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

Worker node: Any node that can run application code in the cluster. Workers are responsible for distributed computing and does the computation by executors present in the worker nodes.

Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in main program (called the driver program). Specifically, to run on a cluster, the SparkContext can connect to several types of cluster managers (either Spark’s own standalone cluster manager or Mesos/YARN), which allocate resources across applications. Once connected, Spark acquires executors on worker nodes in the cluster, which are processes that run computations and store data for your application. Next, it sends your application code (defined by JAR or Python files passed to SparkContext) to the executors. Finally, SparkContext sends tasks for the executors to run.

Executors: These are worker nodes' processes in charge of running individual tasks and computations in a given Spark job. They are launched at the beginning of a Spark application by cluster manager when driver program ask for resources to the cluster manager and typically run for the entire lifetime of an application. Once they have run the task they send the results to the driver. They also provide in-memory storage for RDDs that are cached by user programs

Task: A unit of work that will be sent to executor from the driver. Multiple tasks comprise a job that gets spawned in response to a Spark action.

2. What are the key features of Spark?

* Rich API
* Resilient Distributed Datasets (RDD)
* DAG based execution
* Data Caching (In Memory Processing)
* Tool Support
* Unified Platform

3. What is Spark Driver?

The process running the main() function of the application and creating the SparkContext.

It also does the following at granular level:

* Connects to a cluster manager to allocate resources across applications
* Acquires executors on cluster nodes processes run compute tasks, cache data
* Sends app code to the executors
* Sends tasks for the executors to run

4. What are the benefits of Spark over MapReduce?

It addresses the following limitation of MapReduce

Inefficient handling of iterative algorithms

Disk intensive

Programming Model limitations:

* Developing efficient MapReduce applications requires advanced programming skills and deep understanding of the system architecture
* Every problem has to be broken down into Map and Reduce phases

There are no ‘map’ vs ‘reduce’ tasks in Spark.

5. What is Spark Executor?

These are worker nodes' processes in charge of running individual tasks and computations in a given Spark job. They are launched at the beginning of a Spark application by cluster manager when driver program ask for resources to the cluster manager and typically run for the entire lifetime of an application. Once they have run the task they send the results to the driver. They also provide in-memory storage for RDDs that are cached by user programs