**Session 27**

**Assignment 1**

**Problem Statement**:

Explain the following in brief with an example.

* Map side Join
* Reduce side Join
* Bucket Map Join
* SMBM Join

**Map Side Join:**

* The map side join is performed in the mapper phase of the map reduce process.
* In this join, the small file are stored in distributed cache and another file’s input path has to be given in the argument.
* The inputs for each map must be partitioned and sorted in a certain unique way. Each input dataset must be divided into same number of partitions, and it must be sorted by the same key (the join key) in each source.
* All the records for a particular key must reside in the same partition and this process is mandatory.
* A map side join can be used to join the outputs of several jobs that have the same number of reducers, the same keys and output files which are no bigger than the HDFS block size.
* Map side join is similar to a hive 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.
* Using ‘org.apache.hadoop.mapreduce.lib.join.CompositeInputFormat’ class we can achieve the map side join.

**Advantages of Map-Side join:**

* The cost incurred for sorting and merging in the shuffle and reduce stages can be minimized.
* The performance of the task can be improved by using map side join which decreases the time to finish the task.

**Disadvantages of Map-Side Join:**

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

**Reduce Side Join:**

* In the reduce side join, the reducer is responsible for performing the join operation.
* It is comparatively simple and easier to implement than the map side join as the sorting and shuffling phase sends the values having identical keys to the same reducer and therefore, by default, the data is organized for us.
* But this join is less efficient because datasets have to go through the Map Reduce shuffle phase. The records with the same key are brought together in the reducer. The Secondary Sort technique can also be used to control the order of the records.

The steps in reduce side join:

* Mapper reads the input data which are to be combined based on common column or join key.
* The mapper processes the input and adds a tag to the input to distinguish the input belonging from different sources or databases or having different data sets.
* The mapper gives the output with an intermediate key-value pair where the key is nothing but the join key.
* After the sorting and shuffling phase, the key and the list of values are generated for the reducer.
* Now, the reducer joins the values present in the list with the key to give the final aggregated output.

The reduce side join procedure generates a huge network I/O traffic in the sorting and reducer phase where the values of the same key are brought together. So, if the dataset is different and large in number having millions of values, there is a high chance of encountering an **OutOfMemoryException** i.e. the RAM is full and thus overflown.

**Advantages:**

* It is very easy to implement as the inbuilt sorting and shuffling algorithm takes place in the MapReduce framework which combines values of the same key and send it to the same reducer.
* In the reduce side join, input does not require to follow any strict format and therefore, the join operation can be performed on unstructured data as well.

**Bucket Map Join:**

* A bucket map join is used when the tables are large and all the tables used in the join are bucketed on the join columns.
* In this type of join, one table should have buckets in multiples of the number of buckets in another table.
* For example, if one table has 2 buckets then the other table must have either 2 buckets or a multiple of 2 buckets (2, 4, 6, and so on).
* If the preceding condition is satisfied then the joining can be done at the mapper side only, otherwise a normal inner join is performed. This means that only the required buckets are fetched on the mapper side and not the complete table.
* That is, only the matching buckets of all small tables are replicated onto each mapper. Doing this, the efficiency of the query is improved drastically.
* In a bucket map join, data is not sorted and the Hive does not support a bucket map join by default.
* The following property needs to be set to true for the query to work as a bucket map join:

set **hive.optimize.bucketmapjoin = true;**

Prerequisites for bucket map join:

* Tables being joined are bucketized on the join columns.
* The number of buckets in one table is a multiple of the number of buckets in the other table, the buckets can be joined with each other

**SMB (Sort Merge Bucket) Join:**

* It is another Hive join optimization technique where all the tables need to be bucketed and sorted.
* In this case joins are very efficient because they require a simple merge of the presorted tables.
* Before doing anything, we should do the following steps
* set hive.enforce.bucketing=true;
* set hive.enforce.sorting=true;
* set hive.enforce.sortmergebucketmapjoin=false;
* set hive.auto.convert.sortmerge.join=true;
* set hive.optimize.bucketmapjoin = true;
* set hive.optimize.bucketmapjoin.sortedmerge = true;
* In sortmergebucket map join while creating the bucketed table the contents should be sorted by the column which is going to be used for cluster by.
* In SMB join, each mapper reads a bucket from the first table and the corresponding bucket from the second table and then a merge sort join is performed.

**Advantages of SMB join:**

* SMB join can best be used when the tables are large. In SMB join the columns are bucketed and sorted using the join columns.
* Sort Merge Bucket map join is the fastest of all the joins.
* Sort Merge Bucket (SMB) join in hive is mainly used as there is no limit on file or partitioning or table join.

**Disadvantages of SMB join:**

* All tables should have the same number of buckets in SMB join.