**Documentation of Task on 18th may 2021**

**Apache Spark - >**

* Spark is a cluster computing designed for fast computing, actually a memory computing framework which is used mainly to perform batch or stream processing of data. It is built on Hadoop.
* Spark was introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process.
* As against a common belief, Spark is not a modified version of Hadoop and is not, really, dependent on Hadoop because it has its own cluster management. Hadoop is just one of the ways to implement Spark.
* Spark uses Hadoop mainly for storage processing.
* The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.
* It can run on top of the Apache Mesos cluster manager, Hadoop YARN, Amazon EC2, or without an independent resource manager (“standalone mode”).
* Cluster computing here means – that spark distributes the data across the cluster and process the data in parallel.
* Spark uses master/slave architecture i.e., one central coordinator and many distributed workers. Here, the central coordinator is called the driver.

**Internals of How Apache Spark Master/Slave Architecture works ->**

* The driver runs in its own Java process. These drivers communicate with a potentially large number of distributed workers called executors. Each executor is a separate java process. A Spark Application is a combination of driver and its own executors. With the help of cluster manager, a Spark Application is launched on a set of machines. Standalone Cluster Manager is the default built in cluster manager of Spark.

**3 Ways of Spark Deployment - >**

* Standalone − Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.
* Hadoop Yarn − Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.
* Spark in MapReduce (SIMR) − Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

**Spark Components ->**

* Apache Spark Core -> Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.
* Spark SQL -> is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.
* Spark Streaming -> Streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.
* MLlib -> Spark contains common machine learning Algorithms known as SparkMLlib.
* GraphX - > is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs.

**Spark run-time enviroment components ->**

* Apache Spark Driver – The main() method of the program runs in the driver. The driver is the process that runs the user code that creates RDDs, and performs transformation and action, and also creates SparkContext. When the Spark Shell is launched, this signifies that we have created a driver program. On the termination of the driver, the application is finished.  
  The driver program splits the Spark application into the task and schedules them to run on the executor. The task scheduler resides in the driver and distributes task among workers. The two main key roles of drivers are: Converting user program into the task and Scheduling task on the executor. In Spark Program, the DAG (directed acyclic graph) of operations create implicitly. And when the driver runs, it converts that Spark DAG into a physical execution plan.
* Spark Cluster Manager - > Spark relies on cluster manager to launch executors and in some cases, even the drivers launch through it. It is a pluggable component in Spark. On the cluster manager, jobs and action within a spark application scheduled by Spark Scheduler in a FIFO fashion. Alternatively, the scheduling can also be done in Round Robin fashion. The resources used by a Spark application can dynamically adjust based on the workload. Thus, the application can free unused resources and request them again when there is a demand.
* Apache Spark Executers - > The individual task in the given Spark job runs in the Spark executors. Executors launch once in the beginning of Spark Application and then they run for the entire lifetime of an application. Two main task - 1) Runs the task that makes up the application and returns the result to the driver 2) Provide in-memory storage for RDDs that are cached by the user.

**RDD In Spark ->**

Resilient Distributed Datasets 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.

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.

* Resilient – Fault tolerant with DAG(Directed acyclic graph) it can compute missing or damaged partitions due to node failures.
* Distributed – Present on diferent partitions
* Datasets – Data which you’re working on can be structured or unstructured.

Need of Spark RDD->

* To keep data in memory for iterative algorithms and interative data mining tools.
* Because the generality of DSM(Distributed shared memory) cannot efficiently implement the abstraction and fault tolerant manner on commodity clusters. In distributed computing system data is stored in intermediate stable distributed store such as HDFS or Amazon S3. This makes the computation of job slower since it involves many IO operations, replications, and serializations in the process.

**RDD Feature - >**

* Lazy Evaluations – Spark call the datasets as an object only when they’re needed.
* Fault tolerance - Spark RDDs have a provision of in-memory computation. It stores intermediate results in distributed memory(RAM) instead of stable storage(disk).
* Partitioning - is the fundamental unit of parallelism in Spark RDD. Each partition is one logical division of data which is mutable. One can create a partition through some transformations on existing partitions.

**RDD Partitioning In Spark ->**

* Resilient Distributed Datasets are collection of various data items that are so huge in size, that they cannot fit into a single node and have to be partitioned across various nodes. Spark automatically partitions RDDs and distributes the partitions across different nodes. A partition in spark is an atomic chunk of data (logical division of data) stored on a node in the cluster. Partitions are basic units of parallelism in Apache Spark. RDDs in Apache Spark are collection of partitions.
* Types of Partitioning in Apache Spark -> 1) Hash 2 )Range

**Operations in Spark RDD ->**

* Transformation - Spark RDD Transformations are functions that take an RDD as the input and produce one or many RDDs as the output. They do not change the input RDD (since RDDs are immutable and hence one cannot change it), but always produce one or more new RDDs by applying the computations they represent e.g. Map(), filter(), reduceByKey() etc. They are lazy Operation.
* Actions - are the operations which are applied on the Rdd to instruct apache spark to apply computations and pass the result back to the driver, Now the moment Action is envoked all the computations start to execute which are present in the DAG and gives us the results.