**ASSIGNMENT NO. 06**

**Name – Atharva Shivaji Jadhav Roll Number - DXC262AB12035**

**Batch – Dxc-262-Analytics-B12-Azure Company – DXC Technology**

**Employee Domain –Azure Analytics Training Under – Manipal Pro Learn**

**Trainer Name – Mr. Ajay Kumar**

**Q1- 1. Explain what is in-Memory computation in details?2. Explain advantages of Spark framework ?**

Processing in memory is one approach to overvome the von Neumann bottleneck which is a limitation on throughput caused by latancy inherent in standard computer architecture.

**In-Memory** computing primarily relies on keeping data in servers RAM as a means of processing at faster speeds

**In-memory** computing erpecially applies to processing problems that require extensive access to data-- analytics, reporting or data warehousing and big data applications.

**Q2-Advantages of spark framework—**

1.Fast- provides high performance both batch and streaming data

If hadoop is 1x faster, apache spark is 100x faster

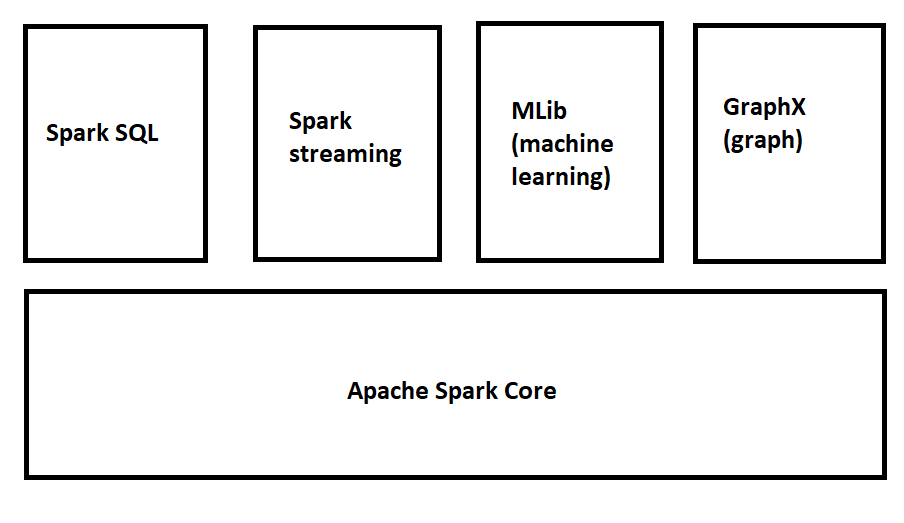
2. Easy to use- Supports vast variety of programming languages

3.supports various libraries-Collection of libraries including SQL, dataframes, MLlib, Graphx and sparkStreaming

4.Supports realtime streaming

5.Lightweight-has light unified analytics engine used for large scale data analysis

**Q3. Explain components of Spark with block diagram ?**

****

**1**.**SparkSQL** – The Spark SQL is built on the top of Spark Core. It provides support for structured data. It allows to query the data via SQL, It also supports various sources of data like Hive tables, Parquet, and JSON.

**2. Spark Streaming** is a Spark component that supports scalable and fault-tolerant processing of streaming data. The log files generated by web servers can be considered as a real-time example of a data stream.

**3. The MLlib** is a Machine Learning library that contains various machine learning algorithms.

**4. The GraphX** is a library that is used to manipulate graphs and perform graph-parallel computations.

It facilitates to create a directed graph with arbitrary properties attached to each vertex and edge.

**4. Explain benifits of in-Memory computation ?**

* Better, faster, decision making.
* Ability to reduce cost.
* Identify competitive opportunities
* Grow revenue
* More efficient application
* Reduce risk—It's best suited for performing real-time analytics, and developing and deploying real-time applications
* In-memory Computing Imperative: Avoid movement of detailed data.

Calculate first, then move the results

**5. Explain major difference between Hadoop & Spark ?**

|  |  |
| --- | --- |
| **Hadoop** | **Spark** |
| 1. Hadoop is an open source framework which uses a MapReduce algorithm | 1. Spark is lightning fast cluster computing technology, which extends the MapReduce model to efficiently use with more type of computations. |
| 1. Spark is lightning fast cluster computing technology, which extends the MapReduce model to efficiently use with more type of computations. | 1. Spark reduces the number of read/write cycles to disk and store intermediate data in-memory, hence faster-processing speed. |
| 1. Hadoop is designed to handle batch processing efficiently | 1. Spark is designed to handle real-time data efficiently. |
| 1. Hadoop is a high latency computing framework, which does not have an interactive mode | 1. Spark is a low latency computing and can process data interactively. |
| 1. With Hadoop MapReduce, a developer can only process data in batch mode only | 1. Spark can process real-time data, from real time events like twitter, facebook |

**6. Explain features of Spark?**

**1.Fast**- provides high performance both batch and streaming data

If hadoop is 1x faster, apache spark is 100x faster

**2. Easy to use**- Supports vast variety of programming languages

3.supports various libraries-Collection of libraries including SQL, dataframes, MLlib, Graphx and spark Streaming

**4.Supports realtime streaming**

**5.Lightweight-** has light unified analytics engine used for large scale data analysis

**6.Write a Py-Spark program to create Dataframe from RDD & explain with screenshots& steps ?**

**#craete pyspark dataframe from RDD consisting of a list of tuples**

rdd = spark.sparkContext.parallelize([

(1,2.,'string1', date(2022,6,6), datetime(2022,6,6,12,30)),

(2,3.,'string2', date(2022,7,6), datetime(2022,6,7,12,30)),

(3,4.,'string3', date(2022,8,6), datetime(2022,6,8,12,30)),

])

df=spark.createDataFrame(rdd, schema=['a','b','c','d','e'])

df

df.show()

step1—

pip install pyspark

step 2—

from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

step3—

#craete pyspark dataframe from RDD consisting of a list of tuples

rdd = spark.sparkContext.parallelize([

      (1,2.,'string1', date(2022,6,6), datetime(2022,6,6,12,30)),

      (2,3.,'string2', date(2022,7,6), datetime(2022,6,7,12,30)),

      (3,4.,'string3', date(2022,8,6), datetime(2022,6,8,12,30)),

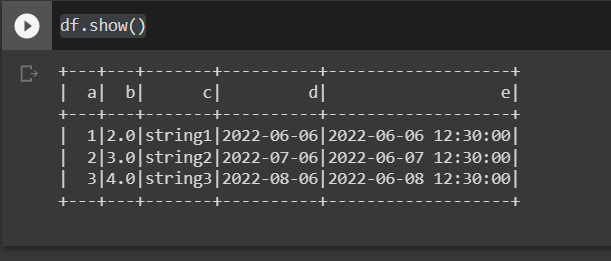
])

df=spark.createDataFrame(rdd, schema=['a','b','c','d','e'])

df

step4—

df.show()



**8. Explain what is RDD & why it is needed ?**

**RDD Resilient Distributed Dataset**

It is basis building block of Spark

The RDD (Resilient Distributed Dataset) is the **Spark's core abstraction.**

It is a collection of elements, partitioned across the nodes of the cluster so that we can execute various **parallel** operations on it.

**There are two ways to create RDDs:**

1. Parallelizing an existing data in the driver program
2. Referencing a dataset in an external storage system, such as a shared filesystem, HDFS, HBase, or any data source offering a Hadoop InputFormat.

Data sharing in MapReduce take a lot of time because of replication, serialization, and disk IO. Hadoop applications take over **90 percent** of the time in read-write operations. So, researchers came up with this **RDD concept** that uses **in-memory** processing computation. Using RDDs increased the data sharing in memory by **10 to 100 times faster than network and disk.**

**9. Write a Py-Spark program to make the column in Upper case & explain with screenshots**

**Step1—**

from pyspark.sql import Column

from pyspark.sql.functions import upper

type(df.c) == type (upper(df.c)) == type(df.c.isNull())

step2—

df.withColumn('upper\_c', upper(df.c)).show()

