05/2/24

Introduction to Pyspark

* Pyspark is an Apache spark library written in Python to run python applns using Apache spark capabilities
* Using Pyspark we can run applns parallely on distributed cluster (multiple nodes)
* Pyspark is python API ⇒ analytical processing engine for large-scale powerful distributed data processing & ML applns.

Apache Spark :-

* open-source unified analytics engine ⇒ used for large scale data processing ⇒ known as spark.

* spark
  - → fast
  - → flexible
  - → easy to use
  - → processing large-scale data sets.
  - → runs operations on billions & trillions of data on distributed clusters.
  - → 100 times faster than traditional applns
  - → can run on single-node machines or multi node machines
  - → uses in-memory processing (solves the limitations of Map Reduce).
  - → process realtime streaming
  - → multi-language engine

Multilanguage engine

It provides APIs and libraries for several programming languages like Scala, Java, Python, R.

* Spark offers
  - → Scala
  - → Java API
  - → Python API
  - → R API

Who uses PySpark?

- → Data Science
- → Machine Learning
- → Numpy, Tensor flow
- → since it efficiently processes large datasets, it is used by many organizations → walmart, Trivago, sanofi etc.,

* For development, we can use Anaconda distribution which contains many useful tools like 1) Jupyter Notebook 2) Spyder IDE. to run Pyspark applns.

* used in many machine learning applns.

Features of Pyspark

1) In-memory computation
2) Distributed processing using parallelize
3) can be used with many cluster managers (Spark, Yarn, Mesos etc.,)
4) Fault-tolerant
5) Immutable
6) Lazy evaluation

7) cache & persistence

8) Inbuild - optimization when using Dataframes

9) Supports ANSI SQL.

Advantages of PySpark

1) process data efficiently in distributed fashion

2) Applns running on pyspark - 100 times faster than traditional systems.

3) used for data ingestion pipelines

4) we can process data from Hadoop HDFS, AWS S3 and many file systems using pyspark.

Kafka - open source distributed event stream platform used by thousands of companies for high performance datapipelines, streaming analytics

5) used to process real-time data using streaming and Kafka.

6) Using pyspark streaming, we can stream files from file system and also from socket.

7) has machine learning & graph libraries.

Version Python PySpark supports

PySpark 3.5 is compatible with
- → Python 3.8 & above
- → R 3.5
- → Java 8, 11, 13, 17 & later
- → Scala 2.12 & 2.13 beyond

deprecated => no longer used.

PySpark Architecture

Apache Spark works in a master-slave architecture

Master => driver

Slaves => workers.

* When we run a Spark appln, Spark Driver creates a context that is entry point to our application.

* All operations (transformations & actions) are executed on worker nodes.

* Resources are managed by Cluster Manager.

Cluster Manager Types

Spark supports below cluster managers :-

1) Standalone :- a simple cluster manager included with Spark that makes it easy to ~~ore~~ set up a cluster.

2) Mesos - It is a cluster manager that can also run Hadoop, Map Reduce and PySpark applns.

3) Hadoop Yarn - resource manager in Hadoop 2. mostly used as cluster manager.

4) Kubernetes - open-source system for automating deployment, scaling.

Local => not a cluster manager => used for master() to run Spark on our laptop/PC

Modules & 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) ⇒ new in PySpark 3.0

Read the data of file :-

⇒ Create RDD :-

There are two ways to create RDD
   (i) loading an external dataset
   (ii) distributing a set of collection of objects.

Parallelize() method

It takes an already existing collection in our program and pass the same to spark Content.

Ex:-

```
① from pyspark.sql import SparkSession
   # create SparkSession
   spark = SparkSession.builder \
           ~~.master("local[1]")~~ \
           .appName("Spark-Examples") \
           .getOrCreate()
   # using parallelize ()
```

Create RDD from parallelize

```
datalist = [("Java",20000), ("Python",10000),
            ("Scala", 3000)]

df ~~rdd~~ = spark.sparkContext.parallelize(datalist)
```

use ~~show~~ collect () method

```
~~rdd~~ ~~df.show()~~ df.collect() .
```

2) using csv file

```
* from pyspark.sql import SparkSession
( spark = SparkSession.builder.appName("Employee
          Details").getOrCreate()

  spark
  df = spark.read.csv(" file path with \\ ")
  df
  df.show()
```

Find type of ~~dataf~~ df1

```
df1 = spark.read.csv(" file path \\", header = True,
                                inferSchema = True)

df1.show()
print(type(df1))  // dataframe
```

Returning top 2 records

```
df1.head(2)
```

Printing schema

```
df1.printSchema()
```

printing columns

```
df1.columns
```

③ using text file

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Data from
            text file").getOrCreate()

spark
df = spark.SparkContext.textFile(" path ")
df
df.collect()
```