**ASSIGNMENT 12.2**

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

* Worker/worker node: A worker is a Spark daemon which can spawn and run multiple executor daemons. Each worker is a JVM.
  + Executor: An executor is a daemon (a JVM) which runs under a worker. It runs tasks and keeps data in memory or disk storage
  + Task: A unit of work that will be sent to one executor. Tasks are executed in executors.

The driver program (which is the main program run by the application) creates a SparkContext. The SparkContext connects to a cluster manager. The SparkContext then acquires executors from the worker via the cluster manager. After acquiring executors the application code (such as JAR or python code as passed to the SparkConext ) is sent to the executors. Finally, SparkContext sends tasks to the executors to run.



1. **What are the key features of Spark?**

The key features of Spark are:

* Rich API: Spark provides a Rich API using which we can perform a variety of operations on our data
* Resilient Distributed Datasets (RDD): RDDs are Immutable and partitioned collection of records. Since data is partitioned, it allows for parallel processing
* DAG based execution: Spark in the form of its RDDs does not store the actual data. But it stores only the transformations on the data in the form of directed acyclic graphs (DAG)
* Data caching (In-Memory Processing): Spark stores RDDs in memory rather than on disk. This allows for faster processing of data as lot of I/O operations are not involved
* Strong ecosystem tool support: Spark ties well with other tools in the Hadoop ecosystem like Apache Flume and Kafka
* Unified platform: Spark has a single underlying component called Spark core. It exposes this platform in various forms for various ways to analyze data(example: Spark R for statistical analysis, MLIb for machine learning)

1. **What is Spark Driver?**

Spark Driver is the process running the main() function of the application and creating the SparkContext. It has the following functions:

* Creates the SparkContext which is the object responsible for interacting with the cluster manager and the executors
* Connects to a cluster manager to allocate resources across applications
* Acquires executors on cluster nodes to perform tasks
* Sends app code to the executors
* Sends tasks for the executors to run

1. **What are the benefits of Spark over MapReduce?**

* Efficient handling of iterative algorithms: In case of MapReduce, the results between two iterations will have to be stored on disk and a second MapReduce job would have to be written to handle the next iteration. In contrast Spark uses a DAG based execution and also it stores data in-memory. Therefore, handles iterative jobs efficiently
* Not disk intensive: MapReduce stores intermediate data on disks. So in a MapReduce job a lot of I/O operations are performed. In contrast, Spark stores data in memory
* No programming limitations: In MapReduce every program needs to be broken down into map and reduce phases. The developer may also be required to break down the problem into multiple map-reduce iterations because high level operations are not supported. Spark has a rich API which supports high-level operations which makes programs simpler

1. **What is Spark Executor?**

An executor is a daemon (a JVM) which runs under a worker. It runs tasks and keeps data in memory or disk storage. An executor is spawned by a worker.