**MAPREDUCE JOIN**

Joining two large dataset can be achieved using MapReduce Join. However, this process involves writing lots of code to perform actual join operation.

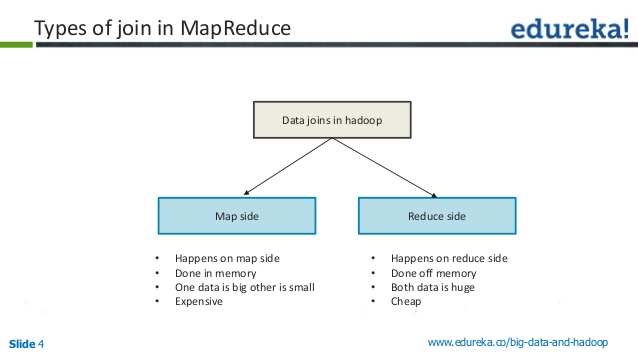
Joining of two datasets begin by comparing size of each dataset. If one dataset is smaller as compared to the other dataset then smaller dataset is distributed to every datanode in the cluster. Once it is distributed, either Mapper or Reducer uses smaller dataset to perform lookup for matching records from large dataset and then combine those records to form output records.

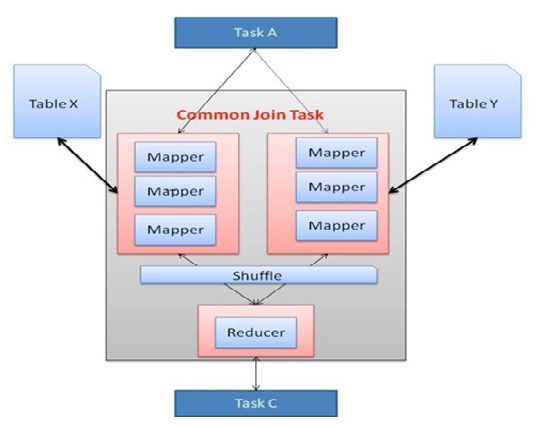
Depending upon the place where actual join is performed, this join is classified into-

**1. Map-side join -** When the join is performed by the mapper, it is called as map-side join. In this type, the join is performed before data is actually consumed by the map function. It is mandatory that the input to each map is in the form of a partition and is in sorted order. Also, there must be an equal number of partitions and it must be sorted by the join key.

**2. Reduce-side join -** When the join is performed by the reducer, it is called as reduce-side join. There is no necessity in this join to have dataset in a structured form (or partitioned).

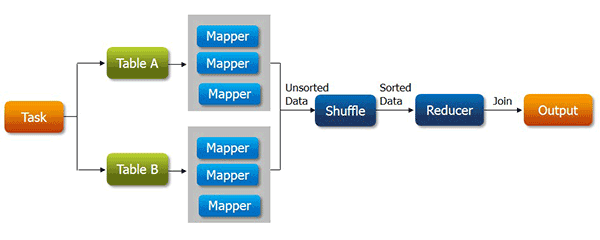
Here, map side processing emits join key and corresponding tuples of both the tables. As an effect of this processing, all the tuples with same join key fall into the same reducer which then joins the records with same join key.





1. **MapSide Join:**

Whenever, we apply join operation, the job will be assigned to a Map Reduce task which consists of two stages- a ***‘Map stage***’ and a ‘***Reduce stage***’. A mapper’s job during Map Stage is to “read” the data from join tables and to “return” the **‘join key’** and **‘join value’** pair into an intermediate file. Further, in the shuffle stage, this intermediate file is then sorted and merged. The reducer’s job during reduce stage is to take this sorted result as input and complete the task of join.

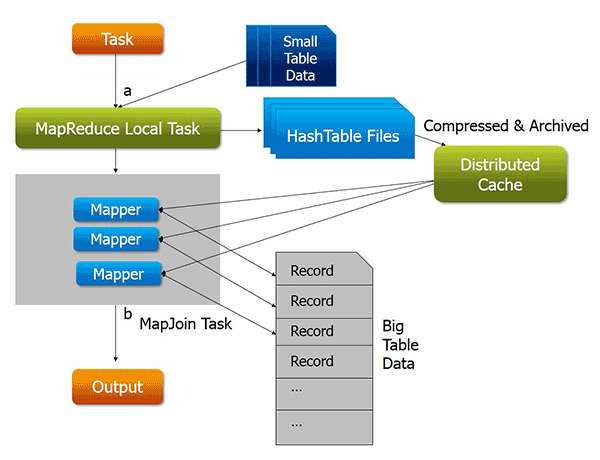


* Map-side Join is similar to a join but  all the task will be performed by the mapper alone.
* The Map-side Join will be mostly suitable for small tables to optimize the task.

**How will the map-side join optimize the task?**

Assume that we have two tables of which one of them is a small table. When we submit a map reduce task, a Map Reduce local task will be created before the original join Map Reduce task which will read data of the small table from HDFS and store it into an in-memory hash table. After reading, it serializes the in-memory hash table into a hash table file.

***In the next stage,*** when the original join Map Reduce task is running, it moves the data in the hash table file to the Hadoop distributed cache, which populates these files to each mapper’s local disk. So all the mappers can load this persistent hash table file back into the memory and do the join work as before. The execution flow of the optimized map join is shown in the figure below. After optimization, the small table needs to be read just once. Also if multiple mappers are running on the same machine, the distributed cache only needs to push one copy of the hash table file to this machine.



## Advantages of Map-side join:

* Map-side join helps in minimizing the cost that is incurred for sorting and merging in the shuffle and reduce stages.
* Map-side join also helps in improving the performance of the task by decreasing the time to finish the task.

#### ****Disadvantages of Map-side join:****

* Map side join is adequate only when one of the tables on which you perform map-side join operation is small enough to fit into the memory.  Hence it is not suitable to perform map-side join on the tables which are huge data in both of them.

Hence, **Map-side Join** is your best bet when one of the tables is small enough to fit in memory to complete the job in a short span of time.

In **Real-time environment**, you will be have data-sets with huge amount of data. So performing analysis and retrieving the data will be time consuming if one of the data-sets is of a smaller size. In such cases *Reduce-side join* will help to complete the job in less time.

2. **ReduceSide Join:**

**Joins** of datasets done in the **reduce** phase are called **reduce side joins**. **Reduce side joins** are easier to implement as they are less stringent than map-**side joins** that require the data to be sorted and partitioned the same way.

