

SPARK 实践

Spark 简介：

Apache Spark 是专门为大数据处理而设计的通用的计算引擎。spark 拥有 MapReduce 所具有的优点，但不同于 Map Reduce 的是 Job 中间输出结果可以缓存到内存中，从而不再需要读写 HDFS，减少磁盘数据交互，因此 Spark 能更好的适应机器学习和数据挖掘等需要迭代的算法。Spark 提供了 Spark RDD 、 Spark SQL 、 Spark Streaming 、 Spark MLlib 、 Spark GraphX 等技术组件，可以一站式地完成大数据领域的离线批处理、交互式查询、流式计算、机器学习、图计算等常见的任务。这就是 spark 一站式开发的特点。

Spark 安装

官网：<https://spark.apache.org/downloads.html>

Download Apache Spark™

1. Choose a Spark release: **3.5.0 (Sep 13 2023)** ▼
2. Choose a package type: **Pre-built for Apache Hadoop 3.3 and later** ▼
3. Download Spark: [spark-3.5.0-bin-hadoop3.tgz](#)
4. Verify this release using the 3.5.0 [signatures](#), [checksums](#) and [project release KEYS](#) by following these [procedures](#).

Note that Spark 3 is pre-built with Scala 2.12 in general and Spark 3.2+ provides additional pre-built distribution with Scala 2.13.

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_2.12
version: 3.5.0
```

Installing with PyPi

[PySpark](#) is now available in pypi. To install just run `pip install pyspark`.

Convenience Docker Container Images

[Spark Docker Container images](#) are available from [DockerHub](#), these images contain non-ASF software and may be subject to different license terms.

Release notes for stable releases

解压即可。在终端中输入 pySpark，如果安装成功会看到下图：

```
Welcome to  

 ____  

 /--\  /--\  /--\  /--\  

 \_/\_/\_/\_/\_/\_/\_/  

 /--\  /--\  /--\  /--\ version 3.5.0  

 \_/\_/\_/\_/\_/\_/\_/  

 ____  

Using Python version 3.11.5 (main, Sep 11 2023 08:31:25)  

Spark context Web UI available at http://172.25.142.242:4040  

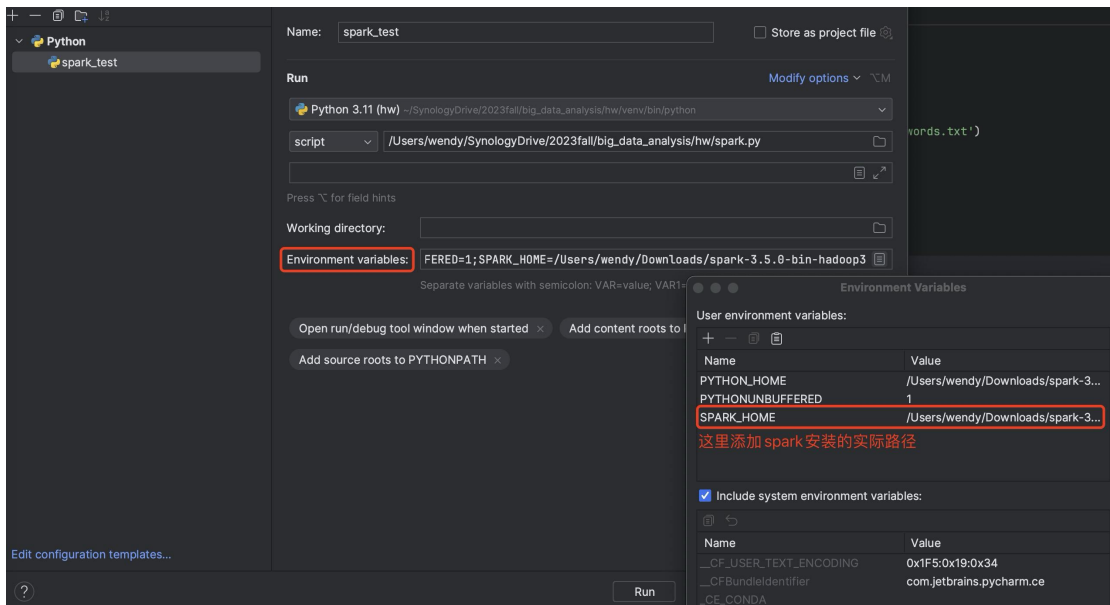
Spark context available as 'sc' (master = local[*, app id = local-1701351226916  

).  

SparkSession available as 'spark'.
```

配置 Pycharm 中调用 pyspark 的加载包

1. 补充配置。PyCharm->Run->Edit Configurations, 添加环境变量。



- ## 2. 初步运行

```
from operator import add
from pyspark import SparkContext

if __name__ == "__main__":
    sc = SparkContext(appName="PythonWordCount")
    lines = sc.textFile('words.txt')
    counts = lines.flatMap(lambda x: x.split(' ')) \
                   .map(lambda x: (x,1)) \
                   .reduceByKey(add)
    output = counts.collect()
    for (word, count) in output:
        print("%s: %i" % (word, count))
    sc.stop()
```

Words.txt 文件

```
words.txt
good bad cool
hadoop spark mlib
good spark mlib
cool spark bad
```

运行结果

```
good: 2
hadoop: 1
bad: 2
cool: 2
spark: 3
mlib: 2
```

Pyspark 使用语法

1. count 返回元素个数

```
from pyspark import SparkContext
sc = SparkContext("local", "count app")
words = sc.parallelize(
    ["scala",
     "java",
     "hadoop",
     "spark",
     "akka",
     "spark vs hadoop",
     "pyspark",
     "pyspark and spark"
])
counts = words.count()
print("Number of elements in RDD -> %i" % counts)
```

```
Number of elements in RDD -> 8
```

2. collect()返回所有元素，并转换为 python 数据类型

```
from pyspark import SparkContext
sc = SparkContext("local", "collect app")
words = sc.parallelize(
    ["scala",
     "java",
     "hadoop",
     "spark",
     "akka",
     "spark vs hadoop",
     "pyspark",|
     "pyspark and spark"
    ])
coll = words.collect()
print("Elements in RDD -> %s" % coll)
Elements in RDD -> ['scala', 'java', 'hadoop', 'spark', 'akka', 'spark vs hadoop', 'pyspark', 'pyspark and spark']
```

3. filter()返回一个包含元素的新的 RDD,满足过滤器内部功能

```
from pyspark import SparkContext
sc = SparkContext("local", "Filter app")
words = sc.parallelize(
    ["scala",
     "java",
     "hadoop",
     "spark",
     "akka",
     "spark vs hadoop",
     "pyspark",
     "pyspark and spark"]
    )
words_filter = words.filter(lambda x: 'spark' in x)
filtered = words_filter.collect()
print("Filtered RDD -> %s" % (filtered))
Filtered RDD -> ['spark', 'spark vs hadoop', 'pyspark', 'pyspark and spark']
```

4. map(f,preservesPartitioning=False), 通过将该函数应用于 RDD 中的每个元素来返回新的 RDD。

```
from pyspark import SparkContext
sc = SparkContext("local", "Map app")
```

```
words = sc.parallelize(  
    ["scala",  
    "java",  
    "hadoop",  
    "spark",  
    "akka",  
    "spark vs hadoop",  
    "pyspark",  
    "pyspark and spark"]  
)  
words_map = words.map(lambda x: (x, 1))  
mapping = words_map.collect()  
print("Key value pair -> %s" % (mapping))
```