第 7 篇： Spark 的安装和基础编程

学习 blog：http://dblab.xmu.edu.cn/blog/931-2/#more-931

一、安装 Spark

1. 下载安装文件

（1）在本书软件下载专区下载文件“spark-1.6.2-bin-without-hadoop.tgz”，放入"/home/下载”目录下。

Spark 部署模式主要有四种：Local 模式（单机模式）、Standalone 模式（使用 Spark 自带的简单集群管理器）、YARN 模式（使用 YARN 作为集群管理器）和 Mesos 模式（使用 Mesos 作为集群管理器）。这里介绍 Local 模式（单机模式）的 Spark 安装。

（2）解压文件，并重命名和修改权限

```
sudo tar -zxf ~/下载/spark-1.6.2-bin-without-hadoop.tgz -C /usr/local/
cd /usr/local
sudo mv ./spark-1.6.2-bin-without-hadoop/ ./spark
sudo chown -R hadoop:hadoop ./spark
```

2. 配置相关文件

安装后，还需要修改 Spark 的配置文件 spark-env.sh

```
cd /usr/local/spark
cp ./conf/spark-env.sh.template ./conf/spark-env.sh
```

编辑 spark-env.sh 文件(vim ./conf/spark-env.sh)，在第一行添加以下配置信息:

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

有了上面的配置信息以后，Spark 就可以把数据存储到 Hadoop 分布式文件系统 HDFS 中，也可以从 HDFS 中读取数据。如果没有配置上面信息，Spark 就只能读写本地数据，无法读写 HDFS 数据。

配置完成后就可以直接使用，不需要像 Hadoop 运行启动命令。

通过运行 Spark 自带的示例，验证 Spark 是否安装成功。

```
cd /usr/local/spark
bin/run-example SparkPi
```

执行时会输出非常多的运行信息，输出结果不容易找到，可以通过 grep 命令进行过滤（命令中的 2>&1 可以将所有的信息都输出到 stdout 中，否则由于输出日志的性质，还是会输出到屏幕中）：

```
bin/run-example SparkPi 2>&1 | grep "Pi is"
```

这里涉及到 Linux Shell 中管道的知识，详情可以参考 Linux Shell 中的管道命令

过滤后的运行结果如下图示，可以得到 π 的 5 位小数近似值：

二、使用 Spark Shell 编写代码

学习 Spark 程序开发，建议首先通过 spark-shell 交互式学习，加深 Spark 程序开发的理解。

这里介绍 Spark Shell 的基本使用。Spark shell 提供了简单的方式来学习 API，并且提供了交互的方式来分析数据。你可以输入一条语句，Spark shell 会立即执行语句并返回结果，这就是我们所说的 REPL（Read-Eval-Print Loop，交互式解释器），为我们提供了交互式执行环境，表达式计算完成就会输出结果，而不必等到整个程序运行完毕，因此可即时查看中间结果，并对程序进行修改，这样可以在很大程度上提升开发效率。

Spark Shell 支持 Scala 和 Python，这里使用 Scala 来进行介绍。

可以执行“spark-shell –help”命令，获取完整的选项列表，具体如下：

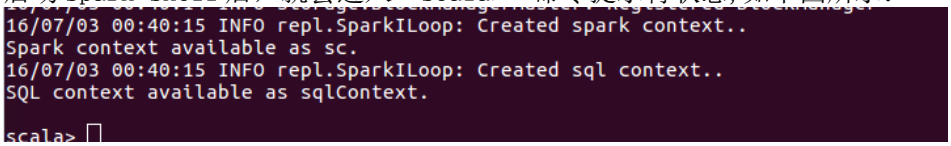
```
cd /usr/local/spark
./bin/spark-shell --help
```

上面是命令使用方法介绍，下面正式使用命令进入 spark-shell 环境，可以通过下面命令启动 spark-shell 环境：

```
bin/spark-shell
```

该命令省略了参数，这时，系统默认是“bin/spark-shell –master local[*]”，也就是说，是采用本地模式运行，并且使用本地所有的 CPU 核心。

启动 spark-shell 后，就会进入“scala>”命令提示符状态,如下图所示：



现在，你就可以在里面输入 scala 代码进行调试了。启动进入 Spark Shell 以后，系统自动为你创建了一个专有的 SparkContext，变量名叫做 sc，可以直接使用（后面章节的 RDD 编程中，就会用到 sc 变量）。

比如，下面在命令提示符后面输入一个表达式“8 * 2 + 5”，然后回车，就会立即得到结果：

```
scala> 8*2+5
res0: Int = 21
```

最后，可以使用命令“:quit”退出 Spark Shell，如下所示：

```
scala>:quit
```

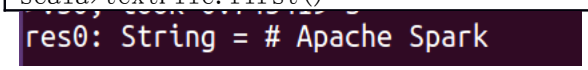
或者，也可以直接使用“Ctrl+D”组合键，退出 Spark Shell。

三、读取文件

1. 读取本地文件

读取本地文件 README.md 的内容，并显示第 1 行

```
cd /usr/local/spark
./bin/spark-shell    #启动 spark
scala> val textFile=sc.textFile("file:///usr/local/spark/README.md")
scala>textFile.first()
```



2. 读取 HDFS 文件

```
cd /usr/local/hadoop
./sbin/start-dfs.sh   #启动 hadoop
#传本地文件
./bin/hdfs dfs -mkdir -p  /user/hadoop
./bin/hdfs dfs -mkdir input2
./bin/hdfs dfs -ls
./bin/hdfs dfs -put /usr/local/spark/README.md  input2
./bin/hdfs dfs -ls input2
./bin/hdfs dfs -cat input2/README.md
#显示第 1 行内容
scala>val textFile=sc.textFile("/user/hadoop/input2/README.md")
scala>textFile.first()
```

3.编写词频统计程序

（1）input1 目录下先放一个文件 wordfile1.txt

```
hadoop@ubuntu:/usr/local/hadoop$ bin/hdfs dfs -ls input1
hadoop@ubuntu:/usr/local/hadoop$ bin/hdfs dfs -put ~/wordfile1.txt input1
hadoop@ubuntu:/usr/local/hadoop$ bin/hdfs dfs -put ~/wordfile2.txt input1
hadoop@ubuntu:/usr/local/hadoop$ bin/hdfs dfs -ls input1
```

（2）统计词频

```
scala>val fileFile=sc.fileFile("/user/hadoop/input1/*.txt")
scala>
val wordCount=fileFile.flatMap(line=>line.split(" ")).map(word=>(word,1)).reduceByKey((a,b)=>a+b)
scala>wordCount.collect()
```

```
res2: Array[(String, Int)] = Array((is,2), (love,2), (fast,1), ("",1), (Spark,2), (I,2), (good,1), (Hadoop,2))
```

解释如下：

一、line=>line.spit(" ")：”表示输入参数 line，执行函数 line.split(" ")，即对一行的内容用空格分割。

二、flatMap(line=>line.spit(" "))：对每行的内容用空格作单词切割，形成单词集合，把单词集合拍扁得到一个大的单词集合。

三、map(word=>(word,1))：遍历集合中的每个单词，遍历一个就复制给 word，并执行 Lamda 表达式 word=>(word,1)

四、word=>(word,1)：word 的作为函数的输入参数，构建得到一个映射，这个映射的 key 是 word，value 是 1

五、reduceByKey((a,b)=>a+b)：该函数将映射中的所有(key,value)按 key 分组，对具有相同 key 的多个 value 进行聚合操作，返回聚合后的(key,value)。例如("hadoop",1)和("hadoop",1)具有相同的 Key，聚合后得到("hadoop",2)

四、编写 Spark 独立应用程序

1.用 Java 语言编写 Spark 独立应用程序

（1）安装 Maven

```
sudo unzip ~/下载/apache-maven-3.3.9-bin.zip -d /usr/local
cd /usr/local
sudo mv apache-maven-3.3.9  ./maven
sudo chown -R hadoop ./maven
```