**Assignment 16.2**

1. **Pen Down the limitations of Map Reduce**

Sol: Some of the limitations are:

* Since Map Reduce is suitable only for batch processing jobs, implementing interactive jobs and models becomes impossible.
* When map phase generate too many keys. Then sorting takes for ever
* Implementing iterative map reduce jobs is expensive due to the huge space consumption by each job.
* Problems that cannot be trivially partition-able or recombinable becomes a candid limitation of Map Reduce problem solving. For instance, Traveling Salesman problem.
* When you have OLTP needs. MR is not suitable for a large number of short on-line transactions.
* Due to the fixed cost incurred by each Map Reduce job submitted, application that requires low latency time or random access to a large set of data is in feasible.
* Also, tasks that has a dependency on each other cannot be parallelized, which is not possible through Map Reduce.
* It does not do well for graph, iterative, incremental and many other kinds.
* Map Reduce cannot cache the intermediate data in-memory for a further requirement which diminishes the performance of hadoop.

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

Sol:

RDD stands for **Resilient Distributed Data set**. An RDD is, essentially, the Spark representation of a set of data, spread across multiple machines, with APIs to let you act on it. An RDD could come from any data source, e.g. text files, a database via JDBC, etc.

**Definition:**

**RDDs are fault-tolerant, parallel data structures that let users explicitly persist intermediate results in memory, control their partitioning to optimize data placement, and manipulate them using a rich set of operators.**

**Features of RDD are:**

* **In-memory computation**

The data inside RDD are stored in memory for as long as you want to store. Keeping the data in-memory improves the performance by an order of magnitudes. refer this comprehensive guide to Learn Spark in-memory computation in detail.

* L**azy Evaluation**

The data inside RDDs are not evaluated on the go. The changes or the computation is performed only after an action is triggered. Thus, it limits how much work it has to do. Follow this guide to learn Spark lazy evaluation in great detail.

* **Fault Tolerance**

Upon the failure of worker node, using lineage of operations we can re-compute the lost partition of RDD from the original one. Thus, we can easily recover the lost data. Learn Fault tolerance is Spark in detail.

* **Immutability**

RDDS are immutable in nature meaning once we create an RDD we can not manipulate it. And if we perform any transformation, it creates new RDD. We achieve consistency through immutability.

* **Persistence**

We can store the frequently used RDD in in-memory and we can also retrieve them directly from memory without going to disk, this speedup the execution. We can perform Multiple operations on the same data, this happens by storing the data explicitly in memory by calling persist() or cache() function. Follow this guide for the detailed study of RDD persistence in Spark.

* **Partitioning**

RDD partition the records logically and distributes the data across various nodes in the cluster. The logical divisions are only for processing and internally it has no division. Thus, it provides parallelism.

* **Parallel**

Rdd, process the data parallelly over the cluster.

* **Location-Stickiness**

RDDs are capable of defining placement preference to compute partitions. Placement preference refers to information about the location of RDD. The DAGScheduler places the partitions in such a way that task is close to data as much as possible. Thus speed up computation. Follow this guide to learn What is DAG?

* **Coarse-grained Operation**

We apply coarse-grained transformations to RDD. Coarse-grained meaning the operation applies to the whole dataset not on an individual element in the data set of RDD.

* **Typed**

We can have RDD of various types like: RDD [int], RDD [long], RDD [string].

* **No limitation**

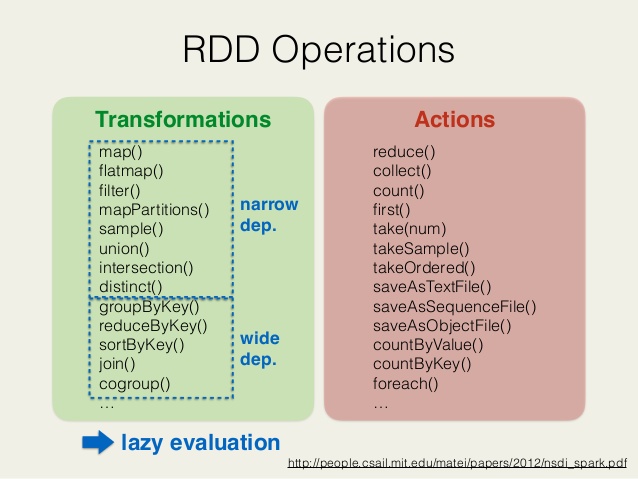
we can have any number of RDD. there is no limit to its number. the limit depends on the size of disk and memory.

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

**Sol:**

RDD in Apache Spark supports two types of operations:

* Transformation
* Actions

****

**1.Transformations**

Spark RDD Transformations are functions that take an RDD as the input and produce one or many RDDs as the output. They do not change the input RDD (since RDDs are immutable and hence one cannot change it), but always produce one or more new RDDs by applying the computations they represent 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. Hence, Transformation creates a new dataset from an existing one.

Certain transformations can be pipelined which is an optimization method, that Spark uses to improve the performance of computations. There are two kinds of transformations: **narrow transformation, wide transformation.**

* **Narrow Transformations**

It is the result of map, filter and such that the data is from a single partition only, i.e. it is self-sufficient. An output RDD has partitions with records that originate from a single partition in the parent RDD. Only a limited subset of partitions used to calculate the result.

Spark groups narrow transformations as a stage known as pipelining.

**Wide Transformations**

It is the result of groupByKey() and reduceByKey() like functions. The data required to compute the records in a single partition may live in many partitions of the parent RDD. Wide transformations are also known as shuffle transformations because they may or may not depend on a shuffle.

2.**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. Lineage graph is dependency graph of all parallel RDDs of RDD.

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.

Using transformations, one can create RDD from the existing one. But when we want to work with the actual dataset, at that point we use Action. When the Action occurs it does not create the new RDD, unlike transformation. Thus, actions are RDD operations that give no RDD values. Action stores its value either to drivers or to the external storage system. It brings laziness of RDD into motion.