Assignment 13.1

1.What is RDD?

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

There are two ways to create RDDs − **parallelizing** an existing collection in your driver program, or **referencing a dataset** in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations.

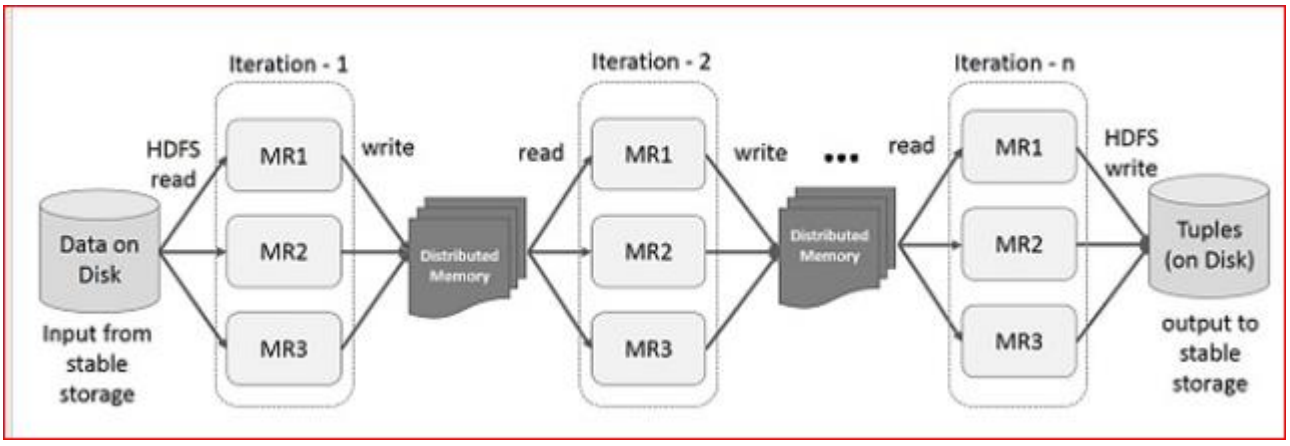
2. Define Partitions.

Large Files are stored in Blocks in distributed file systems, these blocks are associated with a partition in RDD. These partitions are nothing but logical entity similar to InputSplit in HDFS which store date of RDD and resides on worker nodes which are then computed parallelly across the cluster. In layman language Partitions are the logically separated piece of file which points to actual location of the file in distributed or local system which enables parallel processing in spark.

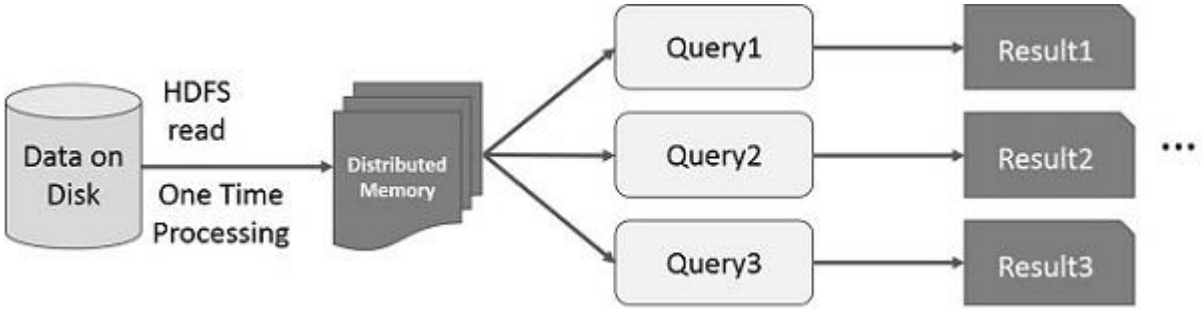
3. What operations does RDD support?

RDD supports Iterative and Interactive Operations.

Iterative Operations on Spark RDD: It will store intermediate results in a distributed memory instead of Stable storage (Disk) and make the system faster. **Note** − If the Distributed memory (RAM) is not sufficient to store intermediate results (State of the JOB), then it will store those results on the disk.



Interactive Operations on Spark RDD: If different queries are run on the same set of data repeatedly, this particular data can be kept in memory for better execution times.



By default, each transformed RDD may be recomputed each time you run an action on it. However, you may also **persist** an RDD in memory, in which case Spark will keep the elements around on the cluster for much faster access, the next time you query it. There is also support for persisting RDDs on disk, or replicated across multiple nodes.

4. What do you understand by Transformations in Spark?

RDD transformations returns pointer to new RDD and allows you to create dependencies between RDDs. Each RDD in dependency chain (String of Dependencies) has a function for calculating its data and has a pointer (dependency) to its parent RDD.

Spark is lazy, so nothing will be executed unless you call some actions that will trigger job creation and execution. Therefore, RDD transformation is not a set of data but is a step in a program (might be the only step) telling Spark how to get data and what to do with it.

In short, Transformations either create or result in RDD. It creates new RDD from previous one. Examples are as below:

myRDD = sc.textFile("FilePath")

mappedRDD = myRDD.map(func)

5. Define Actions.

compute a result based on an RDD, and either return it to the driver program or save it to an external storage system (e.g., HDFS). Examples are as below:

myRDD.count()

myRDD.take(2)