mapPartitionsWithIndex Transformation

Here, we are applying the mapPartitionsWithIndex Transformation to same data as used in mapPartitions Transformation code but there is an another example in the last block.

In [1]:
```
import pyspark

from pyspark.sql import SparkSession

#Create the Spark Session with 4 partitions with master("local[4]")
spark = SparkSession.builder \
    .master("local[4]") \
    .appName('mapPartitionsWithIndex Transformation') \
    .getOrCreate()

#Create an rdd with integers in the range of 1 to 1000
rdd = spark.sparkContext.range(1,1000)

#Printing the count of elements in RDD
print('data count =', rdd.count())

#Check the number of Partitions in the RDD
print("Number of Partitions = ", rdd.getNumPartitions())
```

```
22/10/11 22:40:09 WARN Utils: Your hostname, Vaishalis-MacBook-Pro.local resolves to a loopback address: 127.0.
0.1; using 192.168.0.105 instead (on interface en0)
22/10/11 22:40:09 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/10/11 22:40:10 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builti
n-java classes where applicable
22/10/11 22:40:11 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
22/10/11 22:40:11 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
[Stage 0:>                                                          (0 + 4) / 4]
data count = 999
Number of Partitions =  4
```

In [2]:
```
#Using a function to get the minimum and maximum values of each partition in the input RDD.
#The difference is index is also passed to the function.
def minmax_partition(index, iterator):
    a = True
    for x in iterator:
        if(a):
            local_min = x;
            local_max = x;
            a = False
        else:
            local_min = min(x, local_min)
            local_max = max(x, local_max)
    yield (index, local_min, local_max)

#Using the list of Minimum and Maximum values of each partition
#With above obtained data this Function is to find minimum and maximum of all of RDD
def minmax_numbers(list):
    minimum = []
    maximum =[]

    for element in list:
        minimum.append(element[1])
        maximum.append(element[2])

    return(min(minimum), max(maximum))


#mapPartitions gives out the List of Minimum and Maximum values of each partition as well as their index value
minmax_rdd = rdd.mapPartitionsWithIndex(minmax_partition)
print("List of Minimum and Maximum values of each partition = ", minmax_rdd.collect())

minmax_partition_list = minmax_rdd.collect()

#Minimum and Maximum values of the RDD
minmax_list = minmax_numbers(minmax_partition_list)
print("Minimum value of the list = ", min(minmax_list))
print("Maximum value of the list = ", max(minmax_list))
```

```
List of Minimum and Maximum values of each partition =  [(0, 1, 249), (1, 250, 499), (2, 500, 749), (3, 750, 99
9)]
Minimum value of the list =  1
Maximum value of the list =  999
```

In [5]:
```
#Another example of mapPartitionsWithIndex

"""
There is a text file in rdd-mapPartitionsWithIndex Folder.
We are reading this file and giving the partition count as 4.
Using a custom function process_partition_size, we are counting the length of each element in each partition
of the input RDD.

"""

#Custom Function
def process_partition_size(index, iterator):
    mylist = []
    for element in iterator:
        mylist.append(len(element))
    yield (index, mylist)

#Reading a text file
input_file_path = "/Users/vaishaliyasala/Desktop/Github/Spark/rdd-mapPartitionsWithIndex/file.txt"
files_overview_rdd = spark.sparkContext.textFile(input_file_path, 4)

print("Printing files_overview_rdd: ", files_overview_rdd.collect())

print("Get Partition Count: ", files_overview_rdd.getNumPartitions() )

#Using mapPartitionsWithIndex()
files_overview_words_rdd = files_overview_rdd.mapPartitionsWithIndex(process_partition_size)

for word in files_overview_words_rdd.collect():
    print(word)
```

```
Printing files_overview_rdd:  ['mapPartitionsWithIndex(func)', 'Similar to mapPartitions, but also provides fun
c ', 'with an integer value representing the index of ', 'the partition, so func must be of ', 'type (Int, Iter
ator<T>) => Iterator<U> ', 'when running on an RDD of type T.']
Get Partition Count:  4
(0, [28, 49])
(1, [48])
(2, [34, 39])
(3, [33])
```