rddBasics

February 8, 2026

RDD – Resilient distributed dataset

RDD is the core low-level data structure in Spark that represents an immutable, distributed collection of objects processed in parallel.

Let's break the word:

Term Meaning

Resilient Fault-tolerant (recovers if a node fails)

Distributed Data split across multiple machines

Dataset Collection of records

[1]:
```python
import os

print(os.environ.get("HADOOP_HOME"))
```

```
C:\hadoop
```

[ ]:
```python
from pyspark import SparkContext # this is old way of initializing spark
# old core engine(works with RDD only )

# create SparkContext
sc = SparkContext("local", "ReadCSV") # we can create only once

# read file
d1 = sc.textFile("iris/buy.csv") # reading from a CSV file

print(d1.collect())


# print first line
print(d1.first())

# print only n values
print(d1.take(10))

sc.stop() # close the session
```

```
['age,income,gender,marital,buys', '24,130000,Female,Married,No',
'23,140000,Female,Single,No', '27,150000,Female,Married,Yes',
'51,70000,Female,Married,Yes', '53,50000,Male,Married,Yes',
'56,40000,Male,Single,No', '29,30000,Male,Single,Yes',
'21,80000,Female,Married,No', '19,20000,Male,Single,Yes',
'61,90000,Male,Married,Yes', '17,100000,Male,Single,Yes',
'28,110000,Female,Single,Yes', '31,160000,Male,Married,Yes',
'55,120000,Female,Single,No']
age,income,gender,marital,buys
['age,income,gender,marital,buys', '24,130000,Female,Married,No',
'23,140000,Female,Single,No', '27,150000,Female,Married,Yes',
'51,70000,Female,Married,Yes', '53,50000,Male,Married,Yes',
'56,40000,Male,Single,No', '29,30000,Male,Single,Yes',
'21,80000,Female,Married,No', '19,20000,Male,Single,Yes']
```

[ ]:
```
# new way --> spark Session(modern entry point)
from pyspark.sql import SparkSession

# multiline
# spark = SparkSession.builder \
#     .master("local[*]") \
#     .appName("RDDFromZero") \
#     .getOrCreate()
# config is used to enable/disable features
# appname is temporary (used for debugging ,job identification,logs)

spark=SparkSession.builder.master("local[*]").appName("RDDFromZero").
 ↪getOrCreate()
# here master mentions where spark should run, local means on our own machine
# we may not mention master(it defaultly takes local[*])

# * means uses all cpu cores ,ex: local[2], spark stores our sparksession object

sc=spark.sparkContext # sparkcontext is low level spark engine, used for RDD's

# first RDD
rdd=sc.parallelize([1,2,3,4,5]) # parallelize means split & distribute
print(rdd.collect()) # collect--> brings all data from spark,converts to python␣
 ↪list
# rdd.count()
# rdd.first()
# rdd.take(3) # takes 1st 3 elements

d1 = spark.sparkContext.parallelize([(1, 2), (2, 3), (3, 4), (4, 5), (5, 6)])
# d1.collect()
# d1.first()
# Transformation
rdd2=rdd.map(lambda x:x*2)
print(rdd2.collect())

# filter
even=rdd.filter(lambda x:x%2==0)
even.collect()

# stop spark
# spark.stop() # frees memory,stops background spark
```

```
[1, 2, 3, 4, 5]
[2, 4, 6, 8, 10]
```

[ ]:
```
[2, 4]
```

Read from CSV file

[3]:
```
rdd=sc.textFile('iris/iris.csv') # reads file line by line
print(rdd.take(7))

# remove header
header=rdd.first()
data=rdd.filter(lambda line:line!=header)

# split each line
split_rdd=data.map(lambda line:line.split(","))
print(split_rdd.take(10))

# accessing columns
names=split_rdd.map(lambda row:row[4])
print(names.take(10))

# type conversion
lengths=split_rdd.map(lambda row:float(row[1]))
print(lengths.take(10))

# filter data
above3=split_rdd.filter(lambda row:float(row[1])>3)
print(above3.take(10))
```

```
['Sepal_Length,Sepal_Width,Petal_Length,Petal_Width,Species',
'5.1,3.5,1.4,0.2,setosa', '4.9,3.0,1.4,0.2,setosa', '4.7,3.2,1.3,0.2,setosa',
'4.6,3.1,1.5,0.2,setosa', '5.0,3.6,1.4,0.2,setosa', '5.4,3.9,1.7,0.4,setosa']
[['5.1', '3.5', '1.4', '0.2', 'setosa'], ['4.9', '3.0', '1.4', '0.2', 'setosa'],
['4.7', '3.2', '1.3', '0.2', 'setosa'], ['4.6', '3.1', '1.5', '0.2', 'setosa'],
['5.0', '3.6', '1.4', '0.2', 'setosa'], ['5.4', '3.9', '1.7', '0.4', 'setosa'],
['4.6', '3.4', '1.4', '0.3', 'setosa'], ['5.0', '3.4', '1.5', '0.2', 'setosa'],
['4.4', '2.9', '1.4', '0.2', 'setosa'], ['4.9', '3.1', '1.5', '0.1', 'setosa']]
```

```
['setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
'setosa', 'setosa']
[3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1]
[['5.1', '3.5', '1.4', '0.2', 'setosa'], ['4.7', '3.2', '1.3', '0.2', 'setosa'],
['4.6', '3.1', '1.5', '0.2', 'setosa'], ['5.0', '3.6', '1.4', '0.2', 'setosa'],
['5.4', '3.9', '1.7', '0.4', 'setosa'], ['4.6', '3.4', '1.4', '0.3', 'setosa'],
['5.0', '3.4', '1.5', '0.2', 'setosa'], ['4.9', '3.1', '1.5', '0.1', 'setosa'],
['5.4', '3.7', '1.5', '0.2', 'setosa'], ['4.8', '3.4', '1.6', '0.2', 'setosa']]
```

[ ]:
```python
# RDD Metadata (info about how spark should handle that data)
iris1=sc.textFile('iris/iris_site.csv')
print(iris1.name())

# set rdd name
iris1.setName("import_RD")
print(iris1.name())

rdd=sc.parallelize([1,2,3])
print(rdd.id())
```

```
iris/iris_site.csv
import_RD
```

[ ]:
```python
# store RDD data in server
iris1=sc.textFile('iris/iris_site.csv')
iris1.saveAsTextFile("./output/test1_csv") # windows default native io issue,␣
 ↪so using dataframes would be better
```

[16]:
```python
# if the data is present as an RDD, it can be converted to python
# dictionary object using collectAsMap function

rdd1 = sc.parallelize(
    [
        ("Sepal.Length", [5.1, 4.9, 4.7, 4.6]),
        ("Sepal.Width", [3.5, 3.0, 3.2, 3.1]),
        ("Species", ["setosa", "setosa", "setosa", "setosa"]),
    ]
)
print(rdd1)
print(rdd1.collect())

# convert to dict
dict1=rdd1.collectAsMap()
print(dict1)
print(dict1['Sepal.Length'])
print(dict1['Species'])
```

```
ParallelCollectionRDD[44] at readRDDFromFile at PythonRDD.scala:299
[('Sepal.Length', [5.1, 4.9, 4.7, 4.6]), ('Sepal.Width', [3.5, 3.0, 3.2, 3.1]),
('Species', ['setosa', 'setosa', 'setosa', 'setosa'])]
{'Sepal.Length': [5.1, 4.9, 4.7, 4.6], 'Sepal.Width': [3.5, 3.0, 3.2, 3.1],
'Species': ['setosa', 'setosa', 'setosa', 'setosa']}
[5.1, 4.9, 4.7, 4.6]
['setosa', 'setosa', 'setosa', 'setosa']
```

[ ]:
```
# # creating a rdd
# from pyspark.sql import SparkSession

# spark=SparkSession.builder.appName("Python Spark create RDD example").
 ↪config("random-config","some-value").getOrCreate()

# df=spark.sparkContext.parallelize([(1,2,3,'a b c'),
#                                    (4,5,6,'d e f'),
#                                    (7,8,9,'g h i')]).
 ↪toDF(['col1','col2','col3','col4'])

# df.show()

# df.collect()
```

```
+----+----+----+-----+
|col1|col2|col3| col4|
+----+----+----+-----+
|   1|   2|   3|a b c|
|   4|   5|   6|d e f|
|   7|   8|   9|g h i|
+----+----+----+-----+
```

[ ]:
```
[Row(col1=1, col2=2, col3=3, col4='a b c'),
 Row(col1=4, col2=5, col3=6, col4='d e f'),
 Row(col1=7, col2=8, col3=9, col4='g h i')]
```

[ ]:
```
# d2=spark.createDataFrame([(1,'Joe','70000','1'),
#                           ('2','Henry','80000','2'),
#                           ('3','Sam','60000','1'),
#                           ('4','Max','90000','1')],
#                           ['ID','Name','Salary','DeptId'])

# d2.show()
```

```
+---+-----+------+------+
| ID| Name|Salary|DeptId|
+---+-----+------+------+
|  1|  Joe| 70000|     1|
|  2|Henry| 80000|     2|
|  3|  Sam| 60000|     1|
|  4|  Max| 90000|     1|
+---+-----+------+-------+
```