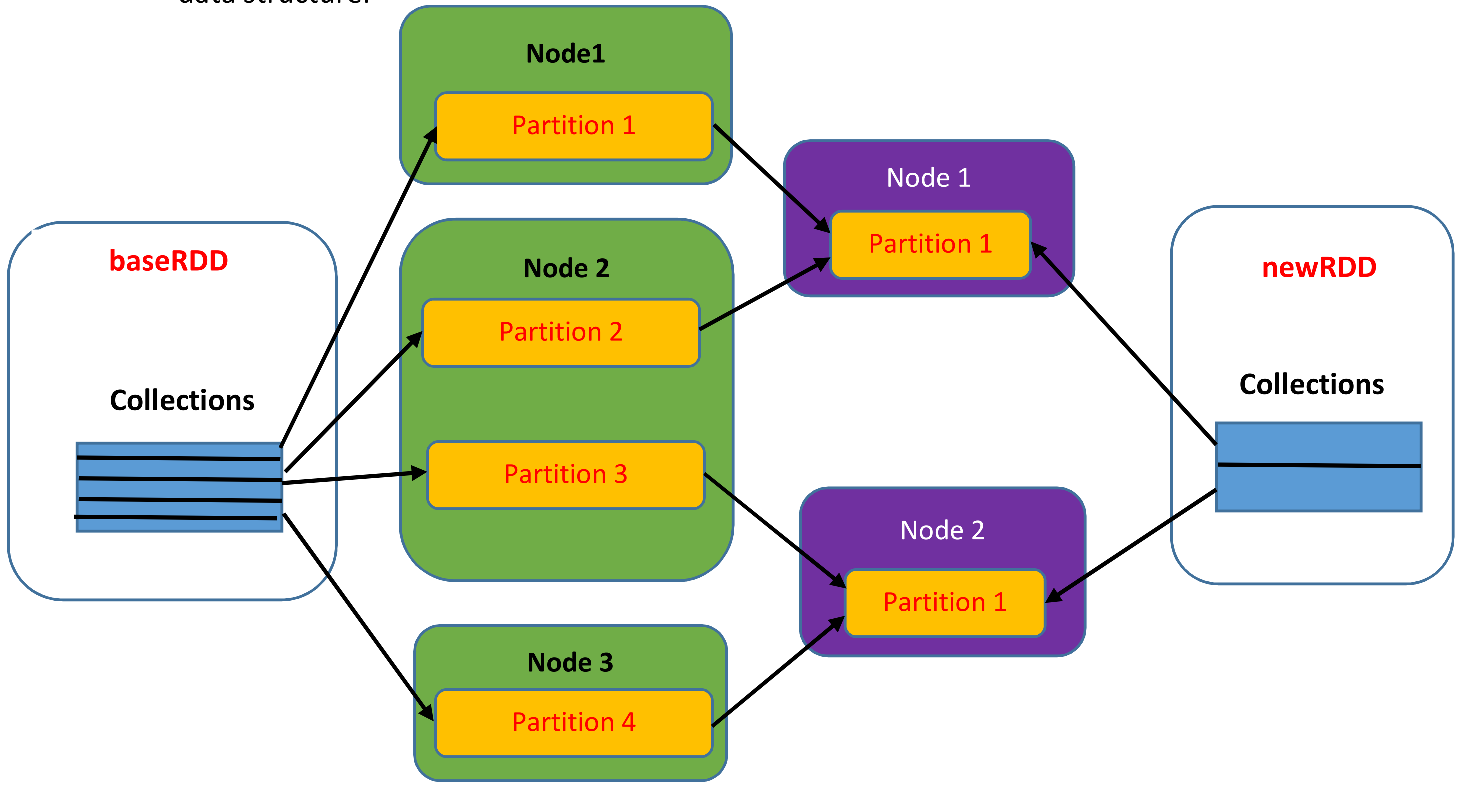
Assignment 16.2

1. Pen down the Limitations of MapReduce.
   - Batch procession is not interactive
   - Designed for a specific problem domain
   - MapReduce programming paradigm is not commonly understood (functional)
   - Lack of trained support professionals
   - API/ Security model are moving targets.
   - We cannot use MapReduce for **Real-time** processing
   - It is not always very easy to implement each and everything as a MR program.
   - MapReduce is not handy when processing requires lot of data to be **shuffled** over the network.
   - No control of the order in which reduce jobs are performed. Only order is that reduce jobs start after map jobs.
2. What is RDD? Explain few features of RDD?

RDD(Resilient Distributed Dataset) is the fundamental data structure of **Apache Spark** which are immutable collection of objects which computes on the different node of the cluster.

Each and every dataset in Spark RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster.

- **Resilient** is 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,CSV,TEXT files or databse via JDBC with no specific data structure.



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 Evaluation**: All transformations in Spark are lazy, in that they do not compute their results right away. Instead, they just remember the transformations applied to the some base dataset.
- **Fault Tolerance**: Spark RDDs are fault tolerant as they track data lineage information to rebuild lost data automatically on failure.
- **Immutability**: Data is safe to share across processes. It can be created or retrieved anytime which makes caching, sharing and replication easy.
- **Partitioning**: Partitioning is the fundamental unit of parallelism in Spark RDD.
- **Persistence**: User can state which RDDs they will reuse and choose a storage strategy for them (e.g. in-memory storage or on Disk)
- **Coarse-grained Operations**: It applies to all elements in datasets through maps or filters or group by operation.

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

Two operations of Spark RDD are :

   a. Transformations
   b. Actions

- **Transformation**: Transformations are kind of operations which will transform your RDD from one form to another. And when you apply this operation on any RDD, you will get a new RDD with transformed data. Operations like **map, filter, flatMap** are transformations.
- **Actions**: This kind of operation will also give you another RDD but this operation will trigger all the lined up transformation on the base RDD(or in the DAG(Directed Acyclic Graph)) and than execute the action operation on the last RDD. Operations like **collect, count, first** are actions.