Introduction to PySpark

» PySpark is an apache spark library written in Python to run Python applications using Apache Spark capabilities.

» It is basically a Python API which is an analytical processing engine for large-scale powerful distributed data processing and ML applications.

What is Apache Spark?

- It is an open-source unified analytics engine used for large-scale data processing.
- It runs operations on billions and trillions of data on distributed clusters 100 times faster than traditional applications.
- It uses in-memory processing which solves the limitations of map Reduce
- It is a multi-language engine → It provides APIs and libraries for several programming lang. like scala, java, python etc.

Who Uses PySpark?

- It is used in Data Science, Machine learning, Numpy, Tensor flow.
- Organizations like Sanofi, Walmart etc.
- Used for development.
- Used in many ML applications.

Features of Pyspark
- In memory Computation
- It is immutable
- Distributed processing and parallelize
- Fault-tolerant.
- Can be used with many cluster managers.
- Lazy evaluation.

Advantages of PySpark
- It processes data efficiently in distributed fashion.
- Used for data ingestion pipelines.
- Can process data from Hadoop HDFS, AWS S3, and many files systems using pyspark.
- Used to process realtime data using Streaming and Kafka

PySpark supports 3.5 and that is compatible with Python 3.8, R 3.5, Java 8, 11, 13, 17 and later, Scala 2.12 and beyond.

PySpark Architecture.
- It works in master-slave architecture
  driver → workers.

- Spark Driver creates a context that is entry point to our application.

- All operations are created on worker nodes.
- Resources are managed by cluster Manager.

Cluster-Manager Types:-

1. Standalone
2. Mesos
3. Hadoop Yarn
4. Kubernetes

Modules and Packages:-

1. PySpark RDD (pyspark.RDD)
2. PySpark Dataframe and SQL (pyspark.sql)
3. PySpark Streaming (pyspark.streaming)
4. PySpark MLib (pyspark.ml, pyspark.mllib)
5. PySpark GraphFrames (GraphFrames)
6. PySpark Resource (pyspark.resource)

Running on Local - Scala.

Read the data from the file

① Create a text file.
② Copy file path
③ Cmnd :- val filePath = "... - Path ..."
④ Read file into an RDD
⑤ Collect and print file contents.

Creating RDD's

→ There are 2 ways.-
① Loading an external dataset
② Distributing an set of collection of objects.

① By using parallelize() function:-

```
from pyspark.sql import SparkSession.

spark = SparkSession \
    .builder \
    .appName("Python Spark create RDD example") \
    .config("spark.some.config.option", "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'])
```

SparkContext has several functions to use with RDDs.
For eg:-

1. parallelize() method is used to create an RDD from a list.

datalist = [("Java", 20000), ("Python", 100000), ("Scala", 3000)]
rdd = spark.sparkContext.parallelize(datalist)

2. Using textFile()

rdd = spark.sparkContext.textFile("path/test.txt").