**-> INTRODUCTION TO APACHE SPARK**

* Spark is an in-memory computing software which stores the data and allows it processing in parallel
* It is similar to map reduce(hadoop) but the main difference is that hadoop does not follow in-memory computing
* In-memory computing means to store data in RAMs of nodes in a cluster and processing it in parallel
* Where as hadoop stores the data to disk, reads from the disk , perform operations and stores the result back to the disk
* So it is very fast compared to hadoop
* It is 100 times faster than hadoop as because of in-memory and is 10 times faster in accessing the disk compared to hadoop
* RDD - data structure of spark and is also called schema-less data structure as it can handle both structured(like rdbms data) and unstructured(like key-value) data
* RDD is an immutable distributed collection of objects where objects can be anything like strings , lines, rows
* RDD is divided into logical partitions which can operated on by various nodes at a single time in parallel, and the data is distributed by spark
* RDD are highly resilient as the same chunks of data is replicated on various nodes
* RDDs have lazy evaluation that is transformations are not performed until actions are applied
* Localhost port of spark is 4040
* In lazy evaluation data is not loaded until its necessary
* Transformation produce new rdds
* Actions produce results
* Workloads that spark cater
  + Batch:
  + Interactive: we go to the shell and execute commands see the result and depending upon the result we execute next command
  + Streaming: program is continuously running and works on the data as it comes

**-> SPARK ARCHITECTURE**

* RDD can be cached or persisted
* If a RDD is lost it will be automatically recomputed by executing logged transformations
* Spark context is a gateway to all spark functionality
* The driver program(program written by us) along with spark context takes care of executing job across the clusters
* A job is split into tasks and these tasks are distributed over the worker nodes

**-> SPARK RDD**

* Access provided by rdd is read only
* Transformations are applied to access, modify , filter the data present in an rdd
* 2 types of transformations
  + Narrow :
  + Wide :
* .first() will return the schema of an rdd
* .take() action used to collect certain number of data points
* .union() transformation to union data of 2 rdds
* Spark considers the schema as the data to be processed so we remove the schema
* partitions.size - gives the number of partitions in which data is distributed
* .count() action used to count the number of rows

**-> SPARK DATA FRAME**

* They are immutable
* They will not throw any output on the screen unless and until an action command is executed
* .creataDataFrame(use data and schema as parameters)
* .printSchema() to print the schema of data frame

**-> SPARK SQL**

* hive is slow because it converts the sql query into the MapReduce form first and then performs the operations
* Where as spark sql uses in memory computation
* SQL cant work with unstructured data
* SQL works only with structured and semi structured data
* Interpreter and optimizer - converts the data frames into rdd and perform the transformations and actions
* Dataset is different from data frame as it provides encoding mechanism
* Performance wise dataset is better than dataframe
* .toDF() is used to convert a sequence to data frame
* Data frame is different from rdd as data frame stores the data in form of columnar storage whereas rdd just stores the data
* Sparl sql operations cant be performed on data frames but only on temporary views

**-> SPARK STREAMING**

* in filter() transformation the output will be smaller as compared to input
* We will be performing windowing operation depending upon the window size and we cant do for partial time
* Accumulators : used to accumulate different knowledge across the nodes(executors), used because its updated value is relayed back to the driver program, also it is recommended to be used with actions, as until transformations are not applied and executed their values wont be updated

**-> SPARK MLlib**

* If we want to write spark programs in python then we use pyspark, just execute “**pyspark**” and it will connect to pyspark
* Sbt package downloads and install all the dependencies required for the program

**-> SPARK GRAPHX**

* In vertex labeled graph vertex set includes the additional info, but the edge set includes on the vertex id
* In cyclic graph there should be atleast 1 cycle
* Vertices with only in-arrows are called sinks
* Vertices with only out-arrows are called sources
* If we are not using the spark-shell then we will be requiring the spark-context
* Graphx maintains vertices and edges in optimized data structures and provide extra functionality to them
* VertexRDD uses hash map data structure which is reusable, try to preserve index to dervive or create new vertexrdd
* Graphx uses edge cut approach to achieve partitioning strategy
* Edges are assigned to single machine whereas vertices are spanned across various machines
* Partitioning strategy is the one which decides how to assign different edges to different machines or diff partitions
* We try to put the identical edges to a single partition so that diff operations like join can be performed on them
* Since Real world graphs have more edges than vertices thus we move vertice attributes to edges
* There is a routing table which broadcasts all the vertices to the partitions so that join operations can be performed easily
* Property operators when applied create a new graph thus original graph can be reused
* Reverse structural operator does not affect edge attribute or vertex properties or change in number of edges and avoid data movement and duplication
* Subgraph is used to restrict the graph to required vertices and edges and eliminate the rest of the components
* Mask operator is used to compare 2 graphs

**-> HADOOP VS SPARK**

* Hadoop is used to store large amounts of data in cluster of nodes which can easily scale depending upon the need and also provides compute capabilities
* Spark is fast because it uses in memory processing and also uses disk for the data that does not fit into memory
* Spark in memory processing makes able to process real time data
* Yarn is a resource manager that can be used with a number of frameworks
* Hive is data warehousing framework which allows storing writing, reading and processing large amounts of data
* Spark imposes higher costs in hardware set up as machines require large and efficient RAMs
* But overall it reduces the cost per computation by reducing the number of machines which are required for the spark cluster
* Hadoop works on batch processing so when we submit query to YARN it collects the data performs the operations and returns the result and writes the result back to the cluster
* In hadoop if any of the slave machine goes down, master machine re schedules the entire job among the cluster of nodes but this indeed increase the time of completing the job. Also this creates problems for the app which are single fault failure app
* Hadoop also ensures fault tolerance by replication of data
* Rdd in spark provides fault tolerance
* RDD can persist a dataset in memory across operations which make them much faster
* Hadoop is best suited for analyzing archived data
* Hadoop and spark do not compute with each other rather they compliment each other

**-> KAFKA SPARK STREAMING**

* Messaging systems reduce the complexity of data pipelines and make communication between systems more easy and manageable
* Messaging system provides u with a common platform which is independent of a programming language and a paradigm
* Messaging system decouples the front end and the backend server and also allows asynchronus communication , so sender does not have to wait for the receiver to receive the messages
* Thus when receivers network is not working properly the messages wont be lost
* Producer sends the messages to Kafka, it stores them and then the consumers can subscribe to Kafka and consume the messages from there
* in Queuing systems pool of users connect to the queue to get access to the message record but only one of them gets access to the record
* In publish subscribe system, message is broadcasted to all the users so that everyone has access to the message
* Topic is a feed name to which messages of particular kind are stored and is multiple consumer subscribable.
* Topic is divided into partitions such that each partition is stored in different machines under the same topic so that diff users can access the data parallely
* Consumer can belong to a consumer group and a consumer group can access the message from the cluster once and inside the consumer group only 1 of the consumer will access the data once
* Zookeeper stored the metadata related to the kafka cluster like broker info , topics details etc, it is the one who is managing the entire kafka cluster
* The id of the replica partition is the same as the id of the broker that hosts it
* The replica which is most insync with the leader replica will become the next leader
* Spark core engine is the one which handles memory scheduling fault recoveries monitoring jobs
* Spark core engine is the heart of spark
* War file = web archive file
* Acks config controls the criteria under which requests are considered complete, acknowledgement

**-> PySpark**