WEEK 9
MAP REDUCE

Arranged By:

Rajendra Rakha Arya Prabaswara

(1941720080/20)

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

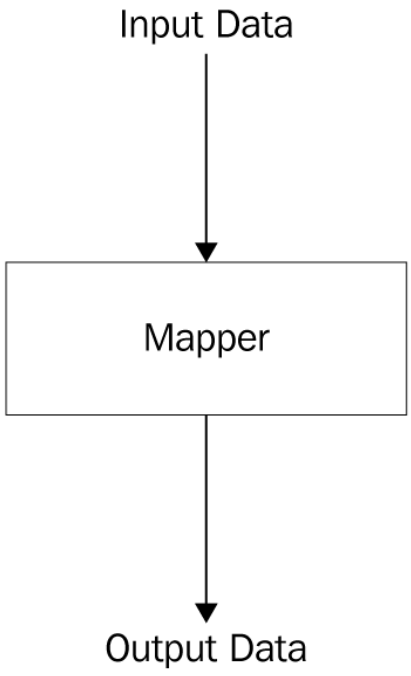
The MapReduce framework

- MapReduce job types:
  - Single mapper jobs
  - Single mapper reducer jobs
  - Multiple mappers reducer jobs
- MapReduce patterns:
  - Aggregation patterns
  - Filtering patterns
  - Join patterns

Explain the types of jobs and the MapReduce pattern on slide 4 along with usage scenarios!

A. MapReduce Job Types

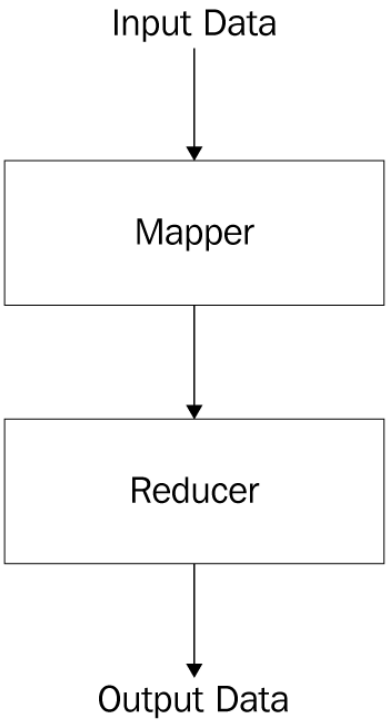
- **Single Mapper Jobs**
  Single mapper jobs are used in transformation use cases. If we want to change only the format of data, such as some kind of transformation, then this pattern is used:



The input to a MapReduce job is a set of files in the data store that is spread out over the HDFS.

The map function takes a series of key/value pairs, processes each, and generates zero or more output key/value pairs.
In the mapper, code is executed on each key/value pair from the record reader to produce zero or more new key/value pairs, called the intermediate output of the mapper (which also consists of key/value pairs).

- **Single Mapper Reducer Jobs**
  Single mapper reducer jobs are used in aggregation use cases. If we want to do some aggregation the such as count, by key, then this pattern is used:

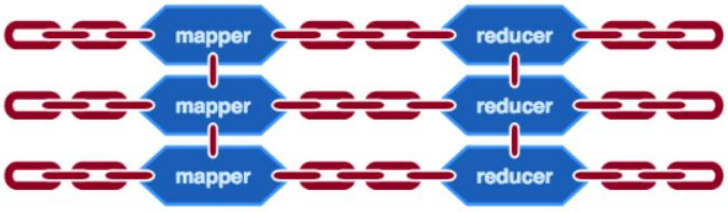


The function Of Reducer is passed the key and an iterator over all of the values associated with that key.

The output format translates the final key/value pair from the reduce function and writes it out to a file by a record writer.

The input to a MapReduce job is a set of files in the data store that is spread out over the HDFS.

- **Multiple Mappers Reducer Jobs**
  MapReduce is a computation abstraction that works well with The Hadoop Distributed File System (HDFS). It comprises of a “Map” step and a “Reduce” step. Map performs filtering and sorting into another set of data while Reduce performs a summary operation.



A lot of data processing tasks involve record-oriented preprocessing and postprocessing. For example, in processing documents for information retrieval, may have one step to remove stop words (words like a, the, and is that occur frequently but aren't too meaningful), and another step for stemming (converting different forms of a word into the same form, such as finishing and finished into finish.) You can write a separate MapReduce job for each of these pre- and postprocessing steps and chain them together, using IdentityReducer (or no reducer at all) for these steps.

B. MapReduce Patterns

- **Aggregation Patterns**

  Each reduce task is responsible for handling a subset of the map task output. Intermediate data is then copied from mapper tasks by the reducer tasks in order to group and aggregate the data. It is incredible what a wide range of problems can be solved with such a straightforward paradigm, from simple numerical aggregations to complex join operations and Cartesian products.
  aggregation is the sum of the counts of each part

  SQL

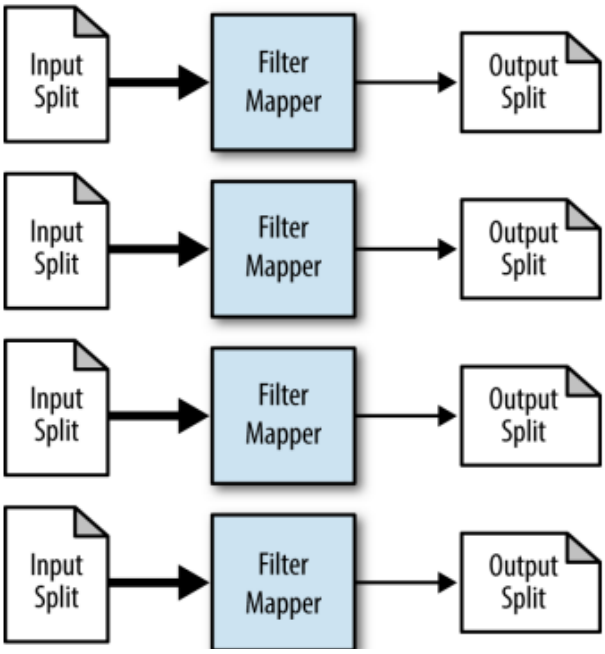
  The Numerical Aggregation pattern is analogous to using aggregates after a GROUP BY in SQL:

```
SELECT MIN(numericalcol1), MAX(numericalcol1),
       COUNT(*) FROM table GROUP BY groupcol2;
```

- **Filtering Patterns**

  As the most basic pattern, filtering serves as an abstract pattern for some of the other patterns. Filtering simply evaluates each record separately and decides, based on some condition, whether it should stay or go.

  The structure of the filter pattern



  Filtering is unique in not requiring the “reduce” part of MapReduce. This is because it doesn’t produce an aggregation. Each record is looked at individually and the evaluation of whether or not to keep that record does not depend on anything else in the data set. The mapper applies the evaluation function to each record it receives. Typically, the mapper outputs the same key/value type as the types of the input, since the record is left unchanged. If the evaluation function returns true, the mapper simply output the key and value verbatim.

- **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

Types Of Join :

1. INNER JOIN
2. OUTER JOIN
3. ANTIJOIN

Structure of the reduce side join pattern