SPARK

How Spark works

* Driver-Executor architecture
* Data is split and distributed over the worker nodes
* A worker node can contain multiple executors

Lazy Evaluation

* Until an action is run, none of the transactions take place
* This prevents the data from being saved in the Hard Disk again and again which will result in slower processing
* Spark is much faster than Hive because of Lazy Evaluation
* The data in Hive is saved in the Hard Disk every time a transformation runs
* For every action, one job is created

Transformations and actions

* Narrow transformation
  + Separate transformations for every node
  + There is no exchange of data between the nodes
* Wide Transformation
  + There is **shuffling** of data from one node to another
  + The nodes exchange data between each other
  + This results in better processing
  + Shuffling is a costly process

Jobs, Stages, Tasks

* as soon as action operations like collect(), count(), etc., is triggered, the driver program, which is responsible for launching the spark application as well as considered the entry point of any spark application, converts this spark application into a single job
* A job is defined as a series of stages combined.
* Whenever there is a shuffling of data over the network, Spark divides the job into multiple stages. Therefore, a stage is created when the shuffling of data takes place.
* These stages can be either processed parallelly or sequentially depending upon the dependencies of these stages between each other. If there are two stages, Stage 0 and Stage 1, and if they are not sequentially dependent, they will be executed parallelly.
* The sequential processing of RDDs in a single stage is called pipelining.
* Use of reduceByKey() function shuffles the data in order to group the same keys. Since shuffling of data is taking place only once, our job will be divided into two stages as shown in the figure below.
  + Two types of staging in Spark
    - ShuffleMapStage
      * As the name suggests, it is a type of stage in the spark that produces data for shuffle operation.
      * The output of this stage acts as an input for the other following stages.
    - ResultStage
      * The final stage in a Job executes an action operation by running a function (example, the action operation can be collect, write, show, count) on an RDD.

In-Memory processing

* Use of node cache to store the data for quick access and transformations
* This avoids the use of HDD to access the data repeatedly
* This makes the entire process much faster

Spilling

* Data overflows from the cache as the cache can’t hold a lot of data being small in size itself

Catalyst & Tungsten

* Parser -> analyzer -> optimizer -> planner -> query execution