Hive on Spark: Sorted Merge Bucket Join

MapReduce Background:

Sorted Merge Bucket join (SMB join) is another join-type, similar to traditional sort-merge join algorithm of databases. It is performed in the mapper.

Conditions:

1. Appropriate flags are turned on, check documentation.
2. All tables bucketed by join column, number of buckets of each table being multiples of each other.
3. All tables sorted by join column.

In an n-way join, this is the algorithm summary:

1. Pick one table as big-table, other n-1 tables are small tables.
2. Mapper runs against the big-table buckets. In each mapper:
    a. While traversing through big-table bucket rows, read rows from corresponding small-table buckets. Both are in sorted order.
    b. Sort-merge Join row from big table with row from small table(s). If a certain table has no more matching rows, advance (fetch next rows) from rest of tables until more matching rows are found.

The primary requirement of the algorithm is that per bucket, rows from big and small tables are read by mapper in sorted order. From then on, it is a standard sort-merge join.

The ability to find, given a bucket of the big table, corresponding bucket of a small tables, comes from the requirement that all tables have compatible number of buckets (multiples of each other). With this, rows with the same key must have hashed predictably to corresponding buckets.

Implementation Details

In the compilation-optimization phase, if a join meets the critieria specified above, it does the following operator-tree transformation from reduce-side join to SMB join:

MapRedWork (Reduce-Side Join)

- MapWork (input=src1, src2, src3)
  - Src1-(Root): TableScan -> Filter -> ReduceSink
  - Src2-(Root): TableScan -> Filter -> ReduceSink
  - Src3-(Root): TableScan -> Filter -> ReduceSink
- ReduceWork
  - JoinOperator -> Filter -> FileSink (output)

MapRedWork (SMB Join)

- MapWork (input=src3)
  - Src1: TableScan1 -> Filter1 -> DummyStoreOp1
  - Src2: TableScan2 -> Filter2 -> DummyStoreOp2
  - Src3 (Root): TableScan3 -> Filter3 ->
    {DSO1, DSO2, Fil3} -> SMBJoinOp -> FileSink (Output)
  - LocalWork
    - Src1 information (TS1)
    - Src2 Information (TS2)

In a nutshell:

1. Remove small tables from direct mapper input. Take these detached map operator-trees, end them with DummyStoreOperator, which does nothing but store the output. We will return later to these.
2. Take big table's map-operator-tree, make it the only root operator for that mapper, ie, now only this operator gets fed from the mapper input.
3. Create an SMBMapJoinOpertor, after the big-table map-operator of (2).
4. Take rest of original post-join reduce-operator tree, append it to after the SMBMapJoinOp. In other words, move the post-join logic of reducer to the mapper.

The SMBMapJoinOperator performs the actual logic of sort-merge join. As noted, the mapper reads rows in sorted order from big table, runs them through the big-table operator tree, to the SMBMapJoinOperator. When appropriate according to the sort-merge algorithm, the SMBMapJoinOperator uses the detached small-table operator-trees to fetch and process next rows from small tables, which are also in also in sorted order.

For getting next rows from small tables:

5. The MapWork uses a LocalWork as a container for back-pointers of the small table operator-tree, as they had been detached in (1). The SMBJoinOperator uses this information to fetch and process the small table rows on demand during its sort-merge join algorithm.

Hive on Spark

This solution is already distributed and will work well for Hive on Spark. We will run the exact same mapper task, with same input, as part of SparkWork.

Enabling SMB Join on Hive on Spark is thus making sure we preserve the same operator-tree, but encapsulate it accordingly in a SparkWork.

SparkWork (SMB Join)

- MapWork (input=src3)
  - Src1: TableScan1 -> Filter1 -> DummyStoreOp1
  - Src2: TableScan2 -> Filter2 -> DummyStoreOp2
  - Src3 (Root): TableScan3 -> Filter3 ->
    {DSO1, DSO2, Fil3} -> SMBJoinOp -> FileSink (Output)
  - LocalWork
    - Src1 information (TS1)
    - Src2 Information (TS2)

Future Considerations:

1. Consider converting SMBJoin to MapJoin if SMBJoin becomes sub-optimal (too many partitions).