



Both caching and persisting are used to save the Spark RDD, Dataframe and Dataset’s. But, the difference is, [RDD cache()](https://sparkbyexamples.com/apache-spark-rdd/spark-rdd-cache-and-persist-example/) method default saves it to memory (**MEMORY\_ONLY**) whereas persist() method is used to store it to user-defined storage level.

When you persist a dataset, each node stores it’s partitioned data in memory and reuses them in other actions on that dataset. And Spark’s persisted data on nodes are fault-tolerant meaning if any partition of a Dataset is lost, it will automatically be recomputed using the original transformations that created it.

What is RDD Persistence and Caching in Spark?

Spark[**RDD**](http://data-flair.training/blogs/rdd-in-apache-spark/)persistence is an optimization technique in which saves the result of RDD evaluation. Using this we save the intermediate result so that we can use it further if required. It reduces the computation overhead.  
We can make persisted RDD through **cache()** and **persist()** methods. When we use the cache() method we can store all the RDD in-memory. We can persist the RDD in memory and use it efficiently across parallel operations.  
The difference between cache() and persist() is that using cache() the default storage level is **MEMORY\_ONLY** while using persist() we can use various storage levels (described below). It is a key tool for an interactive algorithm. Because, when we persist RDD each node stores any partition of it that it computes in memory and makes it reusable for future use. This process speeds up the further computation ten times.  
When the RDD is computed for the first time, it is kept in memory on the node. The cache memory of the [Spark is fault tolerant](http://data-flair.training/blogs/apache-spark-streaming-fault-tolerance/) so whenever any partition of RDD is lost, it can be recovered by [transformation Operation](http://data-flair.training/blogs/rdd-transformations-actions-apis-apache-spark/) that originally created it.

3. Need of Persistence in Apache Spark

In Spark, we can use some RDD’s multiple times. If honestly, we repeat the same process of**RDD** **evaluation** each time it required or brought into action. This task can be time and memory consuming, especially for iterative algorithms that look at data multiple times. To solve the problem of repeated computation the technique of persistence came into the picture.

4. Benefits of RDD Persistence in Spark

There are some advantages of RDD caching and persistence mechanism in spark. It makes the whole system

* Time efficient
* Cost efficient
* Lessen the execution time.

5. Storage levels of Persisted RDDs

Using **persist()** we can use various storage levels to Store Persisted RDDs in Apache Spark. Let’s discuss each RDD storage level one by one-

a. MEMORY\_ONLY

In this storage level, RDD is stored as deserialized Java object in the JVM. If the size of RDD is greater than memory, It will not cache some partition and recompute them next time whenever needed. In this level the space used for storage is very high, the CPU computation time is low, the data is stored in-memory. It does not make use of the disk.

b. MEMORY\_AND\_DISK

In this level, RDD is stored as deserialized Java object in the JVM. When the size of RDD is greater than the size of memory, it stores the excess partition on the disk, and retrieve from disk whenever required. In this level the space used for storage is high, the CPU computation time is medium, it makes use of both in-memory and on disk storage.

c. MEMORY\_ONLY\_SER

This level of Spark store the RDD as serialized Java object (one-byte array per partition). It is more space efficient as compared to deserialized objects, especially when it uses fast serializer. But it increases the overhead on CPU. In this level the storage space is low, the CPU computation time is high and the data is stored in-memory. It does not make use of the disk.

d. MEMORY\_AND\_DISK\_SER

It is similar to **MEMORY\_ONLY\_SER**, but it drops the partition that does not fits into memory to disk, rather than recomputing each time it is needed. In this storage level, The space used for storage is low, the CPU computation time is high, it makes use of both in-memory and on disk storage.

e. DISK\_ONLY

In this storage level, RDD is stored only on disk. The space used for storage is low, the CPU computation time is high and it makes use of on disk storage.  
Refer this guide for the [detailed description of Spark in-memory computation](http://data-flair.training/blogs/apache-spark-in-memory-computing/).

6. How to Unpersist RDD in Spark?

Spark monitor the cache of each node automatically and drop out the old data partition in the LRU (least recently used) fashion. LRU is an algorithm which ensures the least frequently used data. It spills out that data from the cache. We can also remove the cache manually using **RDD.unpersist()** method.

7. Conclusion

Hence,**Caching** or **persistence** are the optimization techniques for interactive and iterative Spark computations. It helps to save intermediate results so we can reuse them in subsequent stages. These intermediate results as RDDs are thus kept in memory (default) or more solid storages like disk and/or replicated.