**Spark with python**

* Framework for distributed computing
* Spark created in scalar
* Platform independent through java
* JVM runs scalar
* Requirements:
  + Java
  + Python
  + Spark
* RDD –Resilient Distributed Dataset
* **Driver program is main (entry point)**
* Two modes – Interactive and Batch

Worker Node

Worker Node

Worker Node

Cluster Manager

Master Node

* Interactive mode spark context is already active
* Spark context object creates RDD.

>>> RDD1 = sc.parallelize([1,2,3,4,5,6,7])

* Above command creates RDD with list data stored
* Two types of **RDD operations**
  + Transformation
    - Will create and return new RDD
  + Action
    - Will return results/values from RDD
* Transformations only execute once an action is executed
  + Queuing transformations execute in order before the action is processed.
  + “Lazy operations”

>>> data = RDD1.collect()

>>> print(data)

**//Creates Spark context object**

**From pyspark import SparkConf, SparkContext**

**con = Spark.conf()**

**sc = SparkContext(conf = con)**

list1 = [74,3,47,92,975]

RDD1 = sc.parallelize(list1)

Data = RDD1.collect()

For A in data:

print(A)

**Transformation**

Map()

//function

Def square1 (x)

Return x\*x

…..

rdd1.map(square1) //will pass rdd1 values to the square1 function and create a new rdd of the results.

Function will split a string where each ‘,’ is and store the sections in the ‘record’ list.

Def recordFunction(Rec):

record = Rec.split(“,”)

return (record[0],record[3] //function will return the first and 4th element of the new list

def add(a,b)

return a+b

addition = rdd1.reduce(add) //Recursively adds values to the returned value (result of (a+b)+b)

**Saving RDDs**

List1= [1,2,3,4,5,6,7,8,9,10,11,12,]

Rdd1 – sc.parallelize(**List1**,**3**)//3 specifies number of partitions

Rdd1.saveFile(“c:\\ABC”) //Specify path to save at

RDD Function

Rdd1.subtract(Rdd2) // Removes elements from rdd1 that exist in rdd2

Rdd1.union(Rdd2) // Combines rdd’s (no duplicates generated)

Rdd1.intersect(Rdd2) // Result is records that exist in both Rdd’s

**DataFrames**

RDD1 = sc.testFile(“FILENAME.TXT”)

RDD2 = RDD1.Map(lambda x: x.split(“,”))

Sql = sqlContext(sc)

df = sql.createDataFrame(RDD2)

df.show()