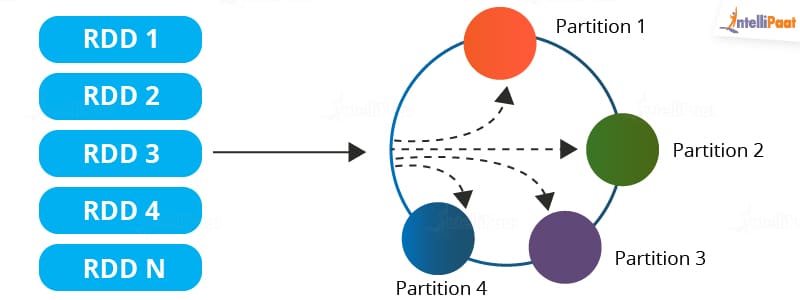
**Resilient Distributed Datasets (RDDs)**

RDDs are the main logical data units in [**Spark**](https://intellipaat.com/blog/what-is-apache-spark/)**.** They are a distributed collection of objects, which are stored in memory or on disks of different machines of a cluster. A single RDD can be divided into multiple logical partitions so that these partitions can be stored and processed on different machines of a cluster.

RDDs are immutable (read-only) in nature. You cannot change an original RDD, but you can create new RDDs by performing coarse-grain operations, like transformations, on an existing RDD.

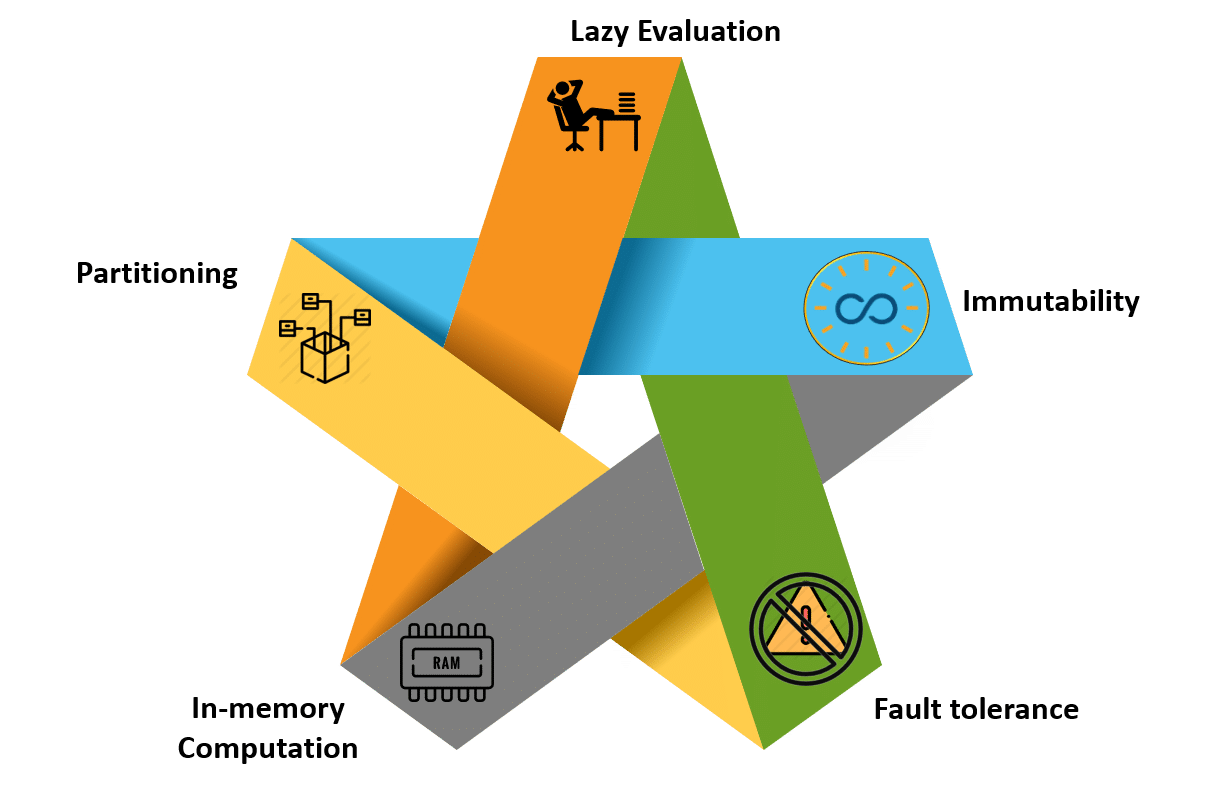


An RDD in Spark can be cached and used again for future transformations, which is a huge benefit for users. RDDs are said to be lazily evaluated, i.e., they delay the evaluation until it is really needed. This saves a lot of time and improves efficiency.

**Features of an RDD in Spark**

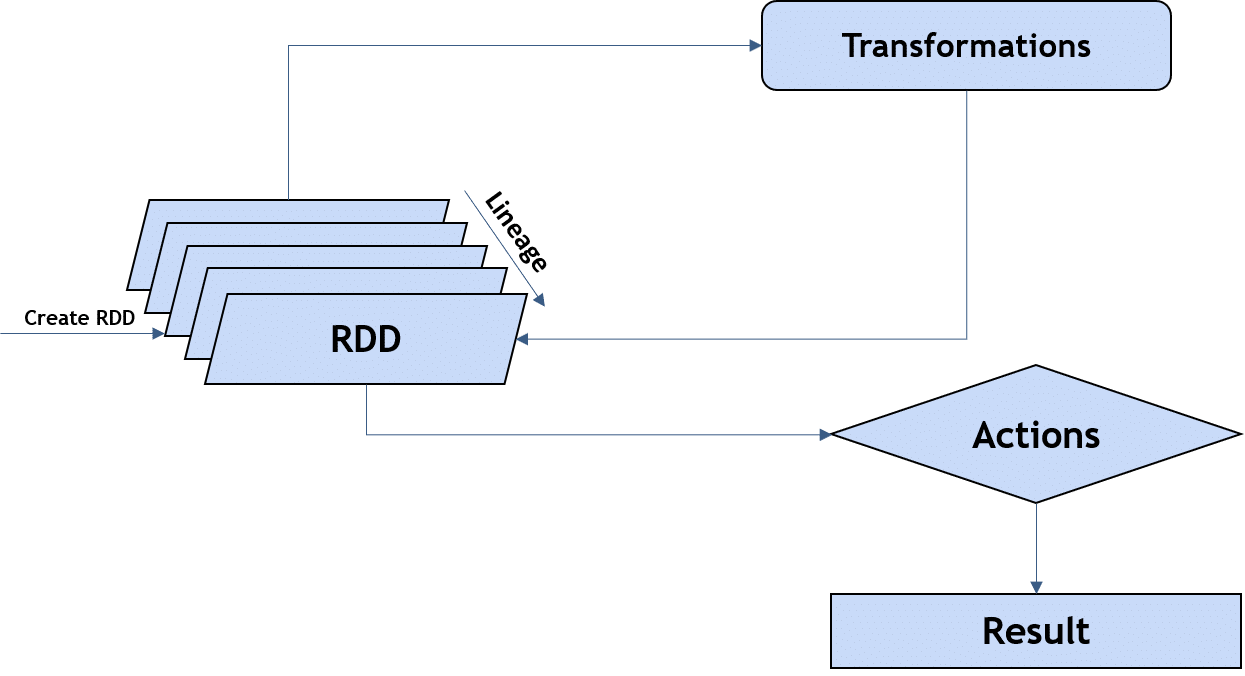
Here are some features of RDD in Spark:

* **Resilience**: RDDs track data lineage information to recover lost data, automatically on failure. It is also called fault tolerance.
* **Distributed**: Data present in an RDD resides on multiple nodes. It is distributed across different nodes of a cluster.
* **Lazy evaluation**: Data does not get loaded in an RDD even if you define it. Transformations are actually computed when you call action, such as count or collect, or save the output to a file system.



**Operations on RDDs**

There are two basic operations that can be done on RDDs. They are **transformations**and **actions**.



### **Transformations**

These are functions that accept the existing RDDs as input and output one or more RDDs. However, the data in the existing RDD in Spark does not change as it is immutable. Some of the transformation operations are provided in the table below:

|  |  |
| --- | --- |
| **Function** | **Description** |
| map() | Returns a new RDD by applying the function on each data element |
| filter() | Returns a new RDD formed by selecting those elements of the source on which the function returns true |
| reduceByKey() | Aggregates the values of a key using a function |
| groupByKey() | Converts a (key, value) pair into a (key, <iterable value>) pair |
| union() | Returns a new RDD that contains all elements and arguments from the source RDD |
| intersection() | Returns a new RDD that contains an intersection of the elements in the datasets |

These transformations are executed when they are invoked or called. Every time transformations are applied, a new RDD is created.

### Actions

Actions in Spark are functions that return the end result of RDD computations. It uses a lineage graph to load data onto the RDD in a particular order. After all of the transformations are done, actions return the final result to the Spark Driver. Actions are operations that provide non-RDD values. Some of the common actions used in Spark are given below:

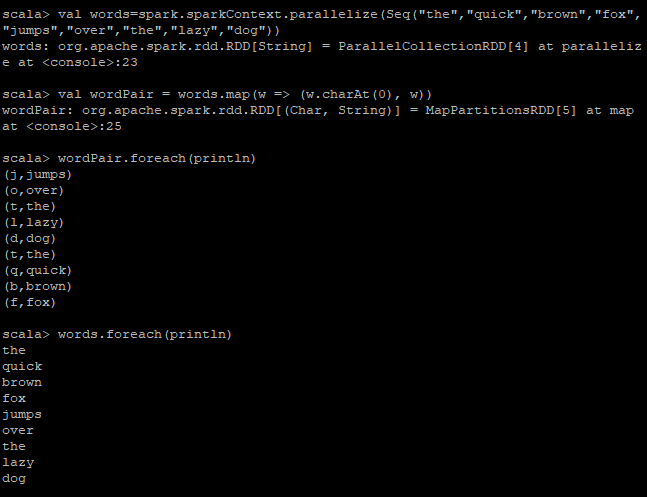
|  |  |
| --- | --- |
| **Function** | **Description** |
| count() | Gets the number of data elements in an RDD |
| collect() | Gets all the data elements in an RDD as an array |
| reduce() | Aggregates data elements into an RDD by taking two arguments and  Returning one |
| take(n) | Fetches the first *n* elements of an RDD |
| foreach(operation) | Executes the **operation** for each data element in an RDD |
| first() | Retrieves the first data element of an RDD |

## Creating an RDD

An RDD can be created in three ways. Let’s discuss them one by one.

### By Loading an External Dataset

You can load an external file onto an RDD. The types of files you can load are csv, txt, JSON, etc. Here is the example of loading a text file onto an RDD:



## **Limitation of RDD**

* There is no input optimization available in RDDs
* One of the biggest limitations of RDDs is that the execution process does not start instantly.
* No changes can be made in RDD once it is created.
* RDD lacks enough storage memory.
* The run-time type safety is absent in RDDs.