BIG DATA ANALYTICS FRAMEWORK

Unit III

MapReduce is a programming model and a framework for processing large amounts of data in a distributed and parallel manner.

MapReduce is often used in big data processing to perform tasks such as data mining, log processing, and web indexing

MapReduce Tasks

- It involves two main tasks: the map task and the reduce task.
- The map task takes a set of input data and transforms it into a key-value pair.
- The reduce task takes the output of the map task and combines the data with the same key into a single value.

Choosing Key and Value Types for MapReduce Jobs

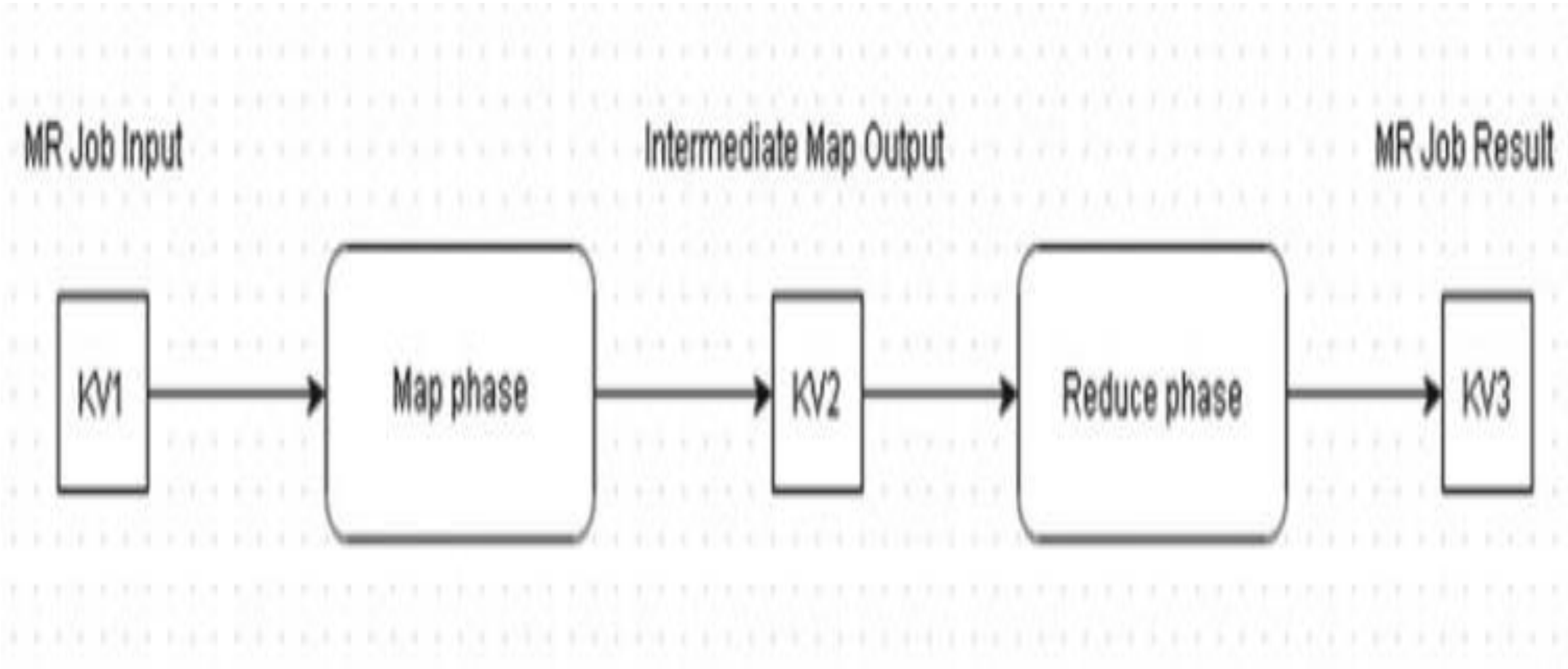
Choosing the appropriate key-value types for MapReduce jobs is an important consideration when designing MapReduce algorithms.

The key-value types determine how the data will be partitioned and sorted during the MapReduce process.

In general, the key should be chosen based on the data that needs to be grouped together for the reduce phase.

> For example, if you are processing log files, you might choose the IP address as the key so that all log entries from the same IP address are grouped together in the reduce phase.

Key/values in MapReduce



The key and value types are significant as follows.

- Each set of key/value pairs is homogeneous, which implies that all key/value pairs are the same type in terms of key type and value type. For example, all the key/value pairs in the KV1 set must be of the same type such as (string, string).
- The key/value pairs are not homogeneous across the different sets/phases; the key/ value pairs need not be the same type across the different phases.

For example, the key/value types in the MR Job input could be (string,string), and intermediate map output could be (integer, string) and MR job output could be (string, integer).

The requirement for the uniqueness of the keys is as follows.

- The MR Job input keys must be unique.
- The intermediate keys are not required to be unique.
- The MR job output keys are also not required to be unique.

Guidelines to determine the data types

.• Choose the map input key and value type based on the input data. For example, in a word count Map Reduce job in which the input is a document with text on each line, the suitable types for the input key/value pairs would be (integer, string) to represent the line number and text on each line.

• Choose the map phase intermediate output key/value types based on the final result of the MR job. For example for a word count MR job the final result represents the number of times a word occurs in the input. The map phase generates a key/value pair for each word in the input with the key as the word and value as 1, which in effect is including each word in the count. The map phase output key/value types are (string, integer). The output from the map phase could include multiple key/value pairs with the same key.

• The map phase output key/value types must match the reduce phase input key/ value types.

• The reduce phase output key/value types represent the required outcome of the MR job. For example, for a word count MR job the reduce phase output is the occurrence frequency for each word. The key/value pairs types for the reduce phase would be (string, integer).

The Lifecycle of a Mapper and a Reducer in a MapReduce Job

1. The InputFormat generates the InputSplits from the input. The InputFormat also generates the RecordReader to convert the InputSplits to records (key/value

pairs)

2. The MR framework launches a map task for each InputSplit in the input.

3. The RecordReader parses the InputSplit and generates key/value pairs (records) to be input to the Mapper's map() function.

4. The MR framework invokes the map() method for each key/value pair generated by the RecordReader from the InputSplit.

5. The map() method processes the input key/value pair.

6. The map() method may optionally use the Reporter object to report progress.

7. An OutputCollector collects the output key/value pairs. The map() function may generate no key/value pair at all.

8. The output key/value pairs are first written to the task's temporary output directory.

9. The map() method is invoked again until all key/value pairs generated by the RecordReader from the InputSplit have been processed.

10. If the map task completes successfully, the OutputCommitter commits the map task's output to the Spill buffer. The map output file has the name part-nnnn

The Relationship of Input Keys to Output Keys

MapReduce provides two phases of data processing, the map phase and the reduce phase