Other Strategies for Query Optimization

There are some other techniques for improving Hive and Impala query performance that are not described in this course. Two of these techniques are briefly described below. Further details of these methods are beyond the scope of this course, but you can follow the links below to learn more.

Bucketing (Hive only)

Bucketing is a Hive-only technique that is similar to partitioning. Recall that partitioning is an approach for improving Hive query performance. Partitioning divides a table's data into separate subdirectories based on the values from one or more partition columns, which have a limited number of discrete values. Hive also offers another way of subdividing data, called bucketing.

Bucketing stores data in separate *files*, not separate subdirectories like partitioning. It divides the data in an effectively random way, not in a predictable way like partitioning. When records are inserted into a bucketed table, Hive computes hash codes of the values in the specified bucketing column and uses these hash codes to divide the records into buckets. For this reason, bucketing is sometimes called *hash partitioning*. The goal of bucketing is to distribute records evenly across a predefined number of buckets. Bucketing can improve the performance of joins if all the joined tables are bucketed on the join key column.

For more on bucketing, see the page of the Hive Language Manual describing bucketed tables, at https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL+BucketedTables.

Indexing (Hive only)

If you have worked with relational databases, you may be familiar with *indexes*.

Indexes can greatly improve the speed of queries that search for specific values in certain columns. By indexing the columns you're filtering by, you can avoid the need to do a full table scan at query time. However, relational database implementations of indexing typically depend on the database system controlling all data that is added to tables. Since Hive and Impala do not work this way, the use of indexing would not confer the same benefits with Hive and Impala.

Early versions of Hive (but not Impala) include a limited implementation of indexing. As in relational databases, indexes in Hive can improve the speed of some queries. However, the speedup from indexing is typically not as dramatic, and building and maintaining indexes with Hive has high costs in terms of disk space and CPU utilization. In fact, indexing is no longer supported in Hive as of version 3.0.0. Cloudera recommends using Cloudera Search (an implementation of Apache Solr) if you need extensive indexing.

For more on indexing, see the page of the Hive Language Manual describing indexing, at https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Indexing.