**DXC\_Assignment\_6-june-2022**

**Shaik\_Mohammed\_Raihan**

**Case-1:-** **Explain what is in-Memory computation in details?**

* In-memory computation (or in-memory computing) is the technique of running computer calculations entirely in computer memory (e.g., in RAM). This term typically implies large-scale, complex calculations which require specialized systems software to run the calculations on computers working together in a cluster. As a cluster, the computers pool together their RAM so the calculation is essentially run across computers and leverages the collective RAM space of all the computers together.
* In-memory computation works by eliminating all slow data accesses and relying exclusively on data stored in RAM. Overall computation performance is greatly improved by removing the latency commonly seen when accessing hard disk drives or SSDs. Software running on one or more computers manages the computation as well as the data in memory, and in the case of multiple computers, the software divides the computation into smaller tasks which are distributed out to each computer to run in parallel. In-memory computation is often done in the technology known as in-memory data grids (IMDG). One such example is Hazelcast IMDG, which lets users run complex computations on large data sets across a cluster of hardware servers while maintaining extreme speed.

**Case-2:- Explain advantages of Spark framework ?**

* **Speed:-**Engineered from the bottom-up for performance, Spark can be 100x faster than Hadoop for large scale data processing by exploiting in memory computing and other optimizations. Spark is also fast when data is stored on disk, and currently holds the world record for large-scale on-disk sorting.
* **Ease of Use:-**Spark has easy-to-use APIs for operating on large datasets. This includes a collection of over 100 operators for transforming data and familiar data frame APIs for manipulating semi-structured data.
* **A Unified Engine:-**Spark comes packaged with higher-level libraries, including support for SQL queries, streaming data, machine learning and graph processing. These standard libraries increase developer productivity and can be seamlessly combined to create complex workflows**.**

**Case-3:- Explain components of Spark with block diagram ?**

Components of spark:-

Shark (SQL)

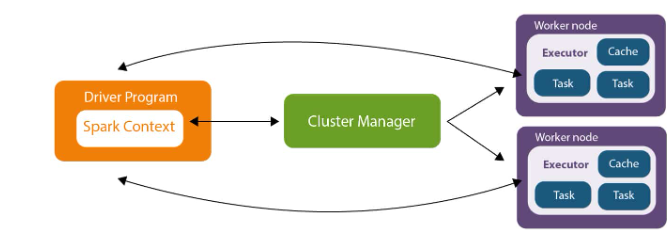
Spark Streaming (Streaming)

MLLib (Machine Learning)

GraphX (Graph Computation)

SparkR (R on Spark)

BlindDB (Approximate SQL)



**Case-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
* Its best suited for performing real time analytics and development and deployment real time applications
* In memory computing imperative:

Avoid movement of detailed data

Calculate first then move the results

**Case-5:-** **Explain major difference between Hadoop & Spark ?**

Hadoop:-

* Hadoop is an open source framework which uses a MapReduce algorithm
* Hadoop’s MapReduce model reads and writes from a disk, thus slow down the processing speed
* Hadoop is designed to handle batch processing efficiently
* Hadoop is a high latency computing framework, which does not have an interactive mode
* With Hadoop MapReduce, a developer can only process data in batch mode only

Spark:-

* Spark is lightning fast cluster computing technology, which extends the MapReduce model to efficiently use with more type of computations.
* Spark reduces the number of read/write cycles to disk and store intermediate data in-memory, hence faster-processing speed.
* Spark is designed to handle real-time data efficiently.
* Spark is a low latency computing and can process data interactively.
* Spark can process real-time data, from real time events like twitter, facebook

**Case-6:- Explain features of Spark?**

* Lighting fast processing speed
* Ease of use
* It offers support for sophisticated analytics
* Real time stream processing
* It is flexible
* Active and expanding community

**Case-7:- Write a Py-Spark program to create Dataframe from RDD & explain with screenshots**

**& steps ?**

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()

+---+---+-------+----------+-------------------+

| a| b| c| d| e|

+---+---+-------+----------+-------------------+

| 1|2.0|string1|2022-06-06|2022-06-06 12:30:00|

| 2|3.0|string2|2022-07-06|2022-06-07 12:30:00|

| 3|4.0|string3|2022-08-06|2022-06-08 12:30:00|

+---+---+-------+----------+-------------------+

**Case-8:-** **Explain what is RDD & why it is needed ?**

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

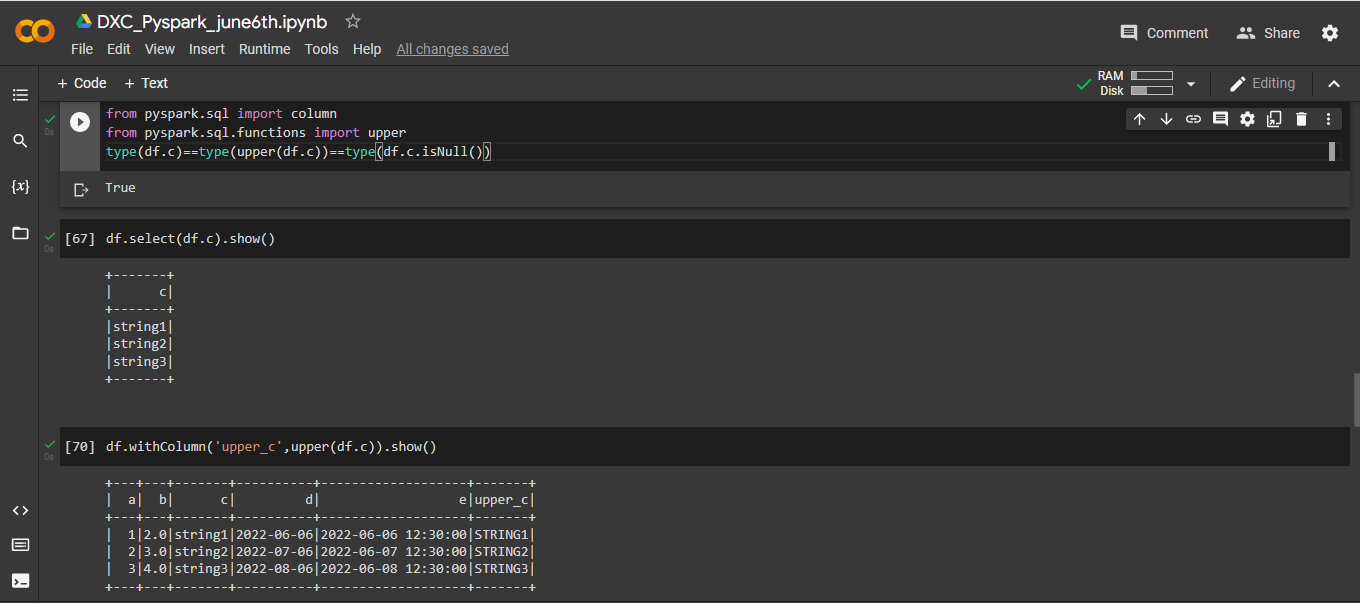
Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

There are two ways to create RDDs − parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations. Let us first discuss how MapReduce operations take place and why they are not so efficient.

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

**& steps ?**

****