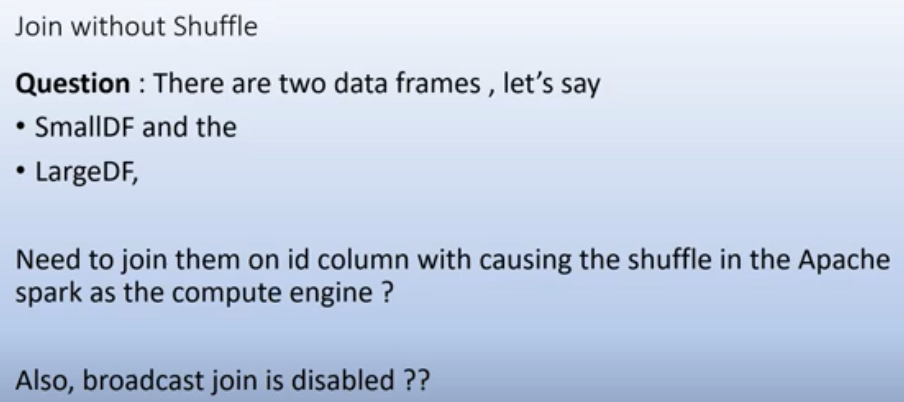
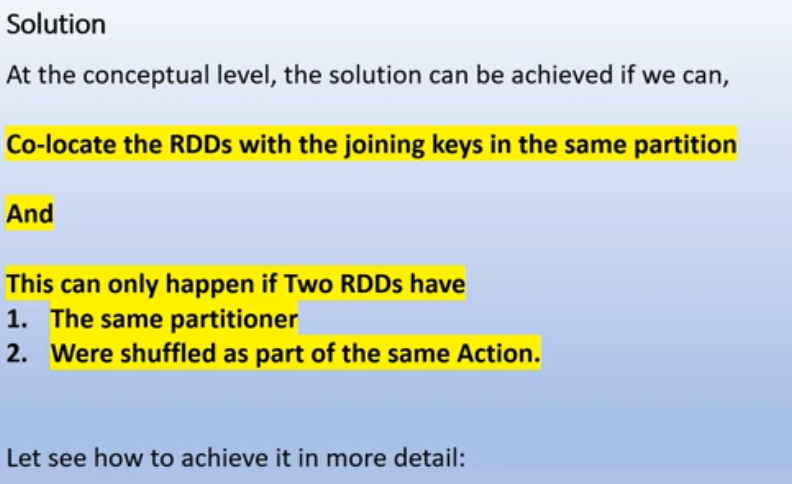
**JOINS WITHOUT SHUFFLE**

To avoid shuffle while using. As we know that most shuffling occurs during joins operations.

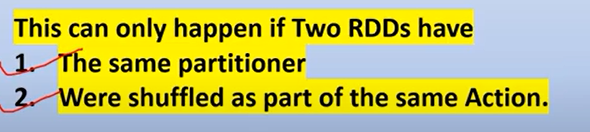


If we apply join on df, in which one df is really small like default threshold for small df is like 10 mb so we can say we use broadcast join and in that shuffling is not occur. By default broadcast join is enabled. In this the small dataframe is copied to the node where the large dataframe and partitions are residing and then the join will happen and partitions will co locate and thus there would not be ay shuffling.

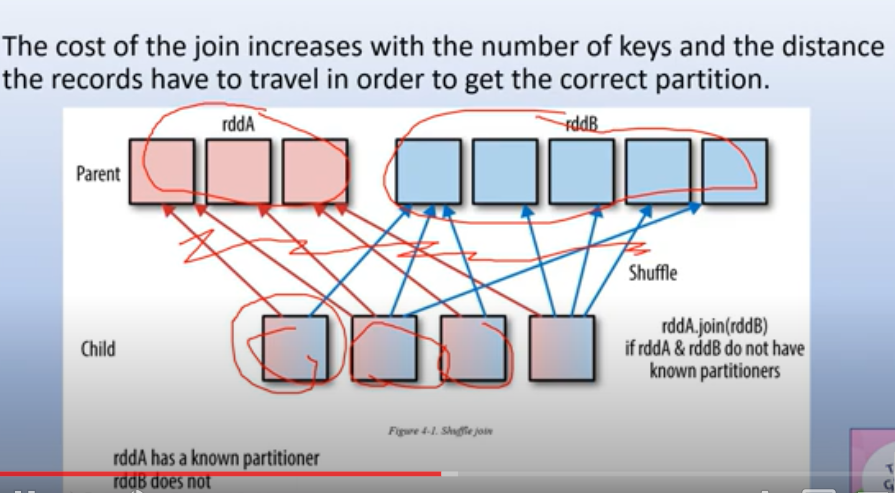
Now if broadcast join disabled so what can you use



Shuffle will not happened if partition of two rdd are co located ,they are part of the same partition and they are on the same node so when the join will happen it will match the corresponding record an then there is no shuffling. The solution lies that we have to co locate the rdds with the partition with the joining keys so we will have to make sure that records with the joining key are co ocated in the same partition. So how we can do that we have to tweek the default partitioner or we have to do partition in the rdds in such a way that joining key with the same value will come in the same partition and this can only happen



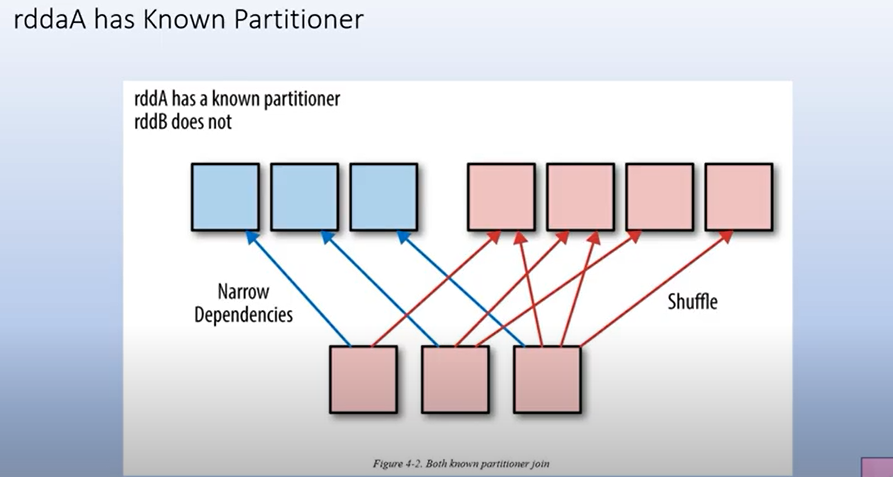
If we do join then shufffling occur in this manner. RddA and rddB both are at diff partition so when you join them both lots of shuffling will occur, bring the data from the common keys



As the both rdds or df need to use same partitioner is one of the key point to avoid shuffling during join.

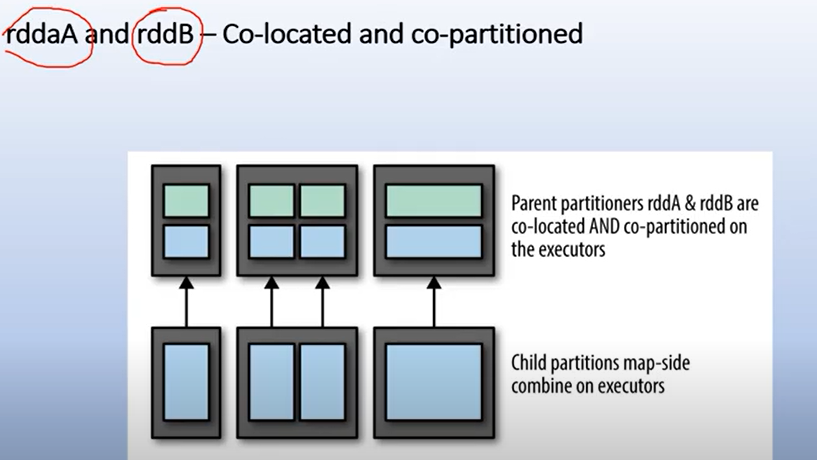
As spark have two algorithm used for partition ie. Hash partitioner and range partitioner.

Now we partition rddA with the known partition. Known partition means if you have rdd, you can simply check rdd.partitioner.get and it will tell you which partitioner is used under the hood of spark. If this give you none then there is no known partitioner that means explicitly you don’t use any partitioner to partition it. Other wise if you use any partitioner like hash partiioner to partition your rdd it will give you name of that partitioner. So here we try to use the known partitioner, means we explicitly di the partitioning of rddA with the hash partitioning algorithm. So you can see that rddA has now narrow transformation, the only shuffling is happen at rddB side because here we don’t have known partitioner



Now we have to parttion both rddA and rddB with the same known partitioner.

In this way the joining key of both rdd are located at same partition



In this code, we disable the broadcast join and sortmerge join and create two partition as by default the shuffle partition is 200

We create the hashpartitioner with 2 partition,in this way shuffling will not happen during the join.

The transformation also get narrow transformation so lots of performance boost occur at that point .

As when we do partition, shuffling occur at that point which is also a onetime process

