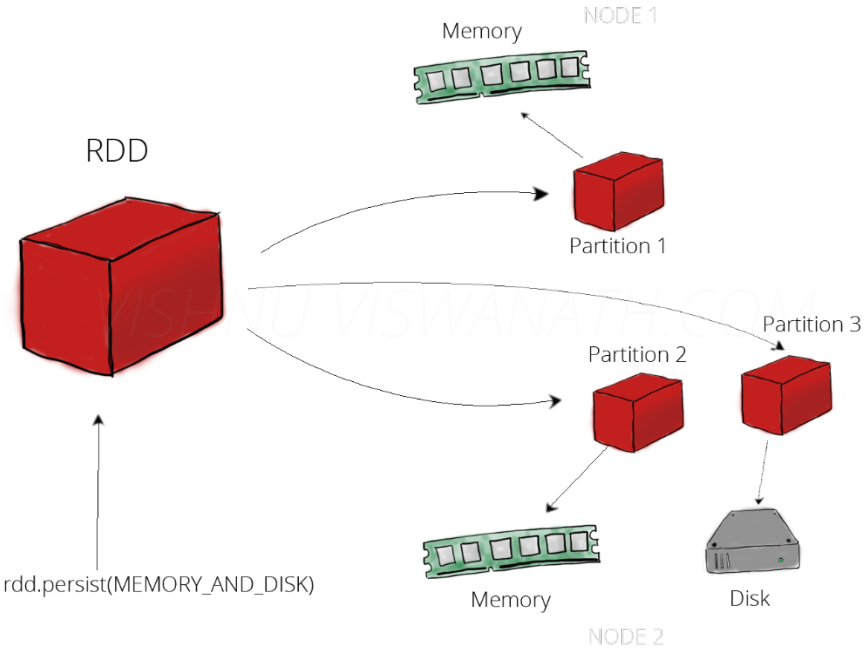
How the RDDS are partitioned in the memory:

one **partition** is created for each block of the file in HDFS which is of size 64MB.

As RDD stores the value in memory, the data which does not fit in memory is either recalculated or the excess data is sent to disk.



partitioning in general is nothing but dividing the data set into parts. If the data set is too large it cannot fit into a single node in the cluster. So we need to split the data set into different parts and store it into different nodes of the cluster.

In Hadoop the data set is divided into different parts known as blocks and they are stored into different nodes of the Hadoop cluster similarly in spark the input data set is divided into different parts called partitions and are stored into worker nodes of the spark cluster. The spark cluster has worker nodes whereas the Hadoop cluster has data nodes.

Since the core abstraction of the spark is rdd and since the data is split into partitions before storing into the spark cluster the process is called rdd partitioning. In spark, every node in the Spark cluster contains one or more partitions .

Apache Spark automatically partitions RDDs and distributes the partitions across different nodes.

If due to a worker node failure any partition of an RDD is lost, then that partition can be re-computed from the original fault-tolerant dataset using the lineage of operations.

HDFS is distributing files on data nodes. So if I put a "file.txt" on the filesystem, it will be split into partitions. For example, If you have a 30GB uncompressed text file stored on HDFS, then with the default HDFS block size setting (128MB) it would be stored in 235 blocks, which means that the RDD you read from this file would have 235 partitions