## Apache Spark RDD

[**RDD**](http://data-flair.training/blogs/rdd-in-apache-spark/) stands **Resilient Distributed Dataset**. RDDs are the fundamental abstraction of Apache Spark. It is an immutable distributed collection of the dataset. Each dataset in RDD is divided into logical partitions. On the different node of the cluster, we can compute These partitions. RDDs are a read-only partitioned collection of record. we can[create RDD in three ways](http://data-flair.training/blogs/how-to-create-rdds-in-apache-spark/):

* **Parallelizing** already existing collection in driver program.
* **Referencing a dataset** in an external storage system (e.g. [HDFS](http://data-flair.training/blogs/apache-hadoop-hdfs-introduction-tutorial/), [Hbase](http://data-flair.training/blogs/hbase-tutorial-beginners-guide/), shared file system).
* Creating RDD **from already existing RDDs**.

**Features of Spark RDD**

There are several advantages of using RDD. Some of them are-

**1)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](http://data-flair.training/blogs/apache-spark-in-memory-computing/) in detail.

**2)Lazy 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](http://data-flair.training/blogs/lazy-evaluation-in-apache-spark-guide/).

**3)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](http://data-flair.training/blogs/apache-spark-streaming-fault-tolerance/).

**4)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.

**5)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](http://data-flair.training/blogs/apache-spark-rdd-persistence-caching/).

**6)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.

**7)Parallel**

Rdd, process the data parallelly over the cluster.

**8)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.

**9)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.

**10)Typed**

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

**11)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.

**Conclusion**

Hence, using RDD we can recover the shortcoming of Hadoop MapReduce and can handle the large volume of data, as a result, it decreases the time complexity of the system. Thus the above-mentioned features of Spark RDD make them useful for fast computations and increase the performance of the system.