5d_transformations_mappartions

February 21, 2022

1 mapPartitions

- Runs on different partitions of a RDD
- can be used to find the number of words in each partition using a function passed to it.

The “mapPartitions” is like a map transformation but runs separately on different partitions of a RDD. So, for counting the frequencies of words ‘cdac’ and ‘dbda’ in each partition of RDD, you can follow the steps.

1. Create a function called “count” which will count the frequencies for these words

2. Then, pass the function defined in step1 to the “mapPartitions” transformation.

[1]:
```
# Import SparkContext and SparkConf
from pyspark import SparkContext, SparkConf

# Initialize spark
conf = SparkConf().setAppName("mapPartitions") # AppName can be any name
sc = SparkContext(conf=conf)
```

```
22/02/21 12:13:29 WARN Utils: Your hostname, ThinkCentre resolves to a loopback
address: 127.0.1.1; using 10.180.5.223 instead (on interface eno1)
22/02/21 12:13:29 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
22/02/21 12:13:30 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
22/02/21 12:13:31 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
22/02/21 12:13:31 WARN Utils: Service 'SparkUI' could not bind on port 4041.
Attempting port 4042.
22/02/21 12:13:31 WARN Utils: Service 'SparkUI' could not bind on port 4042.
Attempting port 4043.
22/02/21 12:13:31 WARN Utils: Service 'SparkUI' could not bind on port 4043.
Attempting port 4044.
```

Example 1: Find out the frequency of the words “to”, “the”

[2]:
```
# Example 1 - Count the number of "to" and number of "the"
def count(iterator):
    count_to = 0
    count_the = 0
    for i in iterator:
        if i =='to':
            count_to = count_to + 1
        if i == 'the':
            count_the = count_the + 1
    return (count_to,count_the)
```

[3]:
```
#wordsRDD = sc.textFile("5b_mydependence.txt")
words = ["to", "the", "and", "am", "the", "to", "the", "the"]
wordsRDD= sc.parallelize(words, 2)
```

[4]:
```
wordsRDD.mapPartitions(count).glom().collect()
#wordsRDD.mapPartitions(count).collect()
```

[4]:
```
[[1, 1], [1, 3]]
```

[ ]:
```
# I have used the "glom" function which is very useful
# when we want to see the data insights for each partition of a RDD.
# So above result shows that 1,1 are the counts of ‘to’, ‘the’
# in partition1 and 1,3 are the counts of ‘to’, ‘the’ in partition2
```