



FREE eBook

LEARNING

pyspark

Free unaffiliated eBook created from
Stack Overflow contributors.

#pyspark

Table of Contents

About	1
Chapter 1: Getting started with pyspark	2
Remarks.....	2
Examples.....	2
Installation or Setup.....	2
Sample Word Count in Pyspark.....	2
Consuming Data From S3 using PySpark.....	2
Credits	4

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [pyspark](#)

It is an unofficial and free pyspark ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official pyspark.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with pyspark

Remarks

This section provides an overview of what pyspark is, and why a developer might want to use it.

It should also mention any large subjects within pyspark, and link out to the related topics. Since the Documentation for pyspark is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting pyspark set up or installed.

Sample Word Count in Pyspark

The underlying example is just the one given in the official pyspark documentation. Please click [here](#) to reach this example.

```
# the first step involves reading the source text file from HDFS
text_file = sc.textFile("hdfs://...")

# this step involves the actual computation for reading the number of words in the file
# flatmap, map and reduceByKey are all spark RDD functions
counts = text_file.flatMap(lambda line: line.split(" ")) \
    .map(lambda word: (word, 1)) \
    .reduceByKey(lambda a, b: a + b)

# the final step is just saving the result.
counts.saveAsTextFile("hdfs://...")
```

Consuming Data From S3 using PySpark

There are two methods using which you can consume data from AWS S3 bucket.

1. Using `sc.textFile` (or `sc.wholeTextFiles`) API: This api can be used for HDFS and local file system as well.

```
aws_config = {} # set your aws credential here
sc._jsc.hadoopConfiguration().set("fs.s3n.awsSecretAccessKey",
aws_config['aws.secret.access.key'])
sc._jsc.hadoopConfiguration().set("fs.s3n.awsSecretAccessKey",
aws_config['aws.secret.access.key'])
s3_keys = ['s3n/{bucket}/{key1}', 's3n/{bucket}/{key2}']
data_rdd = sc.wholeTextFiles(s3_keys)
```

2. Reading it using custom API (Say a boto downloader):

```
def download_data_from_custom_api(key):
    # implement this function as per your understanding (if you're new, use [boto][1] api)
    # don't worry about multi-threading as each worker will have single thread executing your
    job
    return ''

s3_keys = ['s3n/{bucket}/{key1}', 's3n/{bucket}/{key2}']
# numSlices is the number of partitions. You'll have to set it according to your cluster
configuration and performance requirement
key_rdd = sc.parallelize(s3_keys, numSlices=16)

data_rdd = key_rdd.map(lambda key: (key, download_data_from_custom_api(key)))
```

I recommend to use approach 2 because while working with approach 1, the driver downloads all the data and the workers just process it. This has following drawbacks:

1. You'll run out of memory as data size increases.
2. Your workers will be sitting idle till the data has been downloaded

Read [Getting started with pyspark online](https://riptutorial.com/pyspark/topic/5126/getting-started-with-pyspark): <https://riptutorial.com/pyspark/topic/5126/getting-started-with-pyspark>

Credits

S. No	Chapters	Contributors
1	Getting started with pyspark	Ashutosh , Community , Rahul Lakhanpal