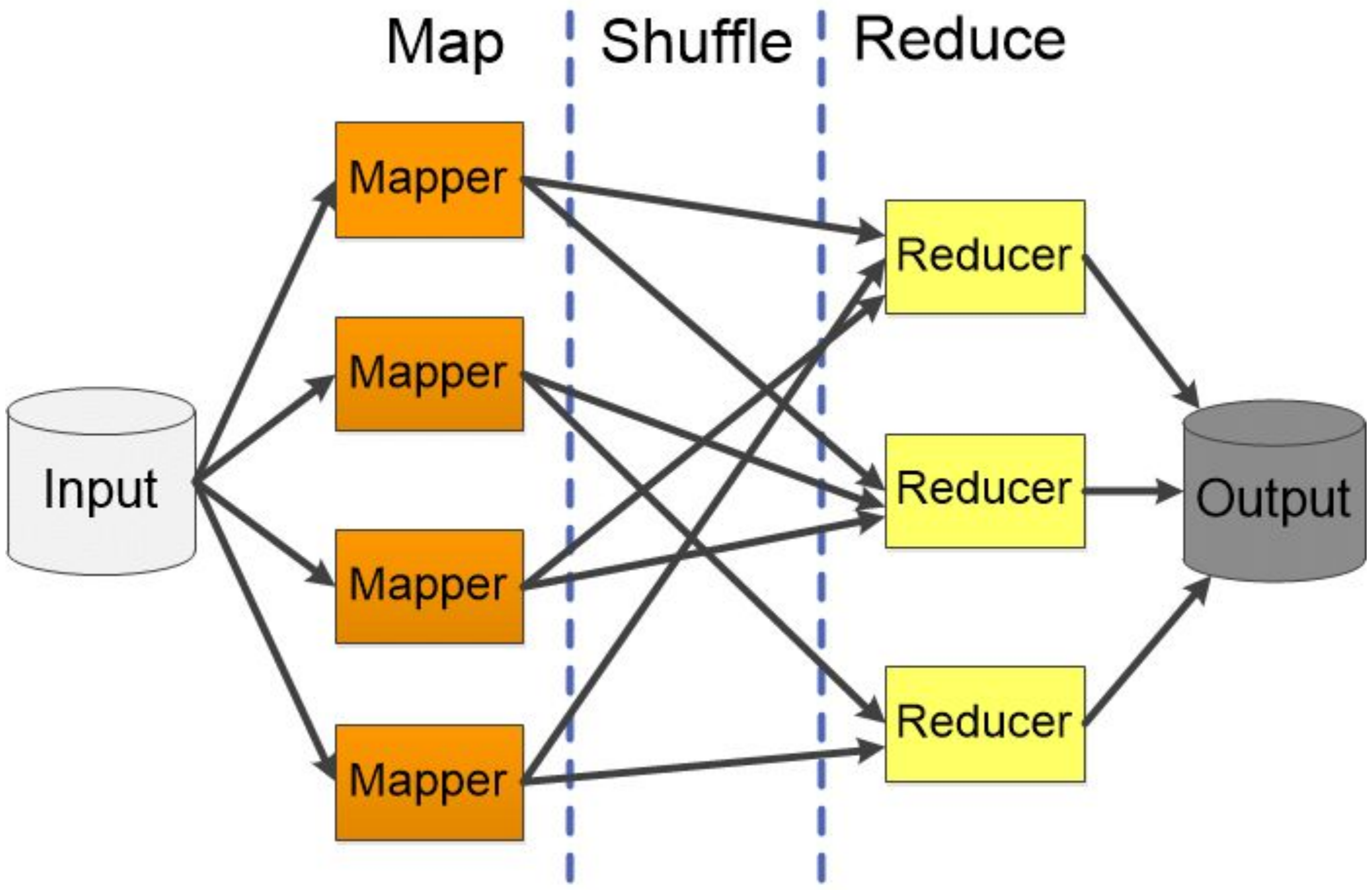
Introduction

- MapReduce is **a software framework for processing (large) data sets in a distributed fashion over a several machines**.
- The core idea behind MapReduce is mapping your data set into a collection of <key, value> pairs, and then reducing over all pairs with the same key.
- MapReduce is **a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster**.
- As the processing component, MapReduce is the heart of Apache Hadoop.
- The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform.
- MapReduce program executes in three stages, namely **map stage, shuffle stage, and reduce stage**.

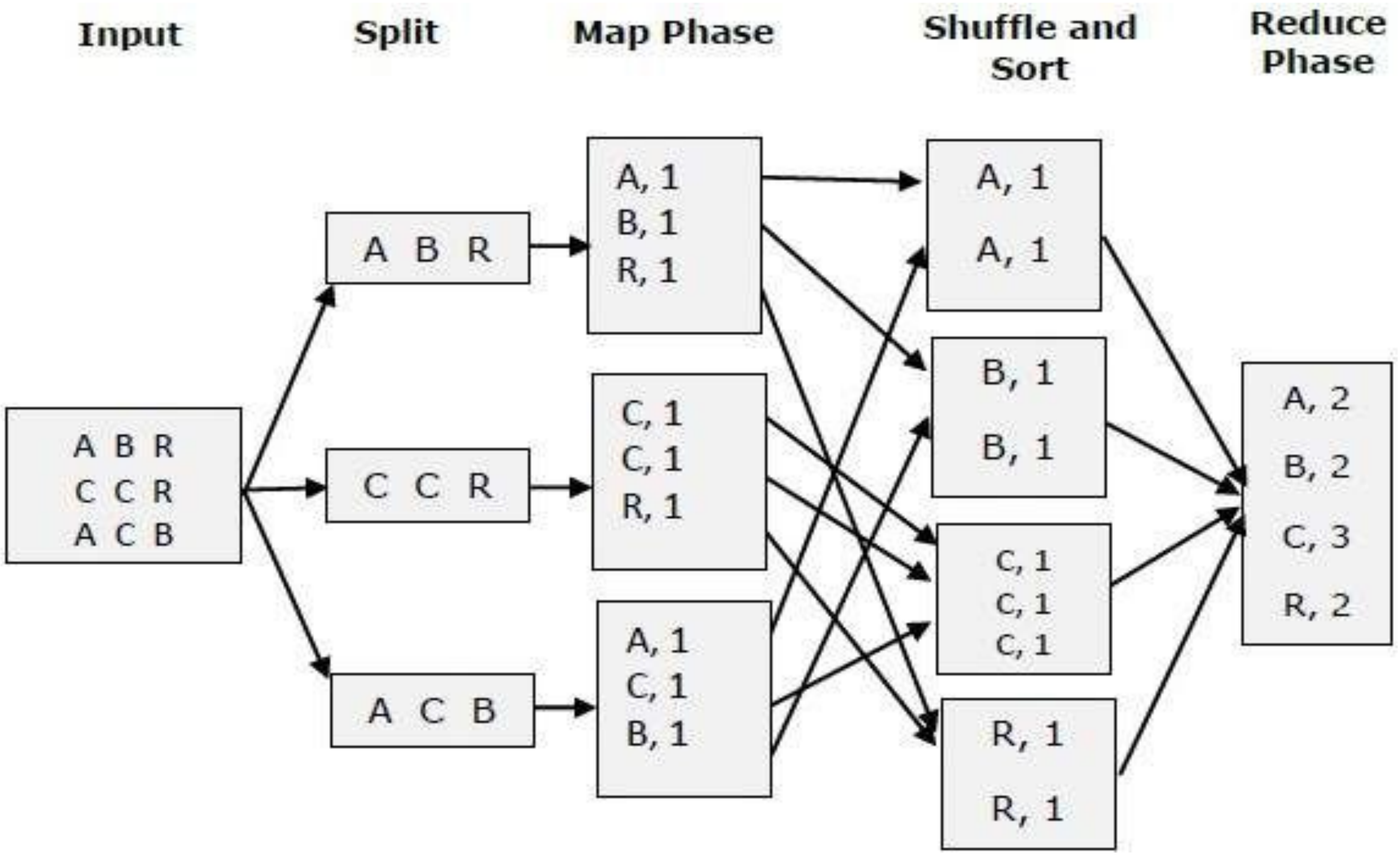
MapReduce Diagram



Task for MapReduce

- The MapReduce algorithm contains two important tasks, namely **Map and Reduce**.
- The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples(key-value pairs).

MapReduce - Example



Input phase

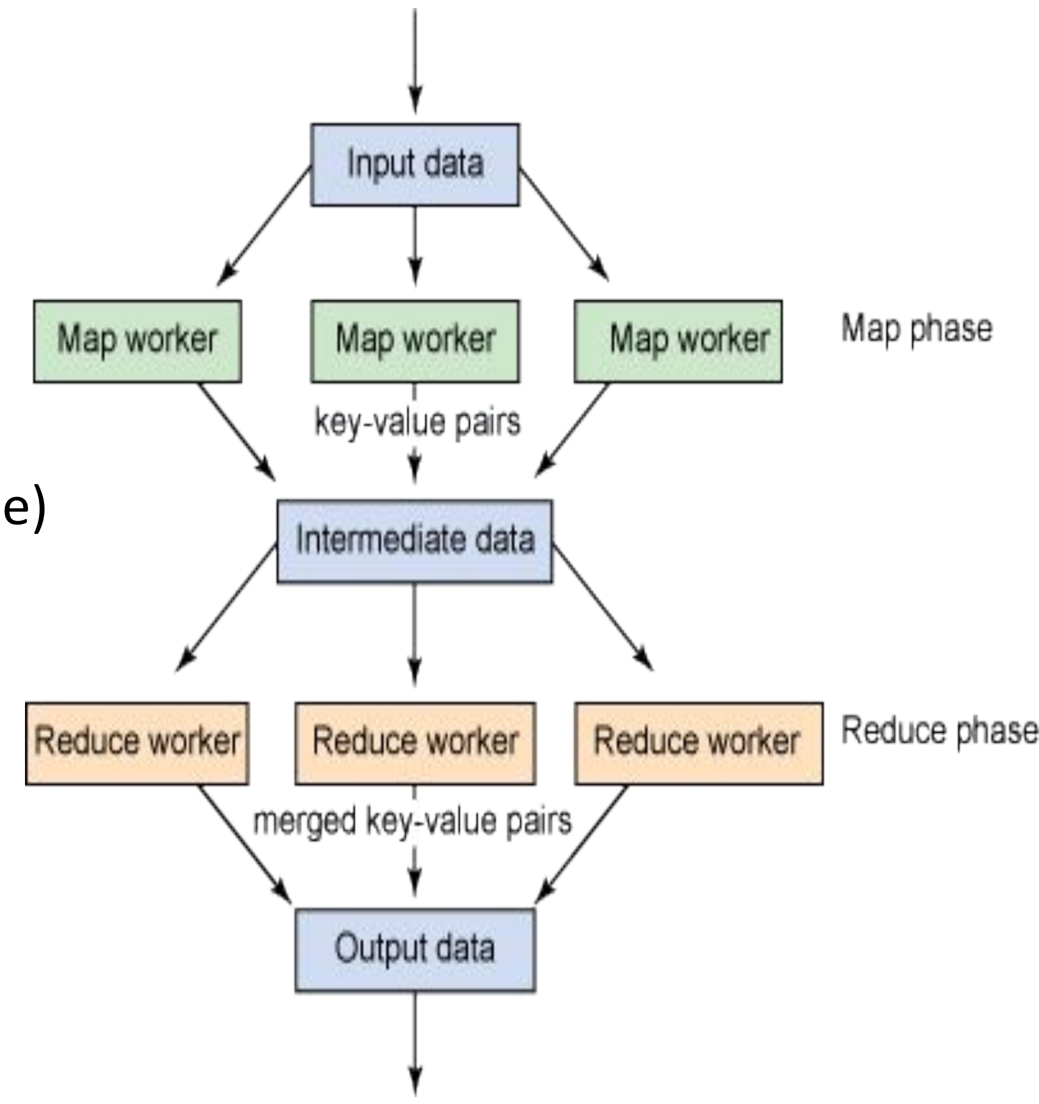
- Map task takes care of loading, parsing, transforming and filtering.
- The responsibility of reduce task is grouping and aggregating data that is produced by map tasks to generate final output.
- Each map task is broken into the following phases :
- 1. RecordReader
- 2. Mapper
- 3. Combiner
- 4. Partitioner

Output phase

- The output produced by map task is known as intermediate keys and values.
- These intermediate keys and values are sent to reducer.
- The reducer tasks are broken into the following phases :
- 1. Shuffle
- 2. Sort
- 3. Reducer
- 4. output Format

Major Components

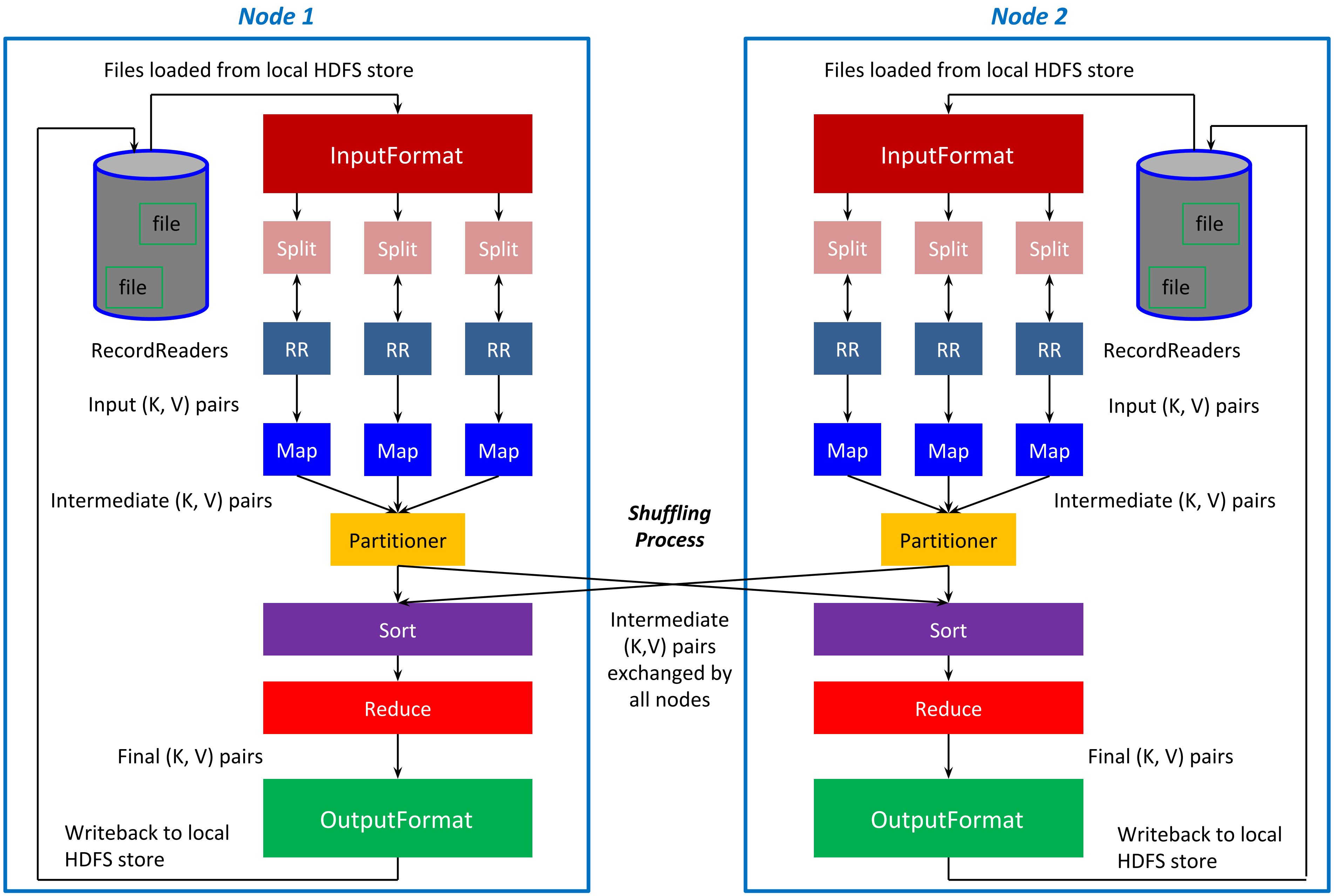
- User Components:
  - Mapper
  - Reducer
  - Combiner (Optional)
  - Partitioner (Optional) (Shuffle)
  - Writable(s) (Optional)
- System Components:
  - Master
  - Input Splitter
  - Output Committer



Mapper

- A mapper maps the input key-value pairs into a set of intermediate key-value pairs.
- Maps are individual tasks that have the responsibility of transforming input records into intermediate key-value pairs.
- Each map task is broken into the following phases :
- 1. RecordReader
- 2. Mapper
- 3. Combiner
- 4. Partitioner

Hadoop MapReduce: A Closer Look



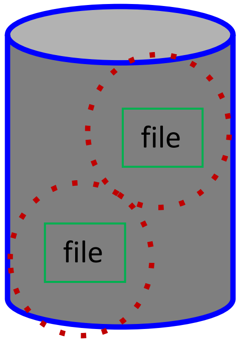
Input Files

- *Input files* are where the data for a MapReduce task is initially stored

- The input files typically reside in a distributed file system (e.g. HDFS)

- The format of input files is arbitrary

  - Line-based log files
  - Binary files
  - Multi-line input records
  - Or something else entirely

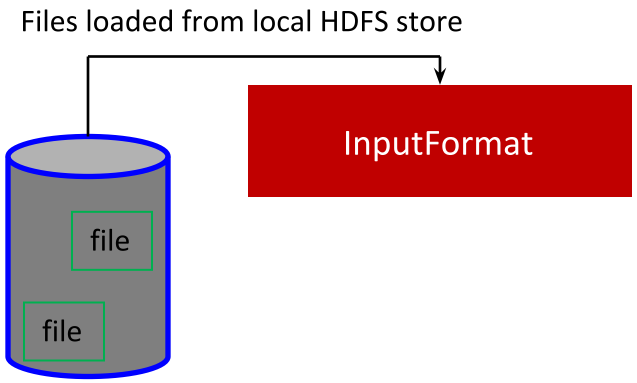


InputFormat

- How the input files are split up and read is defined by the *InputFormat*

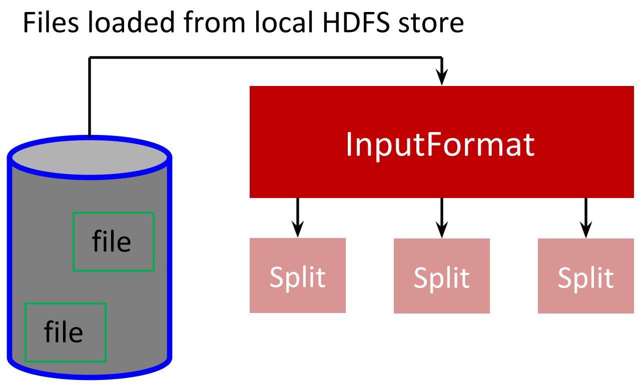
- InputFormat is a class that does the following:

  - Selects the files that should be used for input
  - Defines the *InputSplits* that break a file
  - Provides a factory for *RecordReader* objects that read the file



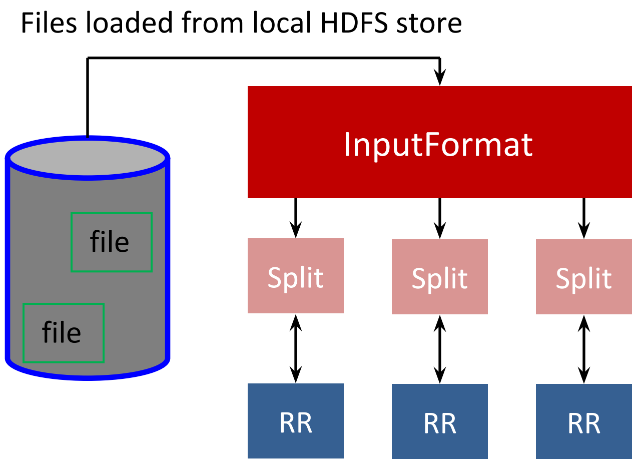
Input Splits

- An *input split* describes a unit of work that comprises a single map task in a MapReduce program
- By default, the InputFormat breaks a file up into 64MB splits
- By dividing the file into splits, we allow several map tasks to operate on a single file in parallel
- If the file is very large, this can improve performance significantly through parallelism
- Each map task corresponds to a *single* input split



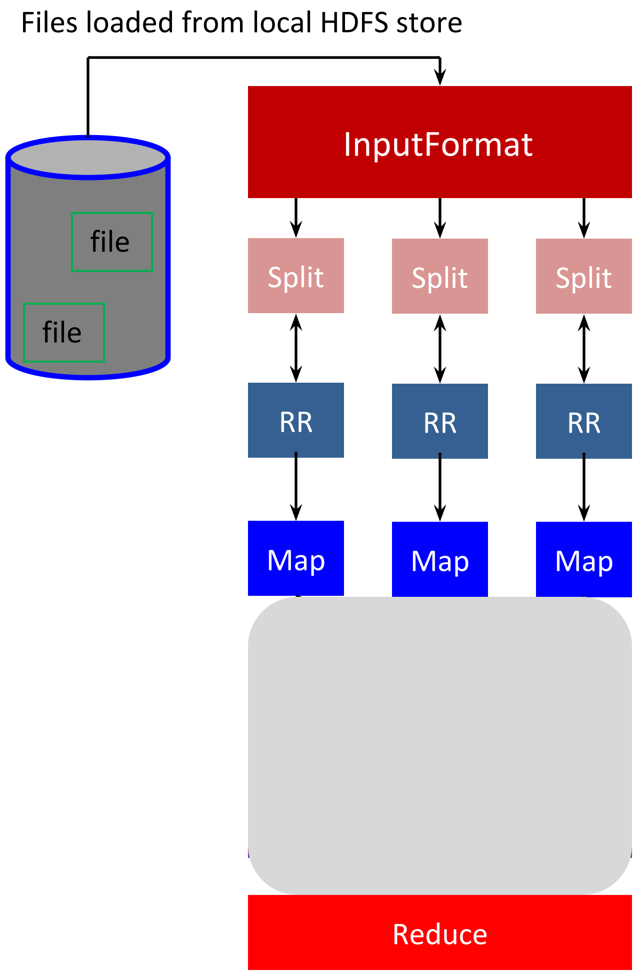
RecordReader

- The input split defines a slice of work but does not describe how to access it
- The *RecordReader* class actually loads data from its source and converts it into (K, V) pairs suitable for reading by Mappers
- The RecordReader is invoked repeatedly on the input until the entire split is consumed
- Each invocation of the RecordReader leads to another call of the map function defined by the programmer



Mapper and Reducer

- The *Mapper* performs the user-defined work of the first phase of the MapReduce program
- A new instance of Mapper is created for each split
- The *Reducer* performs the user-defined work of the second phase of the MapReduce program
- A new instance of Reducer is created for each partition
- *For each key in the partition assigned to a Reducer, the Reducer is called once*



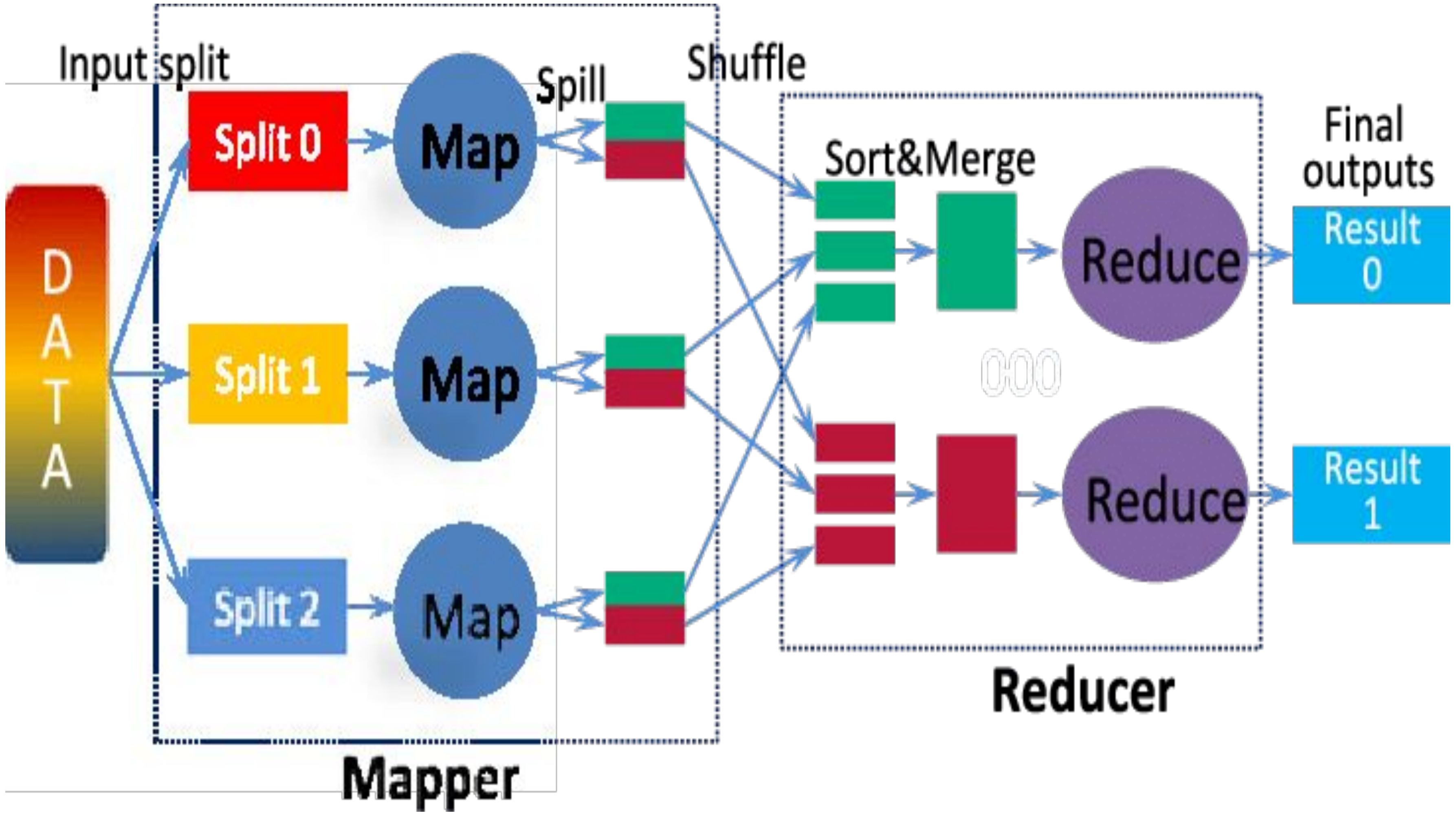
Combiner

- Combiner is an optional function but provides high performance in terms of network bandwidth and disk space.
- It takes intermediate key-value pair provided by mapper and applies user-specific aggregate function to only that mapper.
- It is also known as local reducer.

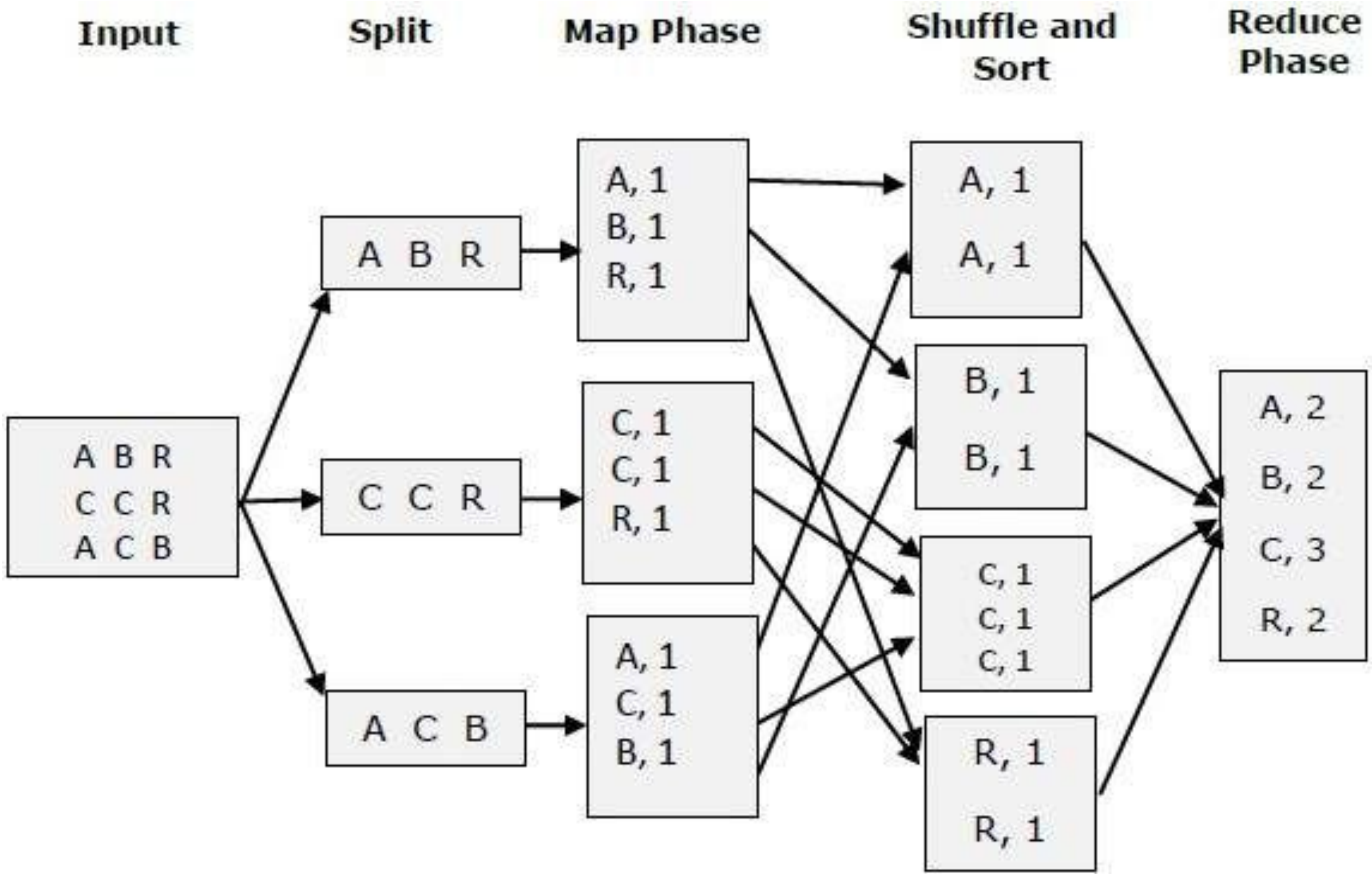
Combiner

- A Combiner, also known as a semi-reducer, is **an optional class that operates by accepting the inputs from the Map class and thereafter passing the output key-value pairs to the Reducer class**.
- The main function of a Combiner is to summarize the map output records with the same key.
- The combiner in MapReduce is also known as 'Mini-reducer'.
- The primary job of Combiner is **to process the output data from the Mapper, before passing it to Reducer**.
- It runs after the mapper and before the Reducer and its use is optional.

Overall Diagram

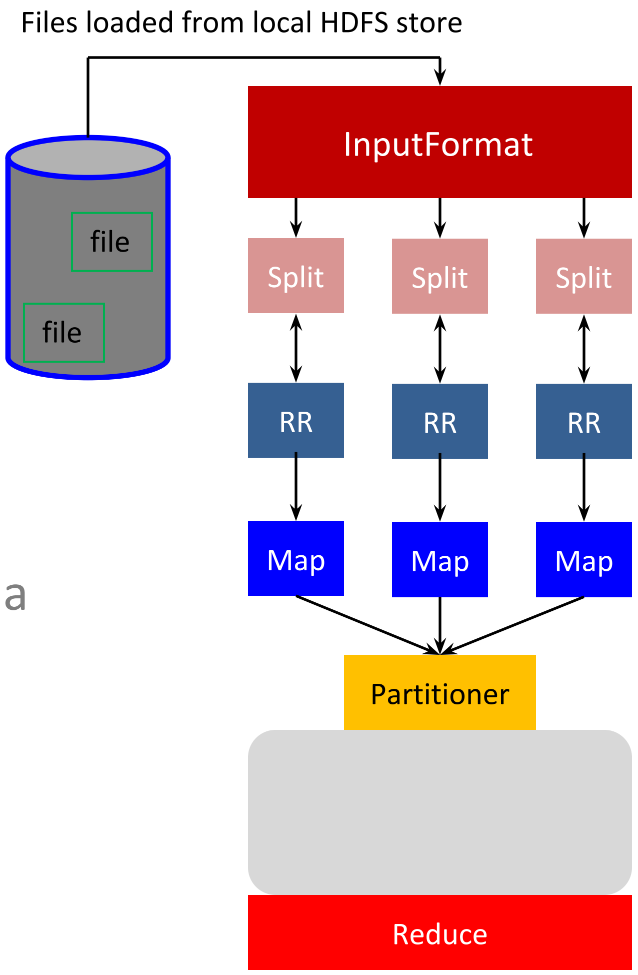


Combiner Task



Partitioner

- Each mapper may emit (K, V) pairs to *any* partition
- Therefore, the map nodes must all agree on where to send different pieces of intermediate data
- The *partitioner* class determines which partition a given (K,V) pair will go to
- The default partitioner computes *a hash value* for a given key and assigns it to a partition based on this result



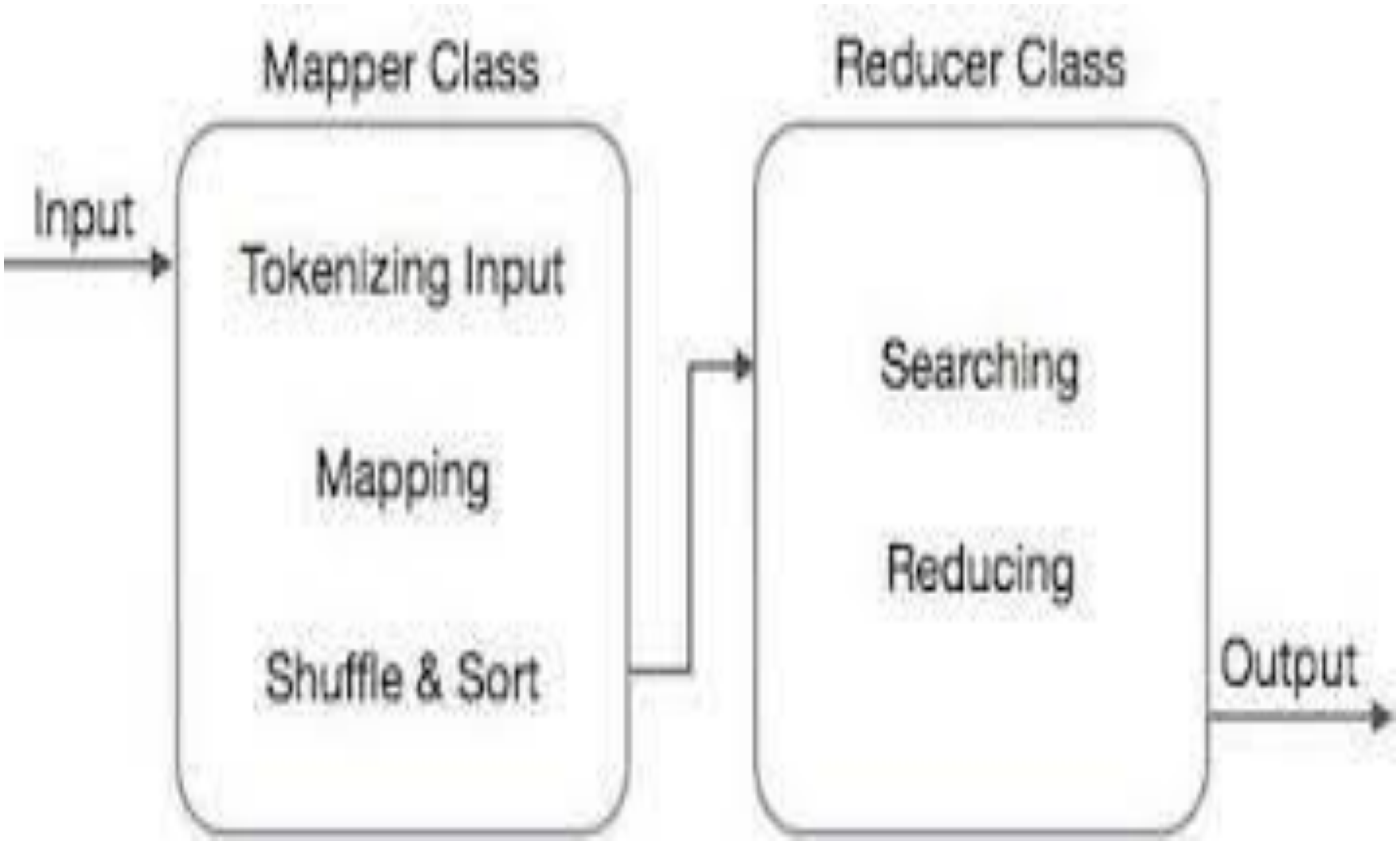
Reducer

- The output produced by map task is known as intermediate keys and values.
- These intermediate keys and values are sent to reducer.
- The reducer tasks are broken into the following phases :
- 1. Shuffle
- 2. Sort
- 3. Reducer
- 4. output Format

Searching

- Searching plays a key role in MapReduce algorithm.
- **It supports in the combiner phase and the Reducer phase**.
- The Map phase processes each input file and provides the data in key-value pairs (<k, v>).
- The combiner phase (searching technique) will accept the input from the Map phase as a key-value pair.

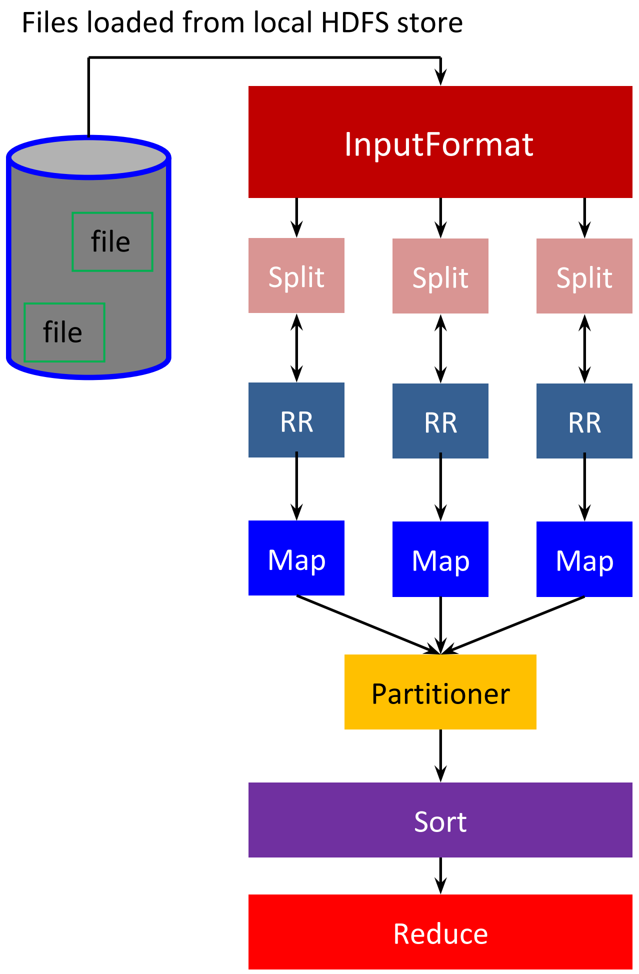
Searching

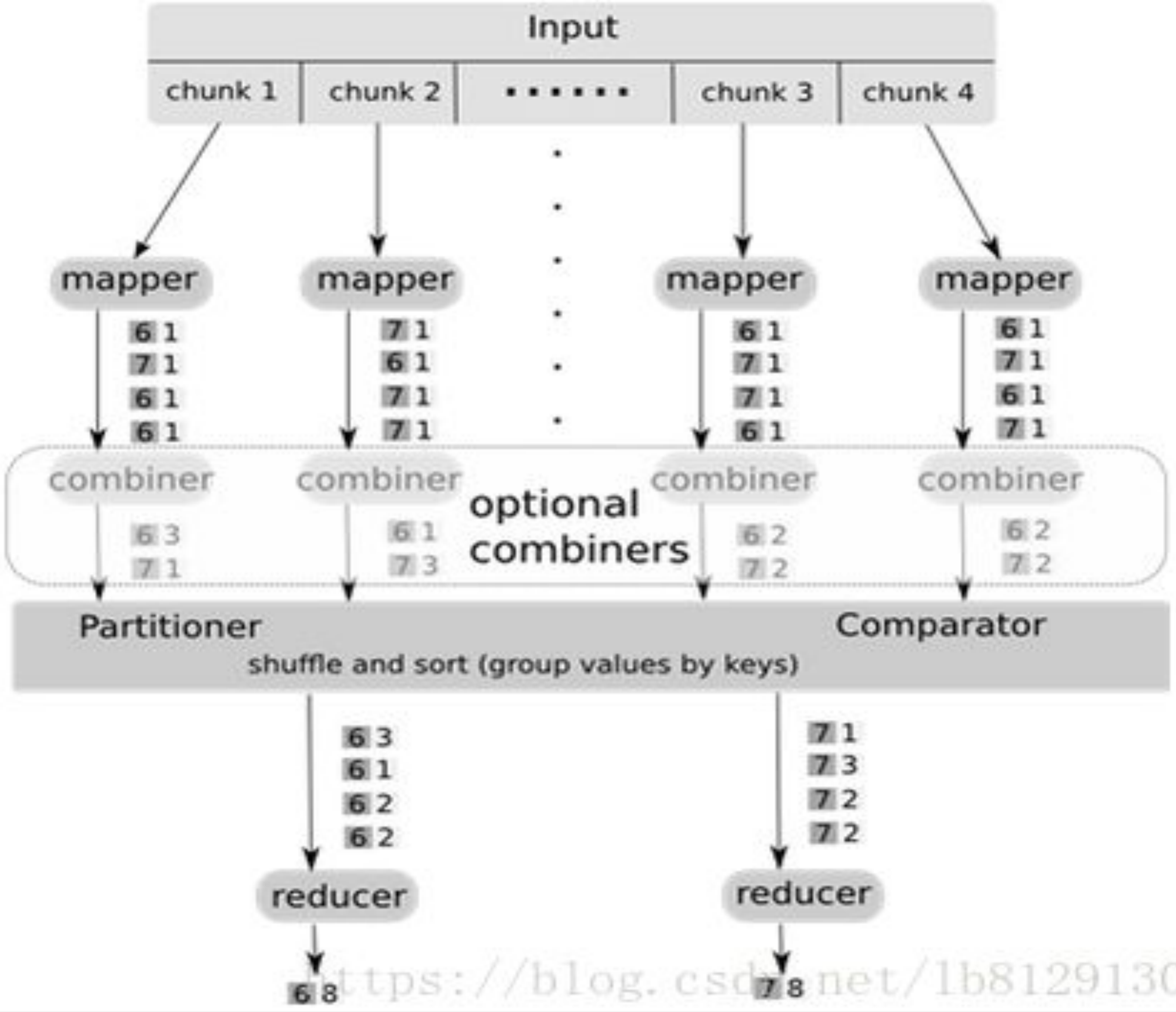

Mapper Class
Reducer Class
Input
Tokenizing Input
Mapping
Shuffle & Sort
Searching
Reducing
Output

Sort

- Each Reducer is responsible for reducing the values associated with (several) intermediate keys

- The set of intermediate keys on a single node is *automatically sorted* by MapReduce before they are presented to the Reducer




Input
chunk 1
chunk 2
chunk 3
chunk 4
mapper
6 1
7 1
6 1
6 1
7 1
6 1
7 1
7 1
6 1
7 1
7 1
6 1
6 1
7 1
6 1
7 1
combiner
optional combiners
6 3
7 1
6 1
7 3
6 2
7 2
6 2
7 2
Partitioner
Comparator
shuffle and sort (group values by keys)
6 3
6 1
6 2
6 2
7 1
7 3
7 2
7 2
reducer
6 8
7 8

Compression

- Compress the Mapreduce output files.
- Compression provide two benefits as,
- Reduce the space to store files.
- Speeds up data transfer across the network.
- **Input files are compressed, they will be automatically decompressed.**
- GripCodec is the compression algorithm for grip. This compresses the output file.