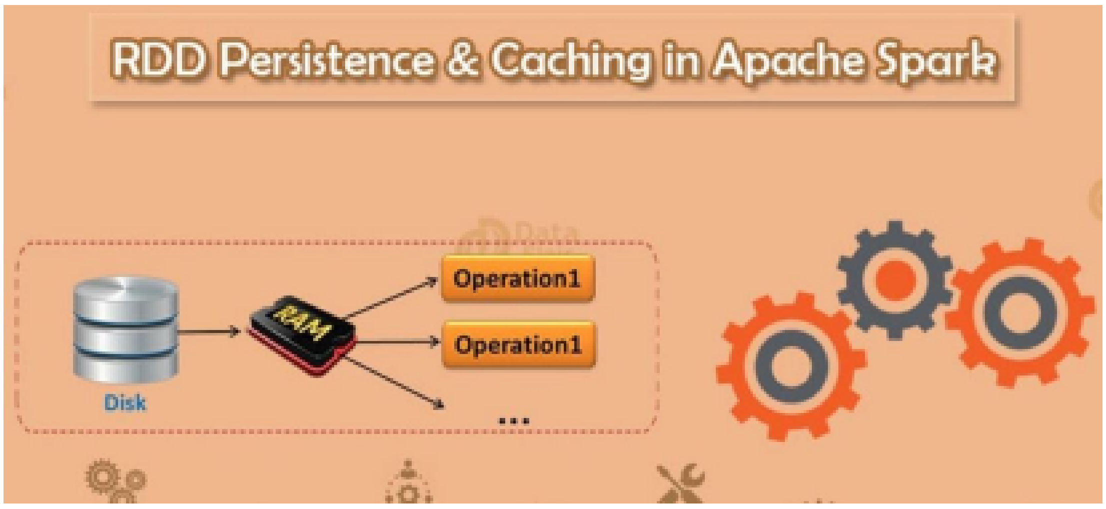
Topic 1:

RDD-persistence?

Spark RDD persistence is an optimization technique which saves the result of RDD evaluation in cache memory. Using this we save the intermediate result so that we can use it further if required. It reduces the computation overhead.



Need of Persistence in Apache Spark

In Spark, we can use some RDD's multiple times. when 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.

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

Storage Level	**Description**
MEMORY_ONLY	It stores the RDD as deserialized Java objects in the JVM. This is the default level. If the RDD doesn't fit in memory, some partitions will not be cached and recomputed each time they're needed.
MEMORY_AND_DISK	It stores the RDD as deserialized Java objects in the JVM. If the RDD doesn't fit in memory, store the partitions that don't fit on disk, and read them from there when they're needed.

MEMORY_ONLY_SER	It stores RDD as serialized Java objects.
(Java and Scala)	This is generally more space-efficient than deserialized objects.
MEMORY_AND_DISK_SER (Java and Scala)	It is similar to MEMORY_ONLY_SER, but spilt partitions that don't fit in memory to disk instead of recomputing them.
DISK_ONLY	It stores the RDD partitions only on disk.
OFF_HEAP (experimental)	It is similar to MEMORY_ONLY_SER, but store the data in off-heap memory. The off-heap memory must be enabled.

<u>Topic 2:RDD Shared Variables</u>

In Spark, when any function passed to a transformation operation, then it is executed on a remote cluster node. It works on different copies of all the variables used in the function. **a. Broadcast variable**

The broadcast variables support a read-only variable cached on each machine rather than providing a copy of it with tasks. Spark uses broadcast algorithms to distribute broadcast variables for reducing communication cost.

B. Accumulator

The Accumulator are variables that are used to perform associative and commutative operations such as counters or sums.

Tpoic 3:Removing RDD from Cache:

We can remove an RDD from the cache by calling the **unpersist()** method on it, which will free up the memory that it was occupying.

myDataFrame.unpersist()

to calling **unpersist()** does not immediately delete the DataFrame from disk or storage. It only releases the memory used by the DataFrame in the Spark cluster. The actual deletion from disk or storage depends on the underlying storage system (e.g., HDFS, S3) and its configuration.

<u>Topic 4: Deploying to a SPARK Cluster</u>

ALREADY MANUAL HAS BEEN SHARED WITH YOU.

-