Map Reduce and Design Patterns
Lecture 5

Fang Yu

Software Security Lab.
Department of Management Information Systems
College of Commerce, National Chengchi University
http://soslab.nccu.edu.tw

Cloud Computation, April 7, 2015

Join Patterns

A join is an operation that combines records from two or more data sets based on a field or set of fields, known as the foreign key. The foreign key is the field in a relational table that matches the column of another table, and is used as a means to cross-reference between tables.

- MapReduce is good at processing large data sets by looking at every record or group in isolation by design.
- Joining two very large data sets together does not fit into the paradigm

Join Patterns

- inner join: records that are on the same foreign key
- outer join (left, right, full): unmatched records are shown in the final table as well
- anti join: full outer join - inner join
- cartesian product: takes each record from a table and matches it up with every record from another table

Reduce Side Join

Simple to implement. It supports all the different join operations

- Mapper: The foreign key is written as the output key, and the entire input record as the output value.
- Partitioner: Distribute the intermediate key/value pairs more evenly across the reducers.
- Reducer: Collect the values of each input group into temporary lists. These lists are then iterated over and the records from both sets are joined together
- Require a large amount of network bandwidth because the bulk of the data is sent to the reduce phase.
- Can be integrated with a Bloom filter to filter out some of mapper output

Replicated Join

A join operation between one large and many small data sets that can be performed on the map-side

- Mapper: The mapper is responsible for reading all files from the distributed cache during the setup phase and storing them into in-memory lookup tables. After this setup phase completes, the mapper processes each record and joins it with all the data stored in-memory.
- No reduce phase at all
- A strict size limit on all but one of the data sets to be joined. All the data sets except the very large one are essentially read into memory during the setup phase of each map task
- Useful only for an inner or a left outer join where the large data set is the left data set

Composite Join

Join very large data sets together. Requires the data to be already organized or prepared in a very specific way

- The driver code handles most of the work in the job configuration stage. It sets up the type of input format used to parse the data sets, as well as the join type to execute. The framework then handles executing the actual join when the data is read.
- Mapper: The two values are retrieved from the input tuple
- No reduce phase at all
- Data must first be sorted by foreign key, partitioned by foreign key, and read in a very particular manner in order to use this type of join.

Cartesian Product

Pair every record of multiple inputs with every other record. Rather than pairing data sets together by a foreign key, a Cartesian product simply pairs every record of a data set with every record of all the other data sets.

- Mapper: The record reader gives a pair of records to a mapper class, which simply writes them both out to the file system.
- No reducer, combiner, or partitioner is needed. This is a map-only job.
- Explosion: Resulting a table with size $|A| \times |B|$, e.g., a self-join of a measly million records produces a trillion records.