* **Pen down the limitations of MapReduce.**
* It is based on disk based computing. Meaning it stores result or intermediate data on disk. Hence the IO operations take more time to process MR jobs
* It is not suitable for iterative task. It needs sequence of MR jobs to run iterative task.
* It needs integration with several other frameworks/tools to solve Big data usecases

Like Apache storm to solve data processing and Apache Mahout for machine learning .

* Programming in MR is cumbersome. For Pig as well one needs to learn the syntax.
* MR doesn’t have Interactive mode.
* MR doesn’t support real time processing on its own.
* **What is RDD? Explain few features of RDD?**

RDD stands for “Resilient Distributed Dataset”. It is the fundamental data structure of Apache Spark. RDD in Apache Spark is an immutable collection of objects which computes on the different node of the cluster.

* **Resilient**, i.e. fault-tolerant with the help of RDD lineage graph(DAG) and so able to recompute missing or damaged partitions due to node failures.
* **Distributed**, since Data resides on multiple nodes.
* **Dataset** represents records of the data you work with. The user can load the data set externally which can be either JSON file, CSV file, text file or database via JDBC with no specific data structure.

There are three ways to create RDDs in Spark such as – Data in stable storage, other RDDs, and parallelizing already existing collection in driver program.

Spark RDD can also be cached and manually partitioned. Caching is beneficial when we use RDD several times. And manual partitioning is important to correctly balance partitions. Generally, smaller partitions allow distributing RDD data more equally, among more executors.

Programmers can also call a persist method to indicate which RDDs they want to reuse in future operations. Spark keeps persistent RDDs in memory by default, but it can spill them to disk if there is not enough RAM.

RDD in Apache Spark supports two types of operations:

* **Transformation**:

Spark RDD Transformations are functions that take an RDD as the input and produce one or many RDDs as the output.

e.g. Map(), filter(), reduceByKey() etc.

Transformations are lazy operations on an RDD in Apache Spark. It creates one or many new RDDs, which executes when an Action occurs.

* **Actions**:

An Action in Spark returns final result of RDD computations. It triggers execution using lineage graph to load the data into original RDD, carry out all intermediate transformations and return final results to Driver program or write it out to file system.

Actions are RDD operations that produce non-RDD values. They materialize a value in a Spark program. An Action is one of the ways to send result from executors to the driver. First(), take(), reduce(), collect(), the count() is some of the Actions in spark

**Features of RDD:**

* In-memory Computation

Spark RDDs have a provision of in-memory computation. It stores intermediate results in distributed memory(RAM) instead of stable storage(disk).

* Lazy Evaluations

All transformations in Apache Spark are lazy, in that they do not compute their results right away. Instead, they just remember the transformations applied to some base data set.

* Fault tolerance:

They rebuild lost data on failure using lineage, each RDD remembers how it was created from other datasets (by transformations like a map, join or groupBy) to recreate itself.

* Immutability:

Data is safe to share across processes. It can also be created or retrieved anytime which makes caching, sharing & replication easy.

* Partitioning:

Partitioning is the fundamental unit of parallelism in Spark RDD. Each partition is one logical division of data which is mutable. One can create a partition through some transformations on existing partitions.

* Persistence:

Users can state which RDDs they will reuse and choose a storage strategy for them (e.g., in-memory storage or on Disk).

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

RDD in Apache Spark supports two types of operations:

* **Transformation**:

Spark RDD Transformations are functions that take an RDD as the input and produce one or many RDDs as the output.

e.g. Map(), filter(), reduceByKey() etc.

Transformations are lazy operations on an RDD in Apache Spark. It creates one or many new RDDs, which executes when an Action occurs.

* + **Map**:

Map will take each row as input and return an RDD for the row.

* **Flat map:**

flatMap will take an iterable data as input and returns the RDD as the contents of the iterator.

* **Filter:**

**filter** returns an RDD which meets the filter condition

* **ReduceByKey:**

reduceByKey takes a pair of key and value pairs and combines all the values for each unique key.

* **Actions:**

An Action in Spark returns final result of RDD computations. It triggers execution using lineage graph to load the data into original RDD, carry out all intermediate transformations and return final results to Driver program or write it out to file system.

Actions are RDD operations that produce non-RDD values. They materialize a value in a Spark program. An Action is one of the ways to send result from executors to the driver. First(), take(), reduce(), collect(), the count() is some of the Actions in spark.

* **Collect:**

It is used to return all the elements in the RDD.

* **Count:**

It is used to return the number of elements in the RDD.

* **CountByValue:**

It is used to count the number of occurrences of the elements in the RDD. (occurrence of key value)

* **Reduce:**

It performs operation on all rows of RDD together.

E.g. Val test= map\_test.map(line => (line(9).toInt))

Test.reduce((a,b)=>a+b) --gives addition of all rows

* **Take:**

It will display the number of records we explicitly specify.

e.g. Test.reduceByKey(\_+\_).collect -- will display all rows.

Test.reduceByKey(\_+\_).take(2) -- will display only two rows.

For detailed info: https://acadgild.com/blog/spark-rdds-scala/