

Spark 安装及 Jupyter 环境搭建

刘磊

2020 年 11 月

目录

Spark local 模式安装.....	1
安装 scala	1
安装 Spark.....	2
执行测试	4
启动 pyspark	5
伪分布式安装	6
设置日志	8
Web 界面.....	10
Jupyter notebook 环境搭建	10

Spark local 模式安装

安装 scala

将 scala-2.12.8.tgz 复制并解压缩到/apps 目录下，并将解压后的目录名改为

/apps/scala。

```
cp ~/big_data_tools/scala-2.12.8.tgz /apps/  
tar zxvf /apps/scala-2.12.8.tgz  
mv /apps/scala-2.12.8/ /apps/scala
```

删除压缩包

```
rm /apps/scala-2.12.8.tgz
```

使用 vim 打开用户 shell 设置文件 ~/.bashrc。

```
vim ~/.bashrc
```

将 scala 的 bin 目录，追加到用户环境变量中，然后保存退出。

```
# Scala
export SCALA_HOME=/apps/scala
export PATH=$SCALA_HOME/bin:$PATH
```

使设置生效

```
source ~/.bashrc
```

安装 Spark

将 spark 的安装包，复制并解压缩到/apps 目录下，并将解压后的目录名重命名为 spark。

```
cp ~/big_data_tools/spark-2.4.3-bin-hadoop2.7.tgz /apps
tar zxvf /apps/spark-2.4.3-bin-hadoop2.7.tgz
mv /apps/spark-2.4.3-bin-hadoop2.7/ /apps/spark
```

删除压缩包

```
rm /apps/spark-2.4.3-bin-hadoop2.7.tgz
```

使用 vim 打开用户 shell 设置文件 ~/.bashrc。

```
vim ~/.bashrc
```

将 Spark 的配置信息追加到用户环境变量中，然后保存退出。

```
# Spark
export SPARK_HOME=/apps/spark
export PATH=$SPARK_HOME/bin:$PATH
```

```
# Java
export JAVA_HOME=/apps/java
export PATH=$JAVA_HOME/bin:$PATH

# Hadoop
export HADOOP_HOME=/apps/hadoop
export PATH=$HADOOP_HOME/bin:$PATH

# Hbase
export HBASE_HOME=/apps/hbase
export PATH=$HBASE_HOME/bin:$PATH

# Hive
export HIVE_HOME=/apps/hive
export PATH=$HIVE_HOME/bin:$PATH

# Sqoop
export SQOOP_HOME=/apps/sqoop
export PATH=$SQOOP_HOME/bin:$PATH

# Scala
export SCALA_HOME=/apps/scala
export PATH=$SCALA_HOME/bin:$PATH

# Spark
export SPARK_HOME=/apps/spark
export PATH=$SPARK_HOME/bin:$PATH
```

执行 source 命令，使用户环境变量生效。

```
source ~/.bashrc
```

不需要对 spark 进行任何配置，就可以启动 spark-shell 进行任务处理了。

在终端中执行

```
spark-shell
```

或执行

```
spark-shell local
```

可以启动本地模式。

```
lei@ubuntu:/apps/spark/bin$ spark-shell
20/09/01 23:52:41 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://ubuntu:4040
Spark context available as 'sc' (master = local[*], app id = local-1598975573738).
Spark session available as 'spark'.
Welcome to

      ____ _
     / ___ \
    /  _ < \
   /  / \  > \
  /  /___>__ \
 /_____/

 version 2.4.3

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_191)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

查看当前运行模式

```
sc.master
```

```
scala> sc.master
res0: String = local[*]
```

显示为 local[*]，该模式被称为 Local[N]模式，是用单机的多个线程来模拟 Spark 分布式计算。

执行测试

在 Spark Shell 中，使用 Scala 加载 Spark 安装目录下文件 README.md 并转变为 RDD。

```
val rdd = sc.textFile("/apps/spark/README.md")
```

对 RDD 进行算子操作，统计文件的行数。

```
rdd.count()
```

可以看到输出为：

```
scala> val rdd = sc.textFile("/apps/spark/README.md")
rdd: org.apache.spark.rdd.RDD[String] = /apps/spark/README.md MapPartitionsRDD[1] at
  textFile at <console>:24

scala> rdd.count()
res0: Long = 105
```

输出以上信息表明安装正确。退出 Spark Shell 交互界面，执行:quit。

```
scala> :quit
lei@ubuntu:/apps$
```

启动 pyspark

在终端中执行

```
PYSPARK_PYTHON=python3 pyspark
```

PYSPARK_PYTHON=python3 指定运行 Spark 的 python 版本。

```
lei@ubuntu:/apps/spark/bin$ PYSPARK_PYTHON=python3 pyspark
/apps/spark/bin/pyspark: line 45: python: command not found
Python 3.6.9 (default, Oct 8 2020, 12:12:24)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
20/11/21 16:02:58 WARN Utils: Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface enp0s3)
20/11/21 16:02:58 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
20/11/21 16:02:59 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

  ____      _
 / ___|  _ \| | | |
 \___ \| |_) | |_| |
  ___) | |_) | | | |
  |___|_|_|\___|_|_|_|

 version 2.4.3

Using Python version 3.6.9 (default, Oct 8 2020 12:12:24)
SparkSession available as 'spark'.
>>>
```

在 Spark Shell 中，使用 Python 加载 Spark 安装目录下文件 README.md 并转变为

RDD

```
rdd = sc.textFile("file:/apps/spark/README.md")
```

对 RDD 进行算子操作，统计文件的行数。

```
rdd.count()
```

可以看到输出为：

```
>>> rdd = sc.textFile("file:/apps/spark/README.md")
>>> rdd.count()
105
```

到此 Spark Local 模式已经安装完成。

伪分布式安装

安装伪分布式，还需要对配置文件做一些修改。进入配置文件目录/apps/spark/conf

```
cd /apps/spark/conf
```

将 slaves.template 重命名为 slaves

```
mv slaves.template slaves
```

之前只有一个节点，保持原样就可以了

```
# A Spark Worker will be started on each of the machines listed below.  
localhost
```

将 spark-env.sh.template 重命名 spark-env.sh

```
mv spark-env.sh.template spark-env.sh
```

在 spark-env.sh 中添加如下内容

```
HADOOP_CONF_DIR=/apps/hadoop/etc/hadoop  
JAVA_HOME=/apps/java  
SPARK_MASTER_IP=ubuntu  
SPARK_MASTER_PORT=7077  
SPARK_MASTER_WEBUI_PORT=8080  
SPARK_WORKER_CORES=1  
SPARK_WORKER_MEMORY=1g  
SPARK_WORKER_PORT=7078  
SPARK_WORKER_WEBUI_PORT=8081  
SPARK_EXECUTOR_INSTANCES=1
```

说明：需要配置 JAVA_HOME 以及 HADOOP 配置文件所在的目录

HADOOP_CONF_DIR。SPARK_MASTER_IP、SPARK_MASTER_PORT、

SPARK_MASTER_WEBUI_PORT，分别指 spark 集群中，master 节点的 ip 地址、端口

号、提供的 web 接口的端口。SPARK_WORKER_CORES、SPARK_WORKER_MEMORY

分布为 worker 节点的内核数、内存大小。此处根据自己机器情况调整配置项参数，比如

ip 地址改为自己的主机名。

配置传递给 spark 应用程序的默认属性

将 spark-defaults.conf.template 重命名 spark-defaults.conf

```
mv spark-defaults.conf.template spark-defaults.conf
```

在其中添加如下内容

```
spark.master                spark://ubuntu:7077
spark.eventLog.enabled      true
spark.eventLog.dir          hdfs://localhost:9000/spark/eventLog
spark.serializer            org.apache.spark.serializer.KryoSerializer
spark.driver.memory         1g
spark.jars.package          Azure:mmlspark:0.12
```

```
spark.master                spark://ubuntu:7077
spark.eventLog.enabled      true
spark.eventLog.dir          hdfs://localhost:9000/spark/eventLog
spark.serializer            org.apache.spark.serializer.KryoSerializer
spark.driver.memory         1g
spark.jars.package          Azure:mmlspark:0.12
```

MMLSpark 是微软开源的用于 Spark 的深度学习库，为 Apache Spark 提供了大量深度学习和数据科学工具，包括将 Spark Machine Learning 管道与 Microsoft Cognitive Toolkit(CNTK)和 OpenCV 进行无缝集成，使您能够快速创建功能强大，高度可扩展的大型图像和文本数据集分析预测模型。

eventLog 用来存放日志，需要手动创建

```
hadoop fs -mkdir -p /spark/eventLog
```

spark-defaults.conf 文件不配置的话，运行演示示例的任务不会显示在 web 界面中。

启动 Spark

切换目录到/apps/spark/sbin 目录下，启动 Spark。注意启动 Spark 之前需要启动

Hadoop。

```
cd /apps/spark/sbin
./start-all.sh
```

执行 jps，查看进程变化。

```
lei@ubuntu:/apps/spark/sbin$ jps
8944 DataNode
15026 Worker
9443 ResourceManager
8773 NameNode
9606 NodeManager
14872 Master
15066 Jps
9181 SecondaryNameNode
```

可以看到 Spark 创建了 Master 和 Worker 两个进程。

运行演示实例

运行计算 pi 的例子

```
/apps/spark/bin/run-example SparkPi
```

```
2020-09-05 16:58:37,426 INFO scheduler.DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 0.933 s
2020-09-05 16:58:37,434 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
2020-09-05 16:58:37,442 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 1.086169 s
Pi is roughly 3.137555687778439
2020-09-05 16:58:37,483 INFO server.AbstractConnector: Stopped Spark@78e1384b{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2020-09-05 16:58:37,492 INFO ui.SparkUI: Stopped Spark web UI at http://ubuntu:4040
2020-09-05 16:58:37,501 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2020-09-05 16:58:37,518 INFO memory.MemoryStore: MemoryStore cleared
2020-09-05 16:58:37,519 INFO storage.BlockManager: BlockManager stopped
```

日志信息很多，很难找到输出结果，下面对日志进行设置。

设置日志

上面运行过程中，由于 Log4j 的日志输出级别为 INFO 级别，所以会在屏幕上输出很多的

日志信息，造成很难定位程序的输出结果。可以通过修改日志级别进行解决。

切换目录到/apps/spark/sbin 目录下，停止 Spark。

```
/apps/spark/sbin/stop-all.sh
```

再切换目录到/apps/spark/conf 目录下，将目录下 log4j.properties.template 重命名为 log4j.properties。

```
cd /apps/spark/conf
```



```
mv log4j.properties.template log4j.properties
```

使用 vim 打开 log4j.properties 文件。

```
vim log4j.properties
```

第 19 行修改 log4j.rootCategory 的值为 WARN

```
log4j.rootCategory=WARN, console
```

```
18 # Set everything to be logged to the console
19 log4j.rootCategory=WARN, console
20 log4j.appender.console=org.apache.log4j.ConsoleAppender
```

启动 Spark, 再次运行演示实例, 可以很容易找到结果。

```
/apps/spark/bin/run-example SparkPi
```

```
lei@ubuntu:/apps/spark/conf$ /apps/spark/bin/run-example SparkPi
20/09/05 17:03:56 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Pi is roughly 3.140995704978525
```

使用 pyspark 统计 HDFS 上文件的行数

在 HDFS 上新建目录/input/spark 并上传文件 README.md 到该目录

```
hadoop fs -mkdir /input/spark/
```

```
hadoop fs -put /apps/spark/README.md /input/spark/
```

启动 pyspark

```
PYSPARK_PYTHON=python3 pyspark
```

使用 python 加载 HDFS 上的 README.md 文件, 并转变为 RDD

```
rdd = sc.textFile("hdfs://localhost:9000/input/spark/README.md")
```

统计文件的行数

```
rdd.count()
```

```
>>> rdd = sc.textFile("hdfs://localhost:9000/input/spark/README.md");
>>> rdd.count()
105
```

查看当前运行模式

```
sc.master
```


```
>>> sc.master
'spark://ubuntu:7077'
```

因为我们在 spark-defaults.conf 中对主节点进行了设置，所以这里显示的运行模式不再是 local。

Web 界面

<http://localhost:8080/>

可以看到只有一个 worker，我们运行的例子显示的已完成的列表里。

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

URL: spark://ubuntu:7077
Alive Workers: 1
Cores in use: 1 Total, 1 Used
Memory in use: 1024.0 MB Total, 1024.0 MB Used
Applications: 1 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20200905173929-10.0.2.15-7078	10.0.2.15:7078	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-20200905174822-0001	(kill) Spark shell	1	1024.0 MB	2020/09/05 17:48:22	lei	RUNNING	1.5 min

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20200905173944-0000	Spark Pi	1	1024.0 MB	2020/09/05 17:39:44	lei	FINISHED	5 s

Jupyter notebook 环境搭建

安装 jupyter notebook

```
sudo apt-get install jupyter-notebook
```

新建工作目录~/work_pyspark

```
mkdir ~/work_pyspark
```

进入目录，执行以下命令在 jupyter notebook 中运行 spark

```
PYSPARK_DRIVER_PYTHON=jupyter PYSPARK_DRIVER_PYTHON_OPTS='notebook'
PYSPARK_PYTHON=python3 pyspark
```

```
lei@ubuntu:~/work_pyspark$ PYSARK_DRIVER_PYTHON=jupyter PYSARK_DRIVER_PYTHON_OPTS='notebook' PYSARK_PYTHON=python3 pyspark
/apps/spark/bin/pyspark: line 45: python: command not found
[I 18:09:04.719 NotebookApp] Serving notebooks from local directory: /home/lei/work_pyspark
[I 18:09:04.720 NotebookApp] 0 active kernels
[I 18:09:04.720 NotebookApp] The Jupyter Notebook is running at:
[I 18:09:04.720 NotebookApp] http://localhost:8888/?token=26eb894edd1e467b55cb0b8d2800a4ca2b1cdea99a5ae78e
[I 18:09:04.720 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 18:09:04.721 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
    http://localhost:8888/?token=26eb894edd1e467b55cb0b8d2800a4ca2b1cdea99a5ae78e
[I 18:09:05.082 NotebookApp] Accepting one-time-token-authenticated connection from 127.0.0.1
```

为方便起见，可以将下面的环境变量添加到~/.bashrc 中

```
export PYSARK_DRIVER_PYTHON=jupyter
export PYSARK_DRIVER_PYTHON_OPTS='notebook'
export PYSARK_PYTHON=python3
```

```
# pyspark
export PYSARK_DRIVER_PYTHON=jupyter
export PYSARK_DRIVER_PYTHON_OPTS='notebook'
export PYSARK_PYTHON=python3
```

使设置生效

```
source ~/.bashrc
```

这样在终端中执行 pyspark，就默认在 jupyter notebook 中运行 spark。

```
lei@ubuntu:~/work_pyspark$ pyspark
/apps/spark/bin/pyspark: line 45: python: command not found
[I 18:10:13.835 NotebookApp] Serving notebooks from local directory: /home/lei/work_pyspark
[I 18:10:13.835 NotebookApp] 0 active kernels
[I 18:10:13.835 NotebookApp] The Jupyter Notebook is running at:
[I 18:10:13.835 NotebookApp] http://localhost:8888/?token=6e881d5594299c54458dd13c9b50ee83d69562aa40a83ba0
[I 18:10:13.835 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 18:10:13.836 NotebookApp]

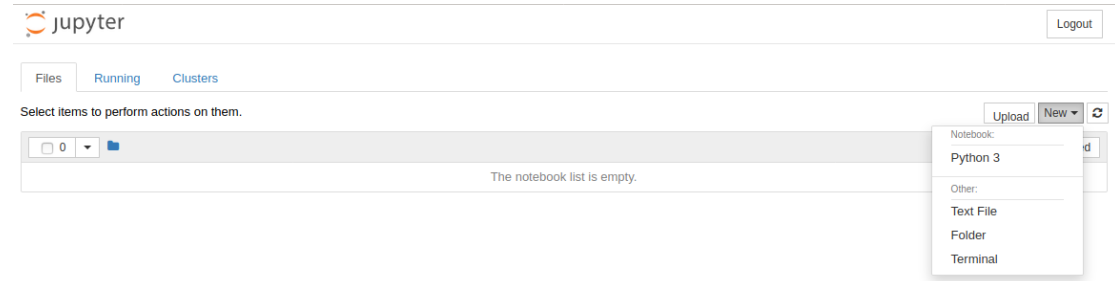
Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
    http://localhost:8888/?token=6e881d5594299c54458dd13c9b50ee83d69562aa40a83ba0
[I 18:10:14.173 NotebookApp] Accepting one-time-token-authenticated connection from 127.0.0.1
```

新建一个工作目录

```
mkdir ~/pyspark-workspace
```

进入目录，执行 pyspark 以后浏览器自动打开，在弹出的页面点击【New】，【Python3】

新建一个 notebook.



在打开的页面中，依次运行以下命令进行测试

```
In [1]: sc.master
```

```
Out[1]: 'spark://ubuntu:7077'
```

```
In [2]: rdd = sc.textFile("file:/apps/spark/README.md")
rdd.count()
```

```
Out[2]: 105
```

```
In [3]: rdd = sc.textFile("hdfs://localhost:9000/input/spark/README.md")
rdd.count()
```

```
Out[3]: 105
```

测试没有问题，jupyter notebook 环境就搭建好了。