr/local/scala中：

```
1 | mv scala-2.11.12 /usr/local/scala
```

## 2、配置系统环境变量

输入命令 `vim /etc/profile`，在文件末尾添加如下内容：

```
1 | export SCALA_HOME=/usr/local/scala
2 | export PATH=$PATH:$SCALA_HOME/bin
```

之后保存退出，输入 `source /etc/profile` 使配置文件生效。输入 `scala -version` 验证安装是否成功，如下图所示：

```
[root@master ~]# scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
```

## 3、发送scala到从节点

输入命令 `scp -r /usr/local/scala/ root@slave1:/usr/local/scala`，将scala发送到slave1节点，发送到其他节点只需修改相应主机名即可。同时修改系统环境变量，参照第2步。

# Spark安装

## 1、安装Spark

到[Spark官网](#)下载Spark，拷贝到主节点中，此处安装版本为spark-2.3.0-bin-hadoop2.7.tgz。解压缩并将解压后文件夹重命名移动到/usr/local/spark中。

```
1 | tar -zxvf spark-2.3.0-bin-hadoop2.7.tgz
2 | mv spark-2.3.0-bin-hadoop2.7 /usr/local/spark
```

## 2、配置系统环境变量

输入命令 `vim /etc/profile`，在文件末尾添加如下内容：

```
1 | export SPARK_HOME=/usr/local/spark
2 | export PATH=$PATH:$SPARK_HOME/bin
```

## 3、spark相关文件配置

spark相关配置文件都在\$SPARK\_HOME/conf文件夹目录下，此处SPARK\_HOME为Spark安装目录，进入Spark配置文件所在目录，修改相应配置文件。

(1)spark-env.sh文件配置



拷贝spark-env.sh.template到spark-env.sh，命令如下：

```
1 | cp spark-env.sh.template spark-env.sh
```

spark-env.sh文件配置如下：

```
1 | export JAVA_HOME=/usr/local/java
2 | export SPARK_MASTER_IP=master
3 | export SPARK_WORKER_CORES=1
4 | export SPARK_WORKER_MEMORY=1g
5 | export SPARK_WORKER_INSTANCES=2
6 | export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=18080 -Dspark.history.retainedApplications=3 -
7 | export SPARK_DAEMON_JAVA_OPTS="-Dspark.deploy.recoveryMode=FILESYSTEM -Dspark.deploy.recoveryDire
```

注意：此处历史服务器日志存放地址为<hdfs://master:9000/historyServerForSpark/logs>，在启动历史服务器前一定要确保该文件夹存在，

可以输入 `hadoop fs -mkdir -p /historyServerForSpark/logs` 来创建该文件夹。

(2)slaves文件配置如下

拷贝slaves.template到slaves，命令如下：

```
1 | cp slaves.template slaves
```

slaves文件配置如下:

```
1 | slave1
2 | slave2
```

(3)spark-defaults.conf文件配置

拷贝spark-defaults.con.template到spark-defaults.conf，命令如下

```
1 | cp spark-defaults.conf.template spark-defaults.conf
```

spark-defaults.conf文件配置如下：

```
1 | spark.eventLog.enabled=true
2 | spark.eventLog.dir=hdfs://master:9000/historyServerForSpark/logs
3 | spark.eventLog.compress=true
```

此处主要是历史服务器相关配置。

4、发送spark安装包到各个从节点

输入命令 `scp -r /usr/local/spark/ root@slave1:/usr/local/spark`，将spark发送到slave节点，发送到其他节点只需修改对应主机名就行。同时修改系统环境变量，参照第2步。

5、启动spark集群

进入SPARK\_HOME/sbin目录，输入命令 `./start-all.sh`，结果如下图所示：

```
[root@master sbin]# ./start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/local/spark/logs/spark-root-org.apache.spark.deploy.master.Master-1-master.out
slave2: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-slave2.out
slave1: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-slave1.out
```

## 6、启动历史服务器

首先确保历史服务器日志存放文件夹已创建，然后进入SPARK\_HOME/sbin目录，输入命令 `./start-history-server.sh`，结果如下图所示：

```
[root@master sbin]# ./start-history-server.sh
starting org.apache.spark.deploy.history.HistoryServer, logging to /usr/local/spark/logs/spark-root-org.apache.spark.deploy.history.HistoryServer-1-master.out
[root@master sbin]#
```

## 7、检查spark集群、历史服务器是否启动成功

在主节点输入 `jps`，查看已启动的java进程，如下图所示，显示master、historyserver启动成功：


```
[root@master sbin]# jps
2370 SecondaryNameNode
3250 Jps
3207 HistoryServer
2906 Master
2189 NameNode
[root@master sbin]#
```

分别进入各个子节点，查看worker是否启动成功，如下图所示，则worker启动成功：

```
[root@slavel conf]# jps
14196 Jps
14122 Worker
13803 DataNode
[root@slavel conf]#
```

## 8、进入集群管理、历史服务器管理页面

在浏览器地址栏输入地址 `http://172.27.35.10:8080`，此处172.27.35.10为master ip地址，进入集群管理界面，成功进入如下图所示：

 **Spark Master at spark://master:7077**

URL: spark://master:7077  
REST URL: spark://master:6066 (cluster mode)

Active Workers: 2  
Cores in use: 2 Total, 0 Used  
Memory in use: 2.0 GB Total, 0.0 B Used  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

**Workers (2)**

Worker Id	Address	State	Cores	Memory
worker-20180605231605-172.27.35.11-49531	172.27.35.11:49531	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20180605231605-172.27.35.12-35053	172.27.35.12:35053	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)


**Running Applications (0)**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

**Completed Applications (0)**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

在浏览器地址栏输入地址 `http://172.27.35.10:18080`，进入历史服务器管理界面，成功进入如下图所示：

 **History Server**

Event log directory: hdfs://master:9000/historyServerForSparklogs  
Last updated: 2018-06-05 23:09:23  
Client local time zone: Asia/Shanghai

**No completed applications found!**

Did you specify the correct logging directory? Please verify your setting of spark.history.fs.logDirectory listed above and whether you have the permissions to access it.  
It is also possible that your application did not run to completion or did not stop the SparkContext.

[Show incomplete applications](#)


因为我们还没有跑过程序，所以历史服务器里记录为空。

## 9、集群测试

下面我们尝试在spark集群中跑个简单的测试程序，进入目录 `$SPARK_HOME/bin`，此处 `SPARK_HOME` 为spark安装目录，输入如下命令：

```
1 ./bin/spark-submit \  
2   --class org.apache.spark.examples.SparkPi \  
3   --master spark://master:6066 \  
4   --deploy-mode cluster \  
5   --supervise \  
6
```

```
[root@elavl bin]# ./bin/spark-submit \
> --class org.apache.spark.examples.SparkPi \
> --master spark://master:6066 \
> --deploy-mode cluster \
> --supervise \
> --executor-memory 1G \
> --total-executor-cores 2 \
> ./examples/jars/spark-examples_2.11-2.3.0.jar \
> 10000
[root@elavl bin]# ./spark-submit \
> --class org.apache.spark.examples.SparkPi \
> --master spark://master:6066 \
> --deploy-mode cluster \
> --supervise \
> --executor-memory 1G \
> --total-executor-cores 2 \
> ./examples/jars/spark-examples_2.11-2.3.0.jar \
> 1000
Running Spark using the REST application submission protocol.
2018-06-05 23:50:18 INFO RestSubmissionClient:54 - Submitting a request to launch an application in spark://master:6066.
2018-06-05 23:50:20 INFO RestSubmissionClient:54 - Submission successfully created as driver-20180605231925-0000. Polling submission state...
2018-06-05 23:50:20 INFO RestSubmissionClient:54 - Submitting a request for the status of submission driver-20180605231925-0000 in spark://master:6066.
2018-06-05 23:50:20 INFO RestSubmissionClient:54 - State of driver driver-20180605231925-0000 is now RUNNING.
2018-06-05 23:50:20 INFO RestSubmissionClient:54 - Driver is running on worker worker-20180605231605-172.27.35.12-35053 at 172.27.35.12:35053.
2018-06-05 23:50:20 INFO RestSubmissionClient:54 - Server responded with CreateSubmissionResponse:
{
  "action": "CreateSubmissionResponse",
  "message": "Driver successfully submitted as driver-20180605231925-0000",
  "serverSparkVersion": "2.3.0",
  "submissionId": "driver-20180605231925-0000",
  "success": true
}
2018-06-05 23:50:20 INFO ShutdownHookManager:54 - Shutdown hook called
2018-06-05 23:50:20 INFO ShutdownHookManager:54 - Deleting directory /tmp/spark-0f307314-a98e-4f5c-8b28-6285cf480b5b
[root@elavl bin]#
```



Spark Master at spark://master:7077

URL: spark://master:7077

REST URL: spark://master:6066 (cluster mode)

Alive Workers: 2

Cores in use: 2 Total, 2 Used

Memory in use: 2.0 GB Total, 2.0 GB Used

Applications: 1 Running, 0 Completed

Drivers: 1 Running, 0 Completed

Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory
worker-20180605231805-172.27.35.11-49531	172.27.35.11:49531	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)
worker-20180605231805-172.27.35.12-35053	172.27.35.12:35053	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20180605231933-0000	(all) Spark Pi	1	1024.0 MB	2018/06/05 23:19:33	root	RUNNING	9 s

Running Drivers (1)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Main Class
driver-20180605231925-0000	(all) 2018/06/05 23:19:25	worker-20180605231805-172.27.35.12-35053	RUNNING	1	1024.0 MB	org.apache.spark.examples.SparkPi

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Completed Drivers (0)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Main Class
---------------	----------------	--------	-------	-------	--------	------------

当应用程序运行结束后，进入历史服务器管理界面，如下图所示：

点击相应应用程序，可查看应用程序具体运行情况，如下图所示：

## 结语

到此CentOS下Hadoop+Spark集群搭建已经成功完成啦，让我们开启愉快的大数据之旅吧！

