National University of Computer and Emerging Sciences

Laboratory Manual-09

for

Fundamentals of Big Data Lab

Course Instructor: Dr. Iqra Safdar
Lab Instructors: Mr. Muhammad Mazarib; Mr Muhammad Aiss Shahid
Section: BDS-4B
Date: 12-Apr-2023
Semester: Spring 2023

Department of Computer Science

FAST-NU, Lahore, Pakistan

Big Data processing systems

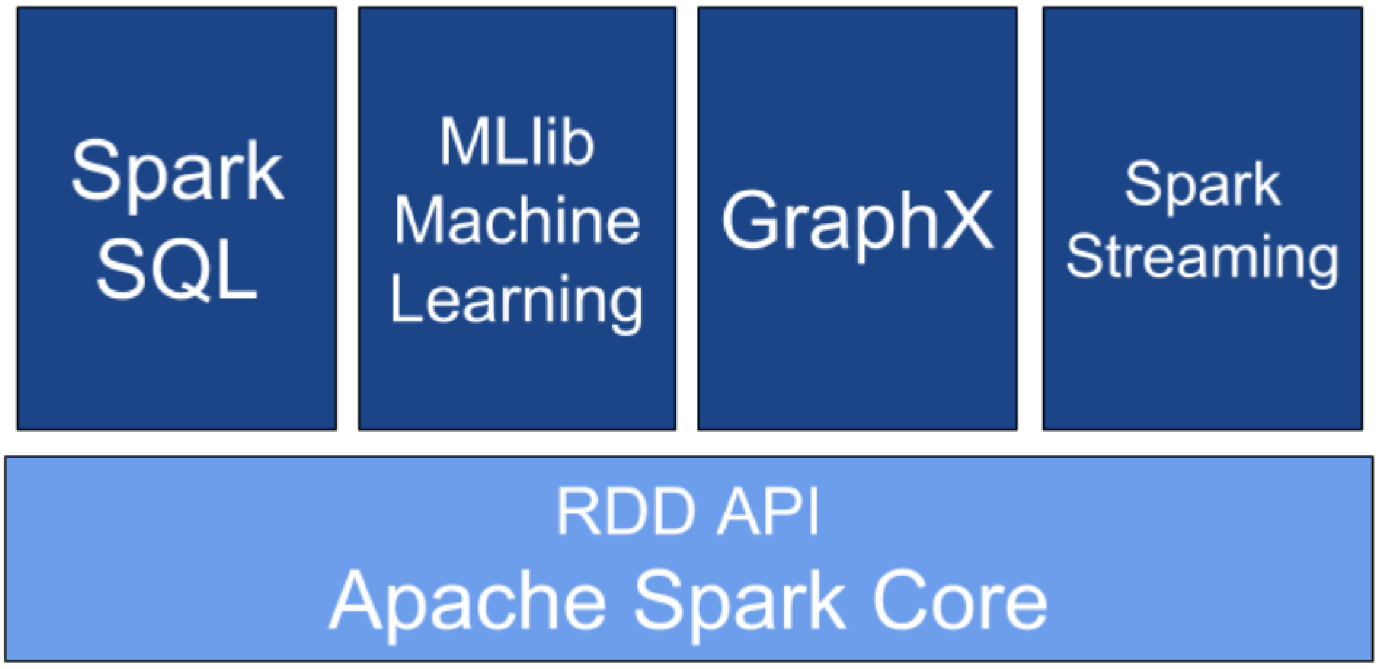
Hadoop/MapReduce:

- Scalable and fault tolerant framework written in Java
- Open source
- Batch processing

Apache Spark:

- General purpose and lightning fast cluster computing system
- Open source
- Both batch and real-time data processing

Apache Spark Components



Spark modes of deployment

Local mode: Single machine such as your laptop.

- Local model convenient for testing, debugging and demonstration
- Cluster mode: Set of pre-defined machines
- Good for production

Overview of PySpark

- Apache Spark is written in Scala
- To support Python with Spark, Apache Spark Community released PySpark
- Similar computation speed and power as Scala
- PySpark APIs are similar to Pandas and Scikit-learn

PySpark Documentation Link : https://spark.apache.org/docs/3.3.2/
Pyspark RDD Documentation Link: https://spark.apache.org/docs/latest/rdd-programming-guide.html

Note: Use google colab or jupyter notebook for PySpark

Configuration of PySpark in System

- Install pyspark using the line: !pip install pyspark
- Import the following library:
  ```
  from pyspark import SparkContext, SparkConf
  ```
- Configure the PySaprk and start the session:
  ```
  conf = SparkConf().setAppName(appName).setMaster(master)
  sc = SparkContext(conf=conf)
  ```

  where appName is your name of your project/lab and "local[*]" is your master if you are working locally.

Understanding SparkContext

A SparkContext represents the entry point to Spark functionality. It's like a key to your car. When we run any Spark application, a driver program starts, which has the main function and your SparkContext gets initiated here.

Use of Lambda function in python - filter()

What are anonymous functions in Python?

- Lambda functions are anonymous functions in Python
- Very powerful and used in Python. Quite efficient with map() and filter()
- Lambda functions create functions to be called later similar to def
- It returns the functions without any name (i.e. anonymous)
- Inline a function definition or to defer execution of a code

Lambda function syntax

- The general form of lambda functions is
  ```
  lambda arguments: expression
  ```
- Example of lambda function is as follow:
  ```
  double = lambda x: x * 2
  print(double(3))
  ```

Difference between def vs lambda functions

- Python code to illustrate cube of a number
  ```
  def cube(x):
  return x ** 3

  g = lambda x: x ** 3
  print(g(10))
  print(cube(10))
  ```

- No return statement for lambda
- Can put lambda function anywhere

Use of Lambda function in Python - map()

- map() function takes a function and a list and returns a new list which contains items returned by that function for each item
- General syntax of map()
  ```
  map(function, list)
  ```
- Example of map()
  ```
  items = [1, 2, 3, 4]
  list(map(lambda x: x + 2 , items))
  ```

Use of Lambda function in python - filter()

- filter() function takes a function and a list and returns a new list for which the function evaluates as true
- General syntax of ,filter()
  ```
  filter(function, list)
  ```
- Example of ,filter()
  ```
  items = [1, 2, 3, 4]
  list(filter(lambda x: (x%2 != 0), items))
  ```

foreach() loop:

Returns only those elements which meet the condition of the function inside foreach. In the following example, we call a print function in foreach, which prints all the elements in the RDD.

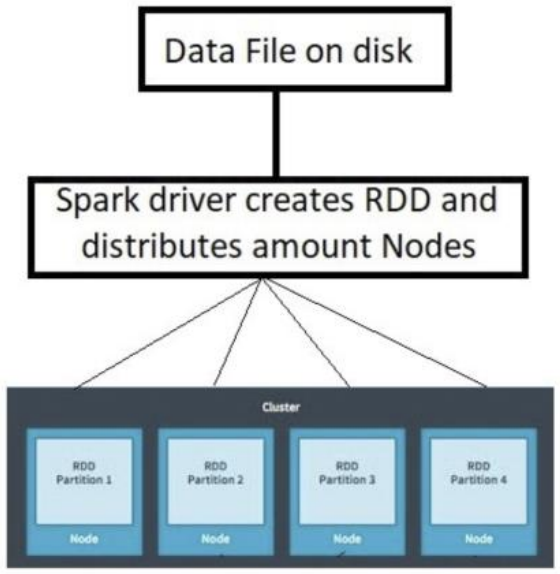
```
words = sc.parallelize (["scala","java", "hadoop", "spark","akka" ,"spark vs hadoop", "pyspark","pyspark and spark"])

def f(x): print(x)

fore = words.foreach(f)
```

What is RDD?

- RDD = Resilient Distributed Datasets



Creating RDDs. How to do it?

- Parallelizing an existing collection of objects
- External datasets:
  - Files in HDFS
  - Objects in Amazon S3 bucket
  - lines in a text ,file

- From existing RDDs

Parallelized collection (parallelizing)

- parallelize() for creating RDDs from python lists
```
 numRDD = sc.parallelize([1,2,3,4])
helloRDD = sc.parallelize("Hello world")
type(helloRDD)
```

- From external datasets
  textFile() for creating RDDs from external datasets
```
fileRDD = sc.textFile("README.md")
type(fileRDD)
```

Understanding Partitioning in PySpark

- A partition is a logical division of a large distributed data set
- parallelize() method
```
numRDD = sc.parallelize(range(10), minPartitions = 6)
```
- textFile() method
```
fileRDD = sc.textFile("README.md", minPartitions = 6)
```
- The number of partitions in an RDD can be found by using getNumPartitions() method

Overview of PySpark operations

- Transformations create new RDDS
- Actions perform computation on the RDDs

RDD Transformations

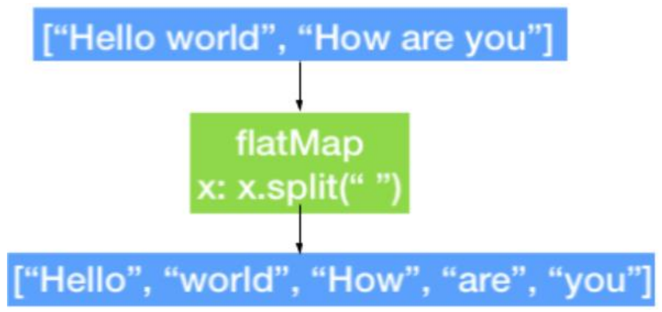
- Transformations follow Lazy evaluation
- Basic RDD Transformations are map() , filter() , flatMap() , and union()

map() Transformation: map() transformation applies a function to all elements in the RDD
```
RDD = sc.parallelize([1,2,3,4])
RDD_map = RDD.map(lambda x: x * x)
```

filter() Transformation: Filter transformation returns a new RDD with only the elements that pass the condition.
```
RDD = sc.parallelize([1,2,3,4])
RDD_filter = RDD.filter(lambda x: x > 2)

inputRDD = sc.textFile("logs.txt")
errorRDD = inputRDD.filter(lambda x: "error" in x.split())
```

flatMap() Transformation: flatatMap() transformation returns multiple values for each element in the original RDD


["Hello world", "How are you"]
flatMap
x: x.split(" ")
["Hello", "world", "How", "are", "you"]

```
RDD = sc.parallelize(["hello world", "how are you"])
RDD_flatmap = RDD.flatMap(lambda x: x.split(" "))
```

RDD Actions

- Operation return a value after running a computation on the RDD
- Basic RDD Actions are collect(),take(N),first(),count()

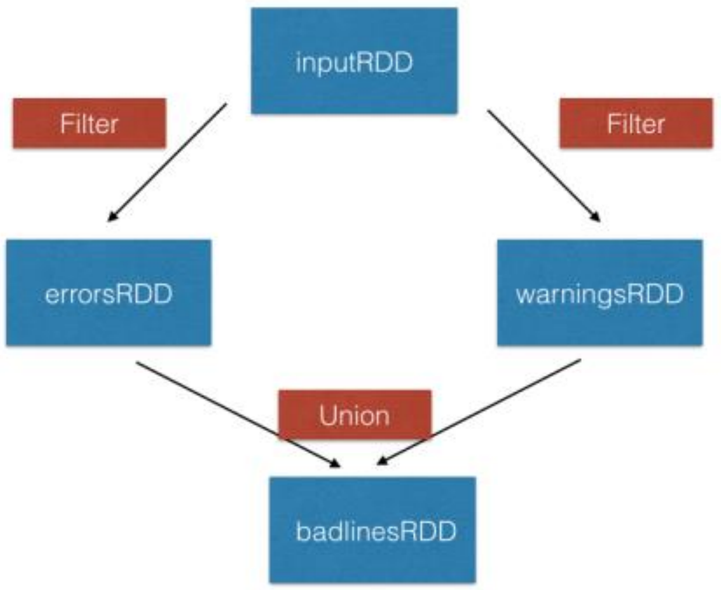
collect() and take() Actions

- collect() return all the elements of the dataset as an array
- take(N) returns an array with the first N elements of the dataset

first() and count() Actions

- first() prints the first element of the RDD
- count() return the number of elements in the RDD

union() Transformation



```
inputRDD = sc.textFile("logs.txt")
errorRDD = inputRDD.filter(lambda x: "error" in x.split())
warningsRDD = inputRDD.filter(lambda x: "warnings" in x.split())
combinedRDD = errorRDD.union(warningsRDD)
```

LAB TASKS

1. Create my_list which contains number from 1 to 10. Print the list . Square each item in my_list using map() and lambda(). Print the result of map function.
2. Create my_list_2 which contains 20 random numbers. Print the list. Filter the numbers divisible by 5 from my_list2 using filter() and lambda().Print the numbers divisible by 5 from my_list2.
3. Create an RDD named RDD from a list of words which is created by yourself. Confirm the object created is RDD.
4. Create an RDD named fileRDD from a given input file. Print the type of the fileRDD created.
5. Find the number of partitions that support fileRDD RDD. Create an RDD named fileRDD_part from the input file but create 5 partitions. Confirm the number of partitions in the new fileRDD_part RDD.
6. Create map() transformation that cubes all of the numbers in numbRDD. Collect the results in a numbers_all variable. Print the output from numbers_all variable.
7. Create filter() transformation on RDD which reads the input file to select the lines containing the keyword beautiful. How many lines in fileRDD_filter contains the keyword beautiful. Print the first four lines of the resulting RDD.
8. Read a string in a list and and store it in RDD and print each word separately using flatmap.
9. You have a list of ten random numbers find the highest number from the list.
10. Compare two lists of names such a way that they are stored in RDDs and in the end print the matching elements of the lists.