Assignment 12.2

1. How are worker, executor and task related to each other?

-Task: It is a unit of work to execute.

Unlike MapReduce it does not have map and reduce tasks.

Each task either partitions its output for "Shuffle" or sends it back to Driver.

-Executor: Executor runs tasks.

It runs in the Child JVM (= 1 UNIX process) allocated by worker.

Executes one or more tasks using Threads in ThreadPool.

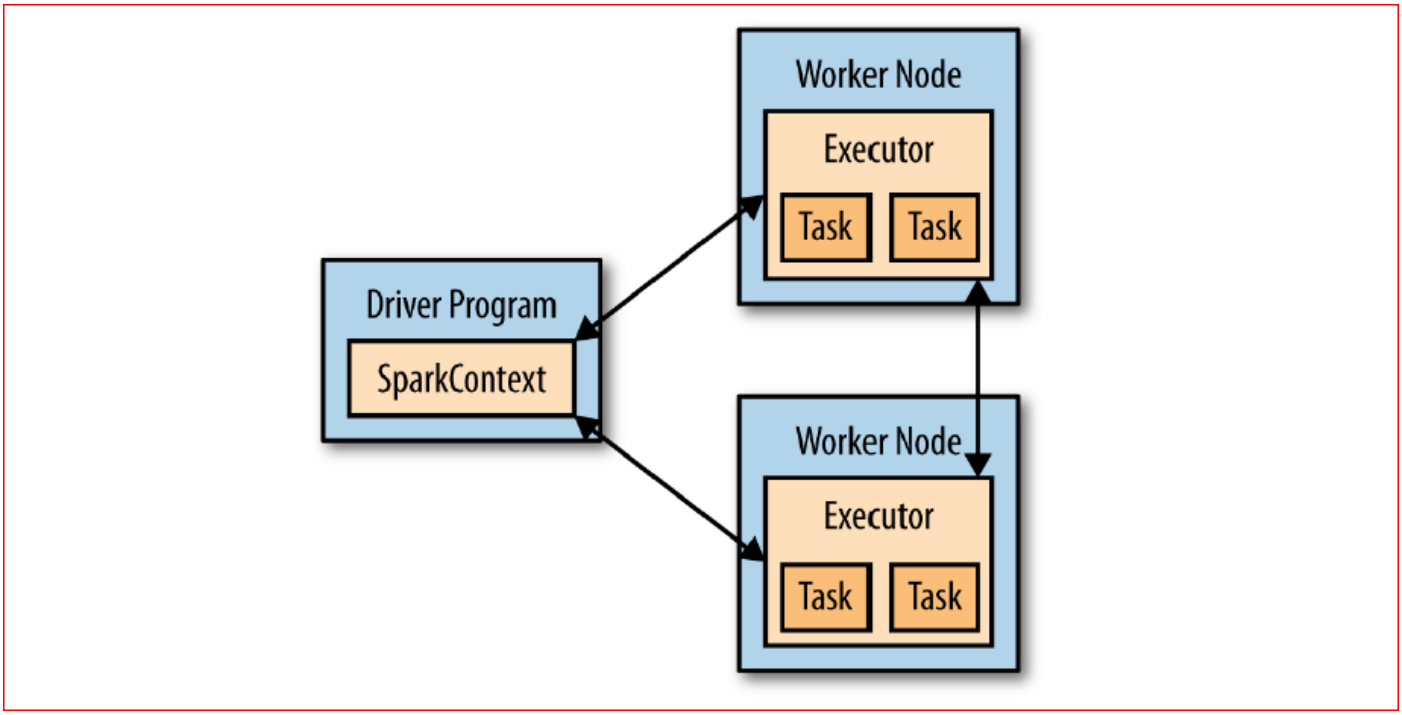
Mainly responsible for computations.

-Worker: is a node/machine that runs Executor.

One JVM runs per worker which is utilised by Executor.

Each Worker can spawn one or more Executor which performs actual computations.

Please see below figure for detailed coordination of the components of the spark.



2. What are the key features of Spark?

- *Rich API* Support for various applications.

- *Support for Resilient Distributed Datasets (RDD)* - API that defines resilient distributed datasets(RDDs), which are Spark's main programming abstraction. RDDs represent a collection of items distributed across many compute nodes that can be manipulated in parallel. Spark Core provides many APIs for building and manipulating these collections.

-*DAG based execution* - Directed Acyclic Graph based execution where edges are data flow and nodes are computational unit.

-*Data caching (In-Memory Processing)* - intermediate results/temp data is stored in memory instead storing it in disk. this mechanism saves so extensive disk read/writes and in memory read writes being faster, computations are done at very faster rate.

-*Strong ecosystem tool support* - Spark is designed to be highly accessible, offering simple APIs in Python, Java, Scala and SQL, and rich built-in libraries. It also integrates closely with other Big Data tools. In particular, Spark can run in Hadoop clusters and access any Hadoop data source, including Cassandra.

-*Unified platform* - The Spark project contains multiple closely integrated components. At its core, Spark is a "computational engine" that is responsible for scheduling, distributing, and monitoring applications consisting of many computational tasks across many worker machines or a computing cluster. Integrated Components are SparkSQL, SparkR, Mlib for m/c learning, Stream computing, GraphX computing.

3. What is Spark Driver?

The driver is the process where the main() method of your program runs. It is the process running the user code that creates a SparkContext, creates RDDs, and performs transformations and actions. When you launch a Spark shell, you've created a driver program. Once the driver terminates, the application is finished. When the driver runs, it performs two duties:

-Converting a user program into tasks

-Scheduling tasks on executors

4. What are the benefits of Spark over MapReduce?

-MapReduce is tightly coupled with Map & Reduce Tasks but Spark does not have these tasks instead it has Task which either partitions its output for "Shuffle" or sends it back to Driver. This allows us to perform analysis over data without following conventional approach ie. map->reduce->map->reduce and new approach can be adopted based on requirements ie. map->reduce->reduce->map->map

-MapReduce can have 0 reducer but cannot have 0 mapper that is not the case in Spark.

-MapReduce writes/reads intermediate results of phases on disk which result in so many disk read/writes which costs a time whereas in spark intermediate results are stored in memory(RAM) itself where read/write is considerably fast which saves time by good margin.

-MapReduce development requires deep understanding of the architecture of hadoop and programming skills, however Spark can be easily adopted due to abstraction level.

5. What is Spark Executor?

-Spark executors are worker processes responsible for running the individual tasks in a given Spark job. Executors are launched once at the beginning of a Spark application and typically run for the entire lifetime of an application, though Spark applications can continue if executors fail. Executors have two roles. First, they run the tasks that make up the application and return results to the driver. Second, they provide in-memory storage for RDDs that are cached by user programs.