

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, TensorFlow.
- Organizations like Sanofi, Walmart etc.
- Used for development.
- Used in many ML applications.

Features of PySpark

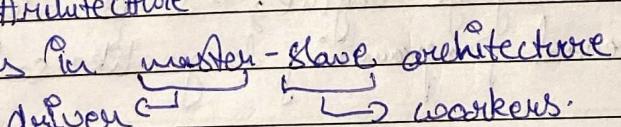
- In memory computation
- It is immutable
- Distributed processing and parallelize
- Fault-tolerant.
- Can be used with many cluster managers.
- Very fast evaluation.

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 file systems using PySpark.
- Used to process real-time data using streaming and kafka.

PySpark supports 3.5 and that is compatible with Python 3.8, 3.9, Java 8, 11, 13, 17 and later, Scala 2.12 and beyond.

PySpark Architecture.

- It works in master-slave architecture.

- 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 MLlib (`pyspark.ml, pyspark.mllib`)
- ⑤ PySpark GraphFrames (`graphframes`)
- ⑥ PySpark Resource (`pyspark.resource`)

running on Local-Scale.

Read the data from the file

① Create a text file.

② Copy file path

cmd := 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 \

• build()

• 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~~ an RDD from a list.

datalist = [("Java", 2000), ("Python", 10000), ("Scala", 3000)]
rdd = spark.sparkContext.parallelize(datalist)

- ② Using textFile()

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