

18-WordCount：从零开始运行你的第一个Spark应用

你好，我是蔡元楠。

今天我们来从零开始运行你的第一个Spark应用。

我们先来回顾一下模块三的学习路径。

首先，我们由浅入深地学习了Spark的基本数据结构RDD，了解了它这样设计的原因，以及它所支持的API。

之后，我们又学习了Spark SQL的DataSet/DataFrame API，了解到它不仅提供类似于SQL query的接口，大大提高了开发者的工作效率，还集成了Catalyst优化器，可以提升程序的性能。

这些API应对的都是批处理的场景。

再之后，我们学习了Spark的流处理模块：Spark Streaming和Structured Streaming。两者都是基于微批处理（Micro batch processing）的思想，将流数据按时间间隔分割成小的数据块进行批处理，实时更新计算结果。

其中Structured Streaming也是使用DataSet/DataFrame API，这套API在某种程度上统一了批处理和流处理，是当前Spark最流行的工具，我们必需要好好掌握。

虽然学习了这么多API以及它们的应用，但是大部分同学还没有从零开始写一个完整的Spark程序，可能更没有运行Spark程序的经历。纸上谈兵并不能帮助我们在工作中用Spark解决实际问题。所以，今天我就和你一起做个小练习，从在本地安装Spark、配置环境开始，为你示范怎样一步步解决之前提到数次的统计词频（Word Count）的问题。

通过今天的学习，你可以收获：

- 怎样安装Spark以及其他相关的模块；
- 知道什么是SparkContext、SparkSession；
- 一个完整的Spark程序应该包含哪些东西；
- 用RDD、DataFrame、Spark Streaming如何实现统计词频。

这一讲中，我们使用的编程语言是Python，操作系统是Mac OS X。

在这一讲以及之前文章的例子中，我们都是用Python作为开发语言。虽然原生的Spark是用Scala实现，但是在大数据处理领域中，我个人最喜欢的语言是Python。因为它非常简单易用，应用非常广泛，有很多的库可以方便我们开发。

当然Scala也很棒，作为一个函数式编程语言，它很容易用链式表达对数据集进行各种处理，而且它的运行速度是最快的，感兴趣的同学可以去学习一下。

虽然Spark还支持Java和R，但是我个人不推荐你使用。用Java写程序实在有些冗长，而且速度上没有优势。

操作系统选Mac OS X是因为我个人喜欢使用Macbook，当然Linux/Ubuntu也很棒。

安装Spark

首先，我们来简单介绍一下如何在本地安装Spark，以及用Python实现的Spark库——PySpark。

在前面的文章中，我们了解过，Spark的job都是JVM（Java Virtual Machine）的进程，所以在安装运行Spark之前，我们需要确保已经安装Java Developer Kit（JDK）。在命令行终端中输入：

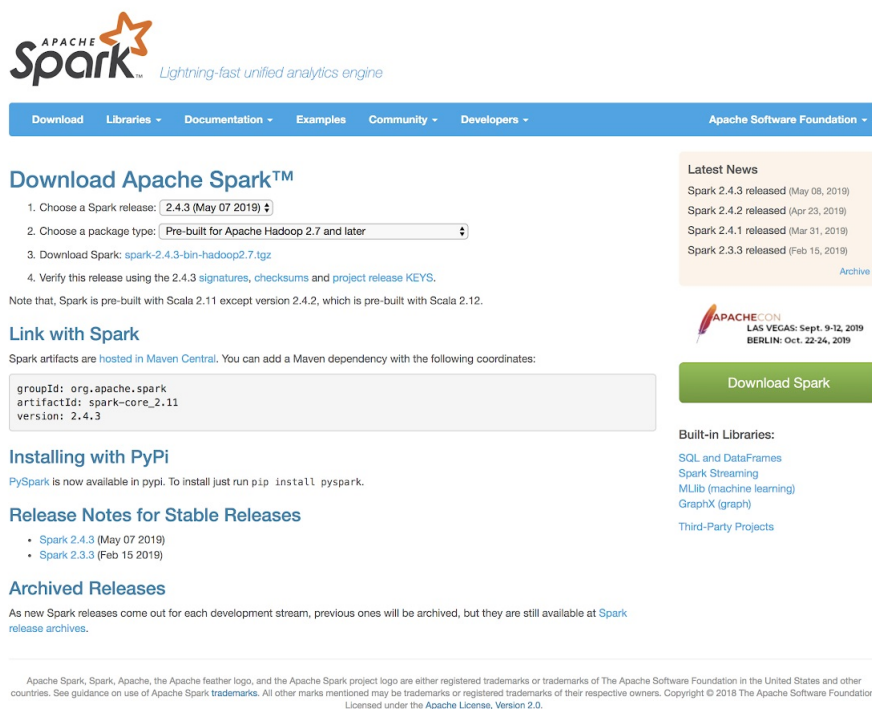
```
java -version
```

如果命令行输出了某个Java的版本，那么说明你已经有JDK或者JRE在本地。如果显示无法识别这个命令，那么说明你还没有安装JDK。这时，你可以去[Oracle的官网](#)去下载安装JDK，然后配置好环境变量。

同样，我们需要确保Python也已经被安装在本地了。在命令行输入“Python”或者“Python3”，如果可以成功进入交互式的Python Shell，就说明已经安装了Python。否则，需要去[Python官网](#)下载安装Python。这里，我推荐你使用Python3而不是Python2。

我们同样可以在本地预装好Hadoop。Spark可以脱离Hadoop运行，不过有时我们也需要依赖于HDFS和YARN。所以，这一步并不是必须的，你可以自行选择。

接下来我们就可以安装Spark。首先去[Spark官网](#)的下载界面。在第一个下拉菜单里选择最新的发布，第二个菜单最好选择与Hadoop 2.7兼容的版本。因为有时我们的Spark程序会依赖于HDFS和YARN，所以选择最新的Hadoop版本比较好。



The screenshot shows the Apache Spark download page. The main heading is "Download Apache Spark™". Below it, there are four steps: 1. Choose a Spark release: 2.4.3 (May 07 2019) (with a dropdown arrow). 2. Choose a package type: Pre-built for Apache Hadoop 2.7 and later (with a dropdown arrow). 3. Download Spark: spark-2.4.3-bin-hadoop2.7.tgz. 4. Verify this release using the 2.4.3 signatures, checksums and project release KEYS. Below these steps, a note states: "Note that, Spark is pre-built with Scala 2.11 except version 2.4.2, which is pre-built with Scala 2.12." There is a "Link with Spark" section with the text "Spark artifacts are hosted in Maven Central. You can add a Maven dependency with the following coordinates:" and a code block showing: groupId: org.apache.spark, artifactId: spark-core_2.11, version: 2.4.3. Below that is "Installing with PyPi" with the text "PySpark is now available in pypi. To install just run pip install pyspark." and "Release Notes for Stable Releases" with a list of links for Spark 2.4.3 (May 07 2019) and Spark 2.3.3 (Feb 15 2019). There is also an "Archived Releases" section with the text "As new Spark releases come out for each development stream, previous ones will be archived, but they are still available at Spark release archives." On the right side, there is a "Latest News" section with links to Spark 2.4.3 released (May 08, 2019), Spark 2.4.2 released (Apr 23, 2019), Spark 2.4.1 released (Mar 31, 2019), and Spark 2.3.3 released (Feb 15, 2019). Below that is a "Download Spark" button. At the bottom right, there is a "Built-in Libraries" section with links to SQL and DataFrames, Spark Streaming, MLlib (machine learning), GraphX (graph), and Third-Party Projects. At the very bottom, there is a small footer with legal information.

下载好之后，解压缩Spark安装包，并且把它移动到/usr/local目录下，在终端中输入下面的代码。

```
$ tar -xzf ~/Downloads/spark-2.4.3-bin-hadoop2.7.tg
$ mv spark-2.4.3-bin-hadoop2.7.tgz /usr/local/spark
```

经过上述步骤，从官网下载并安装Spark的文件，这样我们便完成了Spark的安装。但是，Spark也是要进行相应的环境变量配置的。你需要打开环境变量配置文件。

```
vim ~/.bash_profile
```

并在最后添加一段代码。

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

这样，所需的步骤都做完之后，我们在命令行控制台输入PySpark，查看安装情况。如果出现下面的欢迎标志，就说明安装完毕了。

```

Welcome to

      ____
     /  _/  _  _  _  _/  _/
    _\  \/_  \/_  \/_  \/_  \/_
   /__ /  .__/_\_,_/_/_/_/_\_\_  version 2.4.3
      /_/_

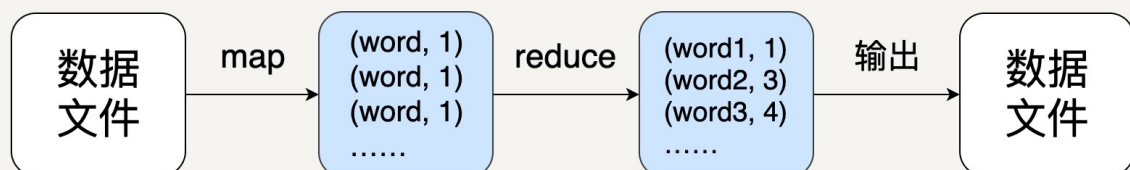
Using Python version 2.7.10 (default, Oct 6 2017 22:29:07)
SparkSession available as 'spark'.

>>>

```

基于RDD API的Word Count程序

配置好所需的开发环境之后，下一步就是写一个Python程序去统计词语频率。我们都知道这个程序的逻辑应该是如下图所示的。



对于中间的先map再reduce的处理，我相信通过前面的学习，所有同学都可以用RDD或者DataFrame实现。

但是，我们对于Spark程序的入口是什么、如何用它读取和写入文件，可能并没有了解太多。所以，接下来

让我们先接触一下Spark程序的入口。

在Spark 2.0之前，**SparkContext**是所有Spark任务的入口，它包含了Spark程序的基本设置，比如程序的名字、内存大小、并行处理的粒度等，Spark的驱动程序需要利用它来连接到集群。

无论Spark集群有多少个节点做并行处理，每个程序只可以有唯一的SparkContext，它可以被SparkConf对象初始化。

```
conf = SparkConf().setAppName(appName).setMaster(master)
sc = SparkContext(conf=conf)
```

这个appName参数是一个在集群UI上展示应用程序的名称，master参数是一个Spark、Mesos 或YARN的集群URL，对于本地运行，它可以被指定为“local”。

在统计词频的例子中，我们需要通过SparkContext对象来读取输入文件，创建一个RDD，如下面的代码所示。

```
text_file = sc.textFile("file://...") //替换成实际的本地文件路径。
```

这里的text_file是一个RDD，它里面的每一个数据代表原文本文件中的一行。

在这些版本中，如果要使用Spark提供的其他库，比如SQL或Streaming，我们就需要为它们分别创建相应的context对象，才能调用相应的API，比如的DataFrame和DStream。

```
hc = HiveContext(sc)
ssc = StreamingContext(sc)
```

在Spark 2.0之后，随着新的DataFrame/DataSet API的普及化，Spark引入了新的**SparkSession**对象作为所有Spark任务的入口。

SparkSession不仅有SparkContext的所有功能，它还集成了所有Spark提供的API，比如DataFrame、Spark Streaming和Structured Streaming，我们再也不用为不同的功能分别定义Context。

在统计词频的例子中，我们可以这样初始化SparkSession以及创建初始RDD。

```
spark = SparkSession
    .builder
    .appName(appName)
    .getOrCreate()
text_file = spark.read.text("file://...").rdd.map(lambda r: r[0])
```

由于SparkSession的普适性，我推荐你尽量使用它作为你们Spark程序的入口。随后的学习中，我们会逐渐了解怎样通过它调用DataFrame和Streaming API。

让我们回到统计词频的例子。在创建好代表每一行文本的RDD之后，接下来我们便需要两个步骤。

1. 把每行的文本拆分成一个个词语；
2. 统计每个词语的频率。

对于第一步，我们可以用flatMap去把行转换成词语。对于第二步，我们可以先把每个词语转换成（word, 1）的形式，然后用reduceByKey去把相同词语的次数相加起来。这样，就很容易写出下面的代码了。

```
counts = lines.flatMap(lambda x: x.split(' '))
                .map(lambda x: (x, 1))
                .reduceByKey(add)
```

这里counts就是一个包含每个词语的（word, count）pair的RDD。

相信你还记得，只有当碰到action操作后，这些转换动作才会被执行。所以，接下来我们可以用collect操作把结果按数组的形式返回并输出。

```
output = counts.collect()
for (word, count) in output:
    print("%s: %i" % (word, count))
spark.stop() // 停止SparkSession
```

基于DataSet API的Word Count程序

讲完基于RDD API的Word Count程序，接下来让我们学习下怎样用DataSet API来实现相同的效果。

在DataSet的世界中，我们可以把所有的词语放入一张表，表中的每一行代表一个词语，当然这个表只有一列。我们可以对这个表用一个groupBy()操作把所有相同的词语聚合起来，然后用count()来统计出每个group的数量。

```
spark = SparkSession
    .builder
    .appName(appName)
    .getOrCreate()
ds_lines = spark.read.textFile("file:///...")
ds = ds_lines.flatMap(lambda x: x.split(' '))
                .groupBy("Value")
                .count()

ds.show()
```

```
spark.stop()
```

从这个例子，你可以很容易看出使用DataSet/DataFrame API的便利性——我们不需要创建（word, count）的pair来作为中间值，可以直接对数据做类似SQL的查询。

小结

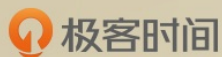
通过今天的学习，我们掌握了如何从零开始创建一个简单的Spark的应用程序，包括如何安装Spark、如何配置环境、Spark程序的基本结构等等。

实践题

希望你可以自己动手操作一下，这整个过程只需要跑通一次，以后就可以脱离纸上谈兵，真正去解决问题。

欢迎你在留言中反馈自己动手操作的效果。

如果你跑通了，可以在留言中打个卡。如果遇到了问题，也请你在文章中留言，与我和其他同学一起讨论。

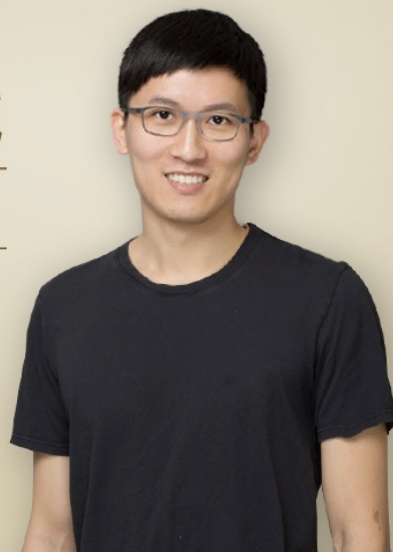


大规模数据处理实战

Google 一线工程师的大数据架构实战经验

蔡元楠

Google Brain 资深工程师



新版升级：点击「👤 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

精选留言：

- 明翼 2019-05-29 20:32:00
这个课程感觉成大数据入门课了………… [8赞]
- Jerry 2019-05-29 15:31:45
一直跟着作者的脚本到现在，学到不少，本来今天第一次非常兴奋可以进入实战了，结果过了今天的课程感觉有些小失望，代码有不少是不work的，也没有一个完整的demo，最后还是自己去pyspark官方网站上看了示例才明白：<https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html> [3赞]

- Michael 2019-05-29 11:14:05

```
spark_session = SparkSession.builder.appName("PySparkShell").getOrCreate()
ds_lines = spark_session.read.textFile("README.md")
ds = ds_lines.flatMap(lambda x: x.split(' ')).groupBy("Value").count()
ds.show()
```

我执行这段的时候报错了

AttributeError: 'DataFrameReader' object has no attribute 'textFile'

如果把textFile()改成text()就对了

再执行flatMap那段，也报错了

AttributeError: 'DataFrame' object has no attribute 'flatMap'

是不是API变动了，我用的是2.4.3版本单机执行的 [3赞]

- 一 2019-05-29 13:59:57

看了这一讲意识到之前对Python欠缺了重视，现在明白Python在大数据处理领域是很有竞争力的，因为Spark和众多的库的原因，甚至超越Java，所以现在要重新重视起来Python的学习了 [1赞]

- 大志 2019-05-29 11:49:55

老师，本地已经安装了Spark，有Demo吗，只看代码片段的话还是无从下手啊 [1赞]

- 朱同学 2019-05-29 08:20:12

java万金油，什么都可以干，人好招，特别是我们这种偏远地区，scala，虽然开发效率高，但是人少，难招，所以我们大数据团队选择了java。至于运行效率，py是最慢的，java和scala应该半斤八两吧 [1赞]

- hua168 2019-05-30 00:30:54

老师我想问一下，如果大数据学习用python、java、还是Scala？

python虽然代码少，但不是说性能上，运行速度上不及java和go吗？

- J Zhang 2019-05-29 23:20:33

用java写 有点冗长 我不敢苟同，因为java8 已经是函数编程了！而且spark开发我觉得大部分还是spark sql多点！这样基本没啥区别

- 斯盖丸 2019-05-29 22:20:05

.groupBy("Value")这个value是什么意思？

- 大张 2019-05-29 20:20:10

又见银银

- fresh 2019-05-29 15:47:37

能用java 写代码吗？

- 石斌 2019-05-29 15:46:42

flatMap是rdd的算子，df不能直接用，可以explode行转列

- 这个名字居然都有 2019-05-29 15:33:34

老师，你给一个完整的案例吧，

- 许童童 2019-05-29 15:21:48

环境搭好了，下一步不知道怎么操作了。