Introduction to Apache Spark / PySpark

As of August 2022, these examples are based on Spark 3.3.0

Reference/API Links

- Apache Spark Quick Start
- PySpark v3.3.0 API
- RDD Programming Guide
- Spark SQL Programming Guide

Double-click (or enter) to edit

```
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
```

```
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
    Requirement already satisfied: pyspark in /usr/local/lib/python3.7/dist-packa
    Requirement already satisfied: py4j==0.10.9.5 in /usr/local/lib/python3.7/dis
    openjdk-8-jdk-headless is already the newest version (8u342-b07-0ubuntu1~18.0
    The following package was automatically installed and is no longer required:
      libnvidia-common-460
    Use 'apt autoremove' to remove it.
    0 upgraded, 0 newly installed, 0 to remove and 20 not upgraded.
```

Imports / Starter Example

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql import types as sparktypes
from pyspark.sql.functions import col, split
import pyspark.sql.functions as pyspark

sc = SparkContext()
spark = SparkSession(sc)
```

```
# create a Resilient Distributed Dataset (RDD) from a sequence of integers perform
```

✓ 0s completed at 9:18 PM

```
# Function to be used in the filter() transformation
def filterSmall(x):
   if x < 20:
      return False
   else:
      return True

# Function to be used in the map() transformation
def mapSquare(x):
    return x*x

# Function to be used in the reduce() action
def reduceSum(x,y):
    return x+y

rdd = sc.parallelize(range(100))         ## create an RDD of 100 numbers from 0 to

#print(rdd.filter(filterSmall).collect())

out1 = rdd.filter(filterSmall).map(mapSquare)  ## perform filter and map transform
out2 = out1.reduce(reduceSum)                  ## perform reduce operation

print(out1.collect())           ## print first output (all numbers less than 20 sq
print(out2)                 ## print second output (sum of all numbers from first
```

```
[400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156
325880
```

```
# download a sample access log for use in demos below
!rm -f apache.access.log
!wget -q https://raw.githubusercontent.com/databricks/reference-apps/master/logs_a
```

Apache HTTP Log Example - Resilient Distributed Dataset (RDD)

A SparkContext instance can be used to create RDDs from various data/files/resources (text files, CSV, Hadoop data files, etc.)

```
# find the top 10 clients, map/reduce style using RDD transformations
access_log_rdd = (sc.textFile("apache.access.log")
                  .map(lambda line: ( line.split(" ")[0], 1 ))  # field 0 = client
                  .reduceByKey(lambda count1, count2: count1 + count2)
                  .sortBy(lambda t: -t[1]))
```

```
print ("Total count of client hostnames:")
print(access_log_rdd.count())

print ("Top 10 client hostnames:")
print(access_log_rdd.take(10))
```

```
Total count of client hostnames:
169
Top 10 client hostnames:
[('64.242.88.10', 452), ('10.0.0.153', 188), ('cr020r01-3.sac.overture.com',
```

Apache HTTP Log Example - DataFrame

A DataFrame is equivalent to a relational table in Spark SQL, and can be created from on a variety of input formats (CSV, JSON, relational database, etc.) using the SparkSession.

```
access_log_df = spark.read.text("apache.access.log")

access_log_df.show(truncate=False)
access_log_df.printSchema()
```

```
+------------------------------------------------------------------------------
|value
+------------------------------------------------------------------------------
|64.242.88.10 - - [07/Mar/2004:16:05:49 -0800] "GET /twiki/bin/edit/Main/Doub
|64.242.88.10 - - [07/Mar/2004:16:06:51 -0800] "GET /twiki/bin/rdiff/TWiki/Ne
|64.242.88.10 - - [07/Mar/2004:16:10:02 -0800] "GET /mailman/listinfo/hsdivis
|64.242.88.10 - - [07/Mar/2004:16:11:58 -0800] "GET /twiki/bin/view/TWiki/Wik
|64.242.88.10 - - [07/Mar/2004:16:20:55 -0800] "GET /twiki/bin/view/Main/DCCA
|64.242.88.10 - - [07/Mar/2004:16:23:12 -0800] "GET /twiki/bin/oops/TWiki/App
|64.242.88.10 - - [07/Mar/2004:16:24:16 -0800] "GET /twiki/bin/view/Main/Pete
|64.242.88.10 - - [07/Mar/2004:16:29:16 -0800] "GET /twiki/bin/edit/Main/Head
|64.242.88.10 - - [07/Mar/2004:16:30:29 -0800] "GET /twiki/bin/attach/Main/Of
|64.242.88.10 - - [07/Mar/2004:16:31:48 -0800] "GET /twiki/bin/view/TWiki/Web
|64.242.88.10 - - [07/Mar/2004:16:32:50 -0800] "GET /twiki/bin/view/Main/WebC
|64.242.88.10 - - [07/Mar/2004:16:33:53 -0800] "GET /twiki/bin/edit/Main/Smtp
|64.242.88.10 - - [07/Mar/2004:16:35:19 -0800] "GET /mailman/listinfo/busines
|64.242.88.10 - - [07/Mar/2004:16:36:22 -0800] "GET /twiki/bin/rdiff/Main/Web
|64.242.88.10 - - [07/Mar/2004:16:37:27 -0800] "GET /twiki/bin/view/TWiki/Don
|64.242.88.10 - - [07/Mar/2004:16:39:24 -0800] "GET /twiki/bin/view/Main/Toky
|64.242.88.10 - - [07/Mar/2004:16:43:54 -0800] "GET /twiki/bin/view/Main/Mike
|64.242.88.10 - - [07/Mar/2004:16:45:56 -0800] "GET /twiki/bin/attach/Main/Po
|64.242.88.10 - - [07/Mar/2004:16:47:12 -0800] "GET /robots.txt HTTP/1.1" 200
|64.242.88.10 - - [07/Mar/2004:16:47:46 -0800] "GET /twiki/bin/rdiff/Know/Rea
+------------------------------------------------------------------------------
only showing top 20 rows

root
 |-- value: string (nullable = true)
 |    value: string (nullable = true)
```

```
access_log_df = spark.read.options(delimiter=" ").csv("apache.access.log")

access_log_df.show(truncate=False)
access_log_df.printSchema()
```

```
+------------+---+---+--------------------+------+------------------------
|_c0         |_c1|_c2|_c3                 |_c4   |_c5
+------------+---+---+--------------------+------+------------------------
|64.242.88.10|-  |-  |[07/Mar/2004:16:05:49|-0800]|GET /twiki/bin/edit/Main/D
|64.242.88.10|-  |-  |[07/Mar/2004:16:06:51|-0800]|GET /twiki/bin/rdiff/TWiki
|64.242.88.10|-  |-  |[07/Mar/2004:16:10:02|-0800]|GET /mailman/listinfo/hsdi
|64.242.88.10|-  |-  |[07/Mar/2004:16:11:58|-0800]|GET /twiki/bin/view/TWiki/
|64.242.88.10|-  |-  |[07/Mar/2004:16:20:55|-0800]|GET /twiki/bin/view/Main/D
|64.242.88.10|-  |-  |[07/Mar/2004:16:23:12|-0800]|GET /twiki/bin/oops/TWiki/
|64.242.88.10|-  |-  |[07/Mar/2004:16:24:16|-0800]|GET /twiki/bin/view/Main/P
|64.242.88.10|-  |-  |[07/Mar/2004:16:29:16|-0800]|GET /twiki/bin/edit/Main/H
|64.242.88.10|-  |-  |[07/Mar/2004:16:30:29|-0800]|GET /twiki/bin/attach/Main
|64.242.88.10|-  |-  |[07/Mar/2004:16:31:48|-0800]|GET /twiki/bin/view/TWiki/
|64.242.88.10|-  |-  |[07/Mar/2004:16:32:50|-0800]|GET /twiki/bin/view/Main/W
|64.242.88.10|-  |-  |[07/Mar/2004:16:33:53|-0800]|GET /twiki/bin/edit/Main/S
|64.242.88.10|-  |-  |[07/Mar/2004:16:35:19|-0800]|GET /mailman/listinfo/busi
|64.242.88.10|-  |-  |[07/Mar/2004:16:36:22|-0800]|GET /twiki/bin/rdiff/Main/
|64.242.88.10|-  |-  |[07/Mar/2004:16:37:27|-0800]|GET /twiki/bin/view/TWiki/
|64.242.88.10|-  |-  |[07/Mar/2004:16:39:24|-0800]|GET /twiki/bin/view/Main/T
|64.242.88.10|-  |-  |[07/Mar/2004:16:43:54|-0800]|GET /twiki/bin/view/Main/M
|64.242.88.10|-  |-  |[07/Mar/2004:16:45:56|-0800]|GET /twiki/bin/attach/Main
|64.242.88.10|-  |-  |[07/Mar/2004:16:47:12|-0800]|GET /robots.txt HTTP/1.1
|64.242.88.10|-  |-  |[07/Mar/2004:16:47:46|-0800]|GET /twiki/bin/rdiff/Know/
+------------+---+---+--------------------+------+------------------------
only showing top 20 rows

root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
```

```
access_log_df = spark.read.options(delimiter=" ").csv("apache.access.log")

named_df = access_log_df.select(col('_c0').alias('host'),
                                col('_c3').alias('timestamp'),
                                col('_c5').alias('path'),
                                col('_c6').cast('integer').alias('status'),
                                col('_c7').cast('integer').alias('content_size'))
```

```
named_df.show(truncate=False)
named_df.printSchema()
```

```
+------------+---------------------+------------------------------------------
|host        |timestamp            |path
+------------+---------------------+------------------------------------------
|64.242.88.10|[07/Mar/2004:16:05:49|GET /twiki/bin/edit/Main/Double_bounce_se
|64.242.88.10|[07/Mar/2004:16:06:51|GET /twiki/bin/rdiff/TWiki/NewUserTemplat
|64.242.88.10|[07/Mar/2004:16:10:02|GET /mailman/listinfo/hsdivision HTTP/1.1
|64.242.88.10|[07/Mar/2004:16:11:58|GET /twiki/bin/view/TWiki/WikiSyntax HTTP
|64.242.88.10|[07/Mar/2004:16:20:55|GET /twiki/bin/view/Main/DCCAndPostFix HT
|64.242.88.10|[07/Mar/2004:16:23:12|GET /twiki/bin/oops/TWiki/AppendixFileSys
|64.242.88.10|[07/Mar/2004:16:24:16|GET /twiki/bin/view/Main/PeterThoeny HTTP
|64.242.88.10|[07/Mar/2004:16:29:16|GET /twiki/bin/edit/Main/Header_checks?to
|64.242.88.10|[07/Mar/2004:16:30:29|GET /twiki/bin/attach/Main/OfficeLocation
|64.242.88.10|[07/Mar/2004:16:31:48|GET /twiki/bin/view/TWiki/WebTopicEditTem
|64.242.88.10|[07/Mar/2004:16:32:50|GET /twiki/bin/view/Main/WebChanges HTTP/
|64.242.88.10|[07/Mar/2004:16:33:53|GET /twiki/bin/edit/Main/Smtpd_etrn_restr
|64.242.88.10|[07/Mar/2004:16:35:19|GET /mailman/listinfo/business HTTP/1.1
|64.242.88.10|[07/Mar/2004:16:36:22|GET /twiki/bin/rdiff/Main/WebIndex?rev1=1
|64.242.88.10|[07/Mar/2004:16:37:27|GET /twiki/bin/view/TWiki/DontNotify HTTP
|64.242.88.10|[07/Mar/2004:16:39:24|GET /twiki/bin/view/Main/TokyoOffice HTTP
|64.242.88.10|[07/Mar/2004:16:43:54|GET /twiki/bin/view/Main/MikeMannix HTTP/
|64.242.88.10|[07/Mar/2004:16:45:56|GET /twiki/bin/attach/Main/PostfixCommand
|64.242.88.10|[07/Mar/2004:16:47:12|GET /robots.txt HTTP/1.1
|64.242.88.10|[07/Mar/2004:16:47:46|GET /twiki/bin/rdiff/Know/ReadmeFirst?rev
+------------+---------------------+------------------------------------------
only showing top 20 rows

root
 |-- host: string (nullable = true)
 |-- timestamp: string (nullable = true)
 |-- path: string (nullable = true)
 |-- status: integer (nullable = true)
 |-- content_size: integer (nullable = true)
```

```
named_df.createOrReplaceTempView("log")
sql_df = spark.sql("SELECT * FROM log WHERE status = 404")

sql_df.show(truncate=False)
sql_df.printSchema()
```

```
+------------------------------+---------------------+------------------------
|host                          |timestamp            |path
+------------------------------+---------------------+------------------------
|h24-70-56-49.ca.shawcable.net|[07/Mar/2004:21:16:17|GET /twiki/view/Main/Web
|61.9.4.61                     |[08/Mar/2004:07:27:36|GET /_vti_bin/owssvr.dll
|61.9.4.61                     |[08/Mar/2004:07:27:37|GET /MSOffice/cltreq.asp
|1513.cps.virtua.com.br        |[11/Mar/2004:02:27:39|GET /pipermail/cipg/2003
|osdlab.eic.nctu.edu.tw        |[11/Mar/2004:07:39:30|GET /M83A HTTP/1.0
+------------------------------+---------------------+------------------------
```

```
root
 |-- host: string (nullable = true)
 |-- timestamp: string (nullable = true)
 |-- path: string (nullable = true)
 |-- status: integer (nullable = true)
 |-- content_size: integer (nullable = true)
```

What About Datasets?

Added in Spark 1.6, a **Dataset** is a distributed collection of data that provides the benefits of RDDs (strong typing, ability to use powerful lambda functions) with the benefits of Spark SQL's optimized execution engine. A Dataset can be constructed from JVM objects and then manipulated using functional transformations (`map`, `flatMap`, `filter`, etc.). The Dataset API is available in Scala and Java. Python does not have the support for the Dataset API. But due to Python's dynamic nature, many of the benefits of the Dataset API are already available (i.e. you can access the field of a row by name naturally row.columnName). The case for R is similar.

A **DataFrame** is a Dataset organized into named columns. (source)

Reporting Tasks (from Lab 2)

1. Most popular URL paths (top 15)
2. Request count for each HTTP response code, sorted by response code
3. Request count for each calendar month and year, sorted chronologically
4. Total bytes sent to the client with a specified hostname or IPv4 address (you may hard code an address)
5. Based on a given URL (hard coded), compute a request count for each client (hostname or IPv4) who accessed that URL, sorted by request count, highest to lowest

(A) RDD Implementations

Perform reporting tasks 1-5 using RDD transformations

RDD APIs PySpark v3.3.0

```
# RDD implementation
# (1) Most popular URL paths (top 15)
# find the top 10 clients, map/reduce style using RDD transformations
access_log_rdd = (sc.textFile("apache.access.log")
                  .map(lambda line: ( line.split(" ")[6], 1 ))  # field 0 = client
                  .reduceByKey(lambda x, y: x + y)
                  .sortBy(lambda t: -t[1]))

print ("Total count of client hostnames:")
print(access_log_rdd.count())

print ("Top 10 client hostnames:")
print(access_log_rdd.take(10))
```

```
Total count of client hostnames:
628
Top 10 client hostnames:
[('/twiki/bin/view/Main/WebHome', 40), ('/twiki/pub/TWiki/TWikiLogos/twikiRob
```

```
# RDD implementation
# (2) Request count for each HTTP response code, sorted by response code

# find the top 10 clients, map/reduce style using RDD transformations
access_log_rdd = (sc.textFile("apache.access.log")
                  .map(lambda line: ( line.split(" ")[6], 1 ))  # field 0 = client
                  .map(lambda s: (s[0][2:5:], s[1])))
                  #.reduceByKey(lambda x, y: x + y)
                  #.sortBy(lambda t: -t[1]))

print ("Total count of client hostnames:")
print(access_log_rdd.count())

print ("Top 10 client hostnames:")
print(access_log_rdd.take(10))
```

```
Total count of client hostnames:
1406
Top 10 client hostnames:
[('wik', 1), ('wik', 1), ('ail', 1), ('wik', 1), ('wik', 1), ('wik', 1), ('wi
```

```
# RDD implementation
# (3) Request count for each calendar month and year, sorted chronologically
from datetime import date

def getYearMonth(line: str):
    str_time = line.split(" ")[3]
    month_str = str_time.split("/")[1]

    if (month_str == "Jan"):
      month = 1
    if (month_str == "Feb"):
      month = 2
    if (month_str == "Mar"):
      month = 3
    if (month_str == "May"):
      month = 5
    if (month_str == "Jun"):
      month = 6
    if (month_str == "Jul"):
      month = 7
    if (month_str == "Aug"):
      month = 8
    if (month_str == "Sep"):
      month = 9
    if (month_str == "Oct"):
      month = 10
    if (month_str == "Nov"):
      month = 11
    if (month_str == "Dec"):
      month = 12

    year = int(str_time.split("/")[2][0:4])

    day = date(year, month, 1)

    return day

access_log_rdd = (sc.textFile("apache.access.log")
                  .map(lambda line : (getYearMonth(line), 1))
                  .reduceByKey(lambda x, y: x + y)
                  .sortBy(lambda x: x[0])
                  .map(lambda data: (str(data[0].year) + '/' + str(data[0].month),

print(access_log_rdd.collect())
```

```
[('2004/3', 1406)]
```

```
# RDD implementation
# (4) Total bytes sent to the client with a specified hostname or IPv4 address (yo

access_log_rdd = (sc.textFile("apache.access.log")
                  .filter(lambda line : line.split(' ')[0] == '10.0.0.153')
                  .map(lambda line : (line.split(' ')[0], int(line.split(' ')[-1])
                  .reduceByKey(lambda x, y: x + y))

print(access_log_rdd.collect())
```

```
[('10.0.0.153', 1200145)]
```

```
# RDD implementation
# (5) Based on a given URL (hard coded), compute a request count for each client (


def filter_url(line: str, target_url: str):
  url = line.split(' ')[6]
  temp_str = url

  if target_url == url:
    return True
  return False

access_log_rdd = (sc.textFile("apache.access.log")
                  .filter(lambda line: filter_url(line, '/'))
                  .map(lambda line: (line.split(' ')[0], 1))
                  .reduceByKey(lambda x, y: x + y)
                  .sortBy(lambda x: -1 * x[1]))

print(access_log_rdd.collect())
```

```
--NORMAL--                                                                   <Esc>
[('66-194-6-70.gen.twtelecom.net', 3), ('66-194-6-79.gen.twtelecom.net', 2),
```

(B) DataFrame Implementations

Perform reporting tasks 1-5 using Spark's DataFrame API

DataFrame API PySpark v.3.1.1

Please Note: Spark SQL **is not** permitted for these exercises. You must use the Spark DataFrame API.

```
# DataFrame implementation
# (1) Most popular URL paths (top 15)


data_frame_query_1 = named_df.withColumn('path', split(named_df.path, ' ')[1])
data_frame_query_11 = data_frame_query_1.groupBy('path').count().sort('count', asc
```

```
+--------------------+-----+
|                path|count|
+--------------------+-----+
|/twiki/bin/view/M...|   40|
|/twiki/pub/TWiki/...|   32|
|                   /|   31|
|        /favicon.ico|   28|
|         /robots.txt|   27|
|         /razor.html|   23|
|/twiki/bin/view/M...|   18|
|/twiki/bin/view/M...|   17|
|/cgi-bin/mailgrap...|   16|
|/cgi-bin/mailgrap...|   16|
|/cgi-bin/mailgrap...|   16|
|/cgi-bin/mailgrap...|   16|
|/cgi-bin/mailgrap...|   16|
|/cgi-bin/mailgrap...|   16|
|/cgi-bin/mailgrap...|   16|
+--------------------+-----+
```

```
# DataFrame implementation
# (2) Request count for each HTTP response code, sorted by response code

data_frame_query_2 = named_df.groupBy('status').count().sort('status', ascending=F
```

```
+------+-----+
|status|count|
+------+-----+
|   404|    5|
|   401|  123|
|   302|    6|
|   200| 1272|
+------+-----+
```

```
# DataFrame implementation
# (3) Request count for each calendar month and year, sorted chronologically


data_frame_query_3 = named_df.withColumn('month_year', pyspark.substring('timestam
```

```
+----------+-----+
|month_year|count|
+----------+-----+
|  Mar/2004| 1406|
+----------+-----+
```

```
# DataFrame implementation
# (4) Total bytes sent to the client with a specified hostname or IPv4 address (yo

given_hostname = '64.242.88.10'
given_hostname = '64.242.88.10'

data_frame_query_4 = named_df.filter(named_df.host == given_hostname).groupBy('hos
```

```
+------------+-----------------+
|        host|sum(content_size)|
+------------+-----------------+
|64.242.88.10|          5745035|
+------------+-----------------+
```

```
# DataFrame implementation
# (5) Based on a given URL (hard coded), compute a request count for each client (

give_url = '/twiki/bin/view'
data_frame_query_5 = named_df.withColumn('path', split(named_df.path, ' ')[1])
data_frame_query_55 = data_frame_query_5.filter(named_df.path.contains(give_url)).
```

```
+--------------------+-----+
|                host|count|
+--------------------+-----+
|        64.242.88.10|  139|
|cr020r01-3.sac.ov...|   25|
|pc3-registry-stoc...|   13|
|ts05-ip44.hevanet...|   13|
|       128.227.88.79|   11|
| prxint-sxb3.e-i.net|   11|
|        212.92.37.62|   11|
|      207.195.59.160|   11|
|mail.geovariances.fr|   10|
|      ogw.netinfo.bg|   10|
|market-mail.pandu...|    9|
|crawl24-public.al...|    8|
|p213.54.168.132.t...|    8|
|      195.246.13.119|    8|
|mth-fgw.ballarat....|    6|
|ip68-228-43-49.tc...|    6|
|c-24-20-163-223.c...|    5|
|user-0c8hdkf.cabl...|    4|
|208-38-57-205.ip....|    4|
|          10.0.0.153|    4|
+--------------------+-----+
only showing top 20 rows
```