1. **Pen down the limitations of MapReduce.**

* It’s based on disk based computing.

●Suitable for single pass computations -not iterative computations.

●Needs a sequence of MR jobs to run iterative tasks.

●Needs integration with several other frameworks/tools to solve bigdata usecases.

○Apache Storm for stream data processing

○Apache Mahout for machine learning

1. **What is RDD? Explain few features of RDD?**

It is the primary abstraction in Spark and is the core of Apache Spark.One could compare RDDs to collections in Scala, i.e. a RDD is computed on many JVMs while a Scala collection lives on a single JVM.

○Resilient, i.e. fault-tolerant with the help of RDD lineage graph and so able to recompute missing or damaged partitions due to node failures

○Distributedwith data residing on multiple nodes in a cluster.

○Datasetis a collection of partitioned data with primitive values or values of values, e.g. tuples or other objects

●Features of RDD

In-Memory, i.e. data inside RDD is stored in memory as much (size) and long (time) as possible.

•Immutableor Read-Only, i.e. it does not change once created and can only be transformed using transformations to new RDDs.

•Lazy evaluated, i.e. the data inside RDD is not available or transformed until an action is executed that triggers the execution.

•Cacheable, i.e. you can hold all the data in a persistent "storage" like memory (default and the most preferred) or disk (the least preferred due to access speed).

•IParallel, i.e. process data in parallel.

•Typed—RDD records have types, e.g. Long in RDD[Long] or (Int, String) in RDD[(Int, String)].

•Partitioned—records are partitioned (split into logical partitions) and distributed across nodes in a cluster.

•Location-Stickiness—RDD can define placement preferences to compute partitions (as close to the records as possible).

1. List down few Spark RDD operations and explain each of them.

count -returns the number of elements in the RDD

●countApprox(long timeout, double confidence) -Approximate version of count() that returns a potentially incomplete result within a timeout, even if not all tasks have finished.

●countByValue(scala.math.Ordering<T> ord) -Return the count of each unique

●distinct() -Return a new RDD containing the distinct elements in this RDD.

●filter(function) -Return a new RDD containing only the elements that satisfy a predicate.

first() -Return the first element in this RDD.

●flatMap(function) -Return a new RDD by first applying a function to all elements of this RDD, and then flattening the results.

●isEmpty()

●top(int num, scala.math.Ordering<T> ord) -Returns the top k (largest) elements from this RDD as defined by the specified implicit Ordering[T] and maintains the ordering.