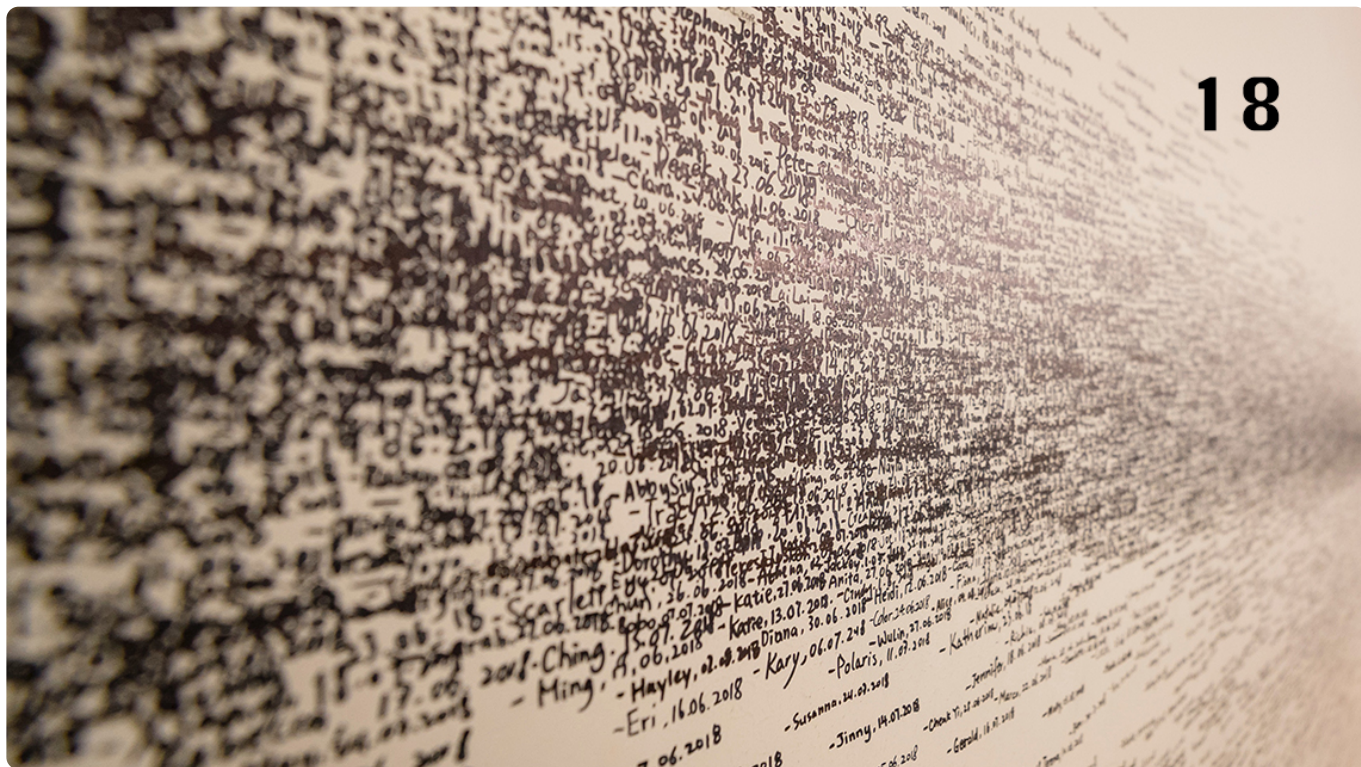


18 | Word Count: 从零开始运行你的第一个Spark应用

2019-05-29 蔡元楠

大规模数据处理实战

[进入课程 >](#)



讲述：巴莫

时长 09:51 大小 9.02M



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生活中用 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：

 复制代码

```
1 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。

Download Apache Spark™

1. Choose a Spark release: **2.4.3 (May 07 2019)**
2. Choose a package type: **Pre-built for Apache Hadoop 2.7 and later**
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](#).

Note that, Spark is pre-built with Scala 2.11 except version 2.4.2, which is pre-built with Scala 2.12.

Link with Spark

Spark artifacts are [hosted in Maven Central](#). You can add a Maven dependency with the following coordinates:

```
groupId: org.apache.spark
artifactId: spark-core_2.11
version: 2.4.3
```

Installing with PyPi

PySpark is now available in pypi. To install just run `pip install pyspark`.

Release Notes for Stable Releases

- [Spark 2.4.3](#) (May 07 2019)
- [Spark 2.3.3](#) (Feb 15 2019)

Archived Releases

As new Spark releases come out for each development stream, previous ones will be archived, but they are still available at [Spark release archives](#).

Latest News

- Spark 2.4.3 released (May 08, 2019)
- Spark 2.4.2 released (Apr 23, 2019)
- Spark 2.4.1 released (Mar 31, 2019)
- Spark 2.3.3 released (Feb 15, 2019)

[Archive](#)



Download Spark


Built-in Libraries:

[SQL and DataFrames](#)
[Spark Streaming](#)
[MLlib \(machine learning\)](#)
[GraphX \(graph\)](#)

[Third-Party Projects](#)


Apache Spark, Spark, Apache, the Apache feather logo, and the Apache Spark project logo are either registered trademarks or trademarks of The Apache Software Foundation in the United States and other countries. See guidance on use of Apache Spark [trademarks](#). All other marks mentioned may be trademarks or registered trademarks of their respective owners. Copyright © 2018 The Apache Software Foundation, Licensed under the [Apache License, Version 2.0](#).

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。

 复制代码


```
1 $ tar -xzf ~/Downloads/spark-2.4.3-bin-hadoop2.7.tgz
2 $ 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 也是要相应进行的环境变量配置的。你需要打开环境变量配置文件。

 复制代码

```
1 vim ~/.bash_profile
```


并在最后添加一段代码。

 复制代码

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


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。

 复制代码

```
1 conf = SparkConf().setAppName(appName).setMaster(master)
2 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。

 复制代码

```
1 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`。


 复制代码

```
1 hc = HiveContext(sc)
2 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。

 复制代码


```
1 spark = SparkSession
2     .builder
3     .appName(appName)
4     .getOrCreate()
5 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。

 复制代码

```
1 counts = lines.flatMap(lambda x: x.split(' '))
2           .map(lambda x: (x, 1))
3           .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。


 复制代码

```
1 output = counts.collect()
2 for (word, count) in output:
3     print("%s: %i" % (word, count))
4 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。

 复制代码

```
1 spark = SparkSession
2     .builder
3     .appName(appName)
4     .getOrCreate()
5 ds_lines = spark.read.textFile("file:///...")
6 ds = ds_lines.flatMap(lambda x: x.split(' '))
7                 .groupBy("Value")
8                 .count()
9 ds.show()
10
11 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，真正去解决实际问题。

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。

 极客时间

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。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 17 | Structured Streaming：如何用DataFrame API进行实时数据分析？

下一篇 19 | 综合案例实战：处理加州房屋信息，构建线性回归模型



明翼

2019-05-29

👍 13

这个课程感觉成大数据入门课了.....

展开 ▾



Jerry

2019-05-29

👍 8

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>

展开 ▾



Michael

2019-05-29

👍 3

```
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()
```

...

展开 ▾



—

2019-05-29

👍 2

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

展开 ▾



青石

2019-05-31

👍 1

```
#!/usr/bin/python3
```

```
import os
from pyspark import SparkContext, SparkConf
```

...

展开 ∨



J Zhang

2019-05-29

👍 1

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



大张

2019-05-29

👍 1

又见银银

展开 ∨



大志

2019-05-29

👍 1

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



朱同学

2019-05-29

👍 1

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

展开 ∨



hua168

2019-05-30

👍

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？



斯盖丸

2019-05-29

👍

.groupBy("Value")这个value是什么意思？

展开 ∨



fresh

2019-05-29



能用java 写代码吗?

展开 ∨



石斌

2019-05-29



flatMap是rdd的算子，df不能直接用，可以explode行转列

展开 ∨



这个名字居...

2019-05-29



老师，你给一个完整的案例吧，

展开 ∨



许童童

2019-05-29



环境搭好了，下一步不知道怎么操作了。

展开 ∨