
hadoop

Hadoop MapReduce

- **What is MapReduce?**

- MapReduce is the processing layer of Hadoop. MapReduce programming model is designed for **processing large volumes of data in parallel** by dividing the work into a set of independent tasks. You need to put **business logic** in the way MapReduce works and rest things will be taken care by the framework. Work (complete job) which is submitted by the user to master is divided into small works (tasks) and assigned to slaves.
- MapReduce programs are written in a particular style influenced by functional programming constructs, specifical idioms for processing lists of data. In MapReduce, we get inputs from a list and it converts it into output which is again a list. It is the heart of Hadoop.

Hadoop MapReduce

- Hadoop is so much powerful and efficient due to MapRreduce as here **parallel processing** is done.
- Map-Reduce divides the work into small parts, each of which can be done in parallel on the cluster of servers. A problem is divided into a large number of smaller problems each of which is processed to give individual outputs. These individual outputs are further processed to give final output.
- Hadoop Map-Reduce is scalable and can also be used across many computers. Many small machines can be used to process jobs that could not be processed by a large machine.

Hadoop MapReduce

- **Apache MapReduce Terminologies**

➢ Map-Reduce is the data processing component of Hadoop. Map-Reduce programs transform **lists of input data elements** into **lists of output data elements**. A Map-Reduce program will do this twice, using two different list processing idioms-

  – Map
  – Reduce

➢ In between Map and Reduce, there is small phase called **Shuffle** and **Sort** in MapReduce.

Hadoop MapReduce

- **What is a MapReduce Job?**

➢ MapReduce Job or a “full program” is an **execution of a Mapper and Reducer** across a data set. It is an execution of 2 processing layers i.e mapper and reducer. A MapReduce job consists of the **input data, the MapReduce Program, and configuration info**.

- So client needs to submit input data, he needs to write Map Reduce program and set the configuration info (These were provided during Hadoop setup in the configuration file and also we specify some configurations in our program itself which will be specific to our map reduce job).

Hadoop MapReduce

- **What is Task in Map Reduce?**

➢ A task in MapReduce is an **execution of a Mapper or a Reducer** on a slice of data. It is also called Task-In-Progress (TIP). It means processing of data is in progress either on mapper or reducer.

- **What is Task Attempt?**

➢ Task Attempt is a particular instance of an attempt to execute a task on a node. There is a possibility that anytime any machine can go down. For example, while processing data if any node goes down, framework reschedules the task to some other node. This **rescheduling of the task cannot be infinite**. There is an **upper limit** for that as well. The default value of task attempt is **4**. If a task (Mapper or reducer) fails 4 times, then the **job is considered as a failed job**. For high priority job or huge job, the value of this task attempt can also be increased.

Hadoop MapReduce

- **Map Abstraction**

➢ The first phase of MapReduce paradigm, what is a map/mapper, what is the input to the mapper, how it processes the data, what is output from the mapper?

➢ The map takes key/value pair as input. Whether data is in structured or unstructured format, framework converts the incoming data into key and value.

  – Key is a reference to the input value.

  – Value is the data set on which to operate.

- **Map Processing**:

➢ A function defined by user – user can write custom business logic according to his need to process the data.

➢ Applies to every value in value input.

Hadoop MapReduce

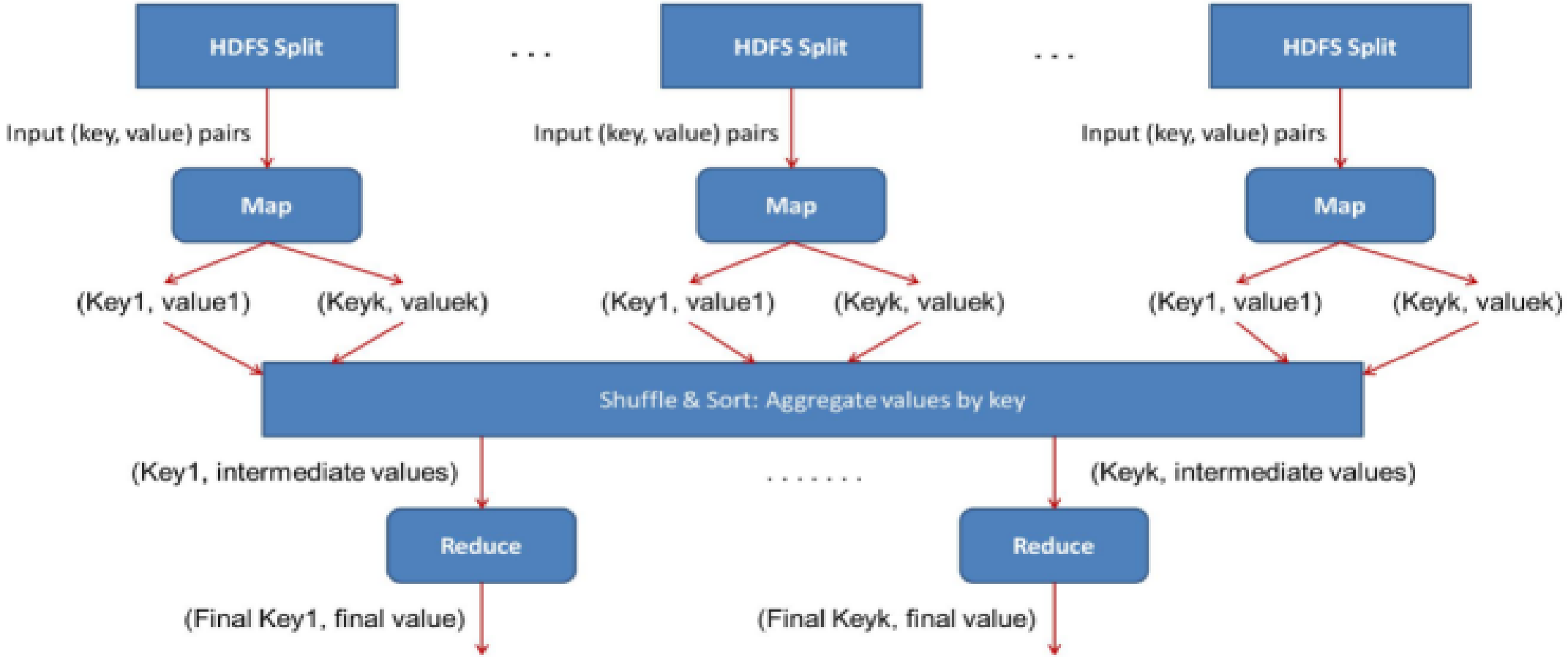
- Map produces a new list of key/value pairs:
- An output of Map is called intermediate output.
- Can be the different type from input pair.
- An output of map is stored on the **local disk** from where it is shuffled to reduce nodes.
- **Reduce Abstraction**
- The second phase of MapReduce – **Reducer** , what is the input to the reducer, what work reducer does, where reducer writes output?
- **Reduce** takes intermediate Key / Value pairs as input and processes the output of the mapper. Usually, in the reducer, we do aggregation or summation sort of computation.
  - Input given to reducer is generated by Map (intermediate output)
  - Key / Value pairs provided to reduce are sorted by key

Hadoop MapReduce

- **Reduce processing**:
  - A function defined by user – Here also user can write custom business logic and get the final output.
  - Iterator supplies the values for a given key to the Reduce function.
- Reduce produces a final list of key/value pairs:
  - An output of Reduce is called Final output.
  - It can be a different type from input pair.
  - An output of Reduce is stored in HDFS.

Hadoop MapReduce

- How Map and Reduce work Together?



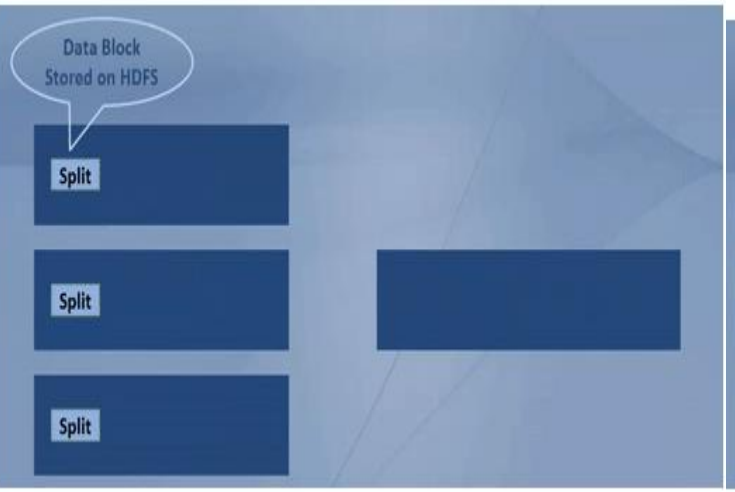
Hadoop MapReduce

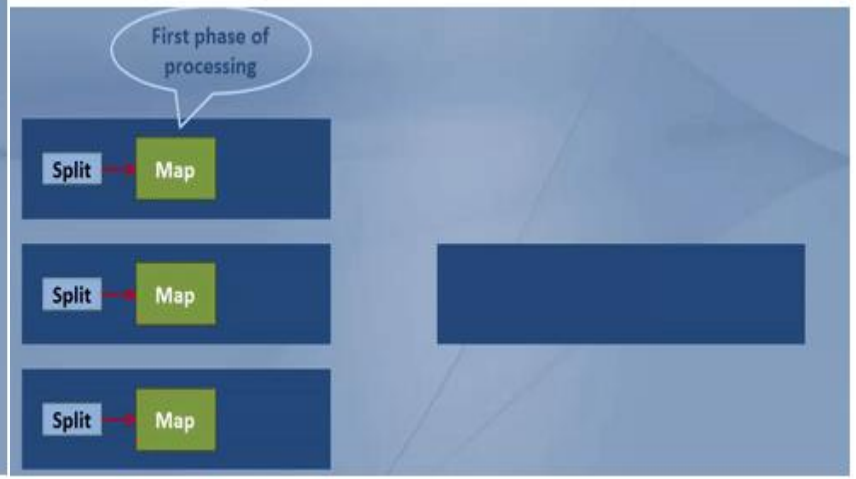
- How Map and Reduce work Together?

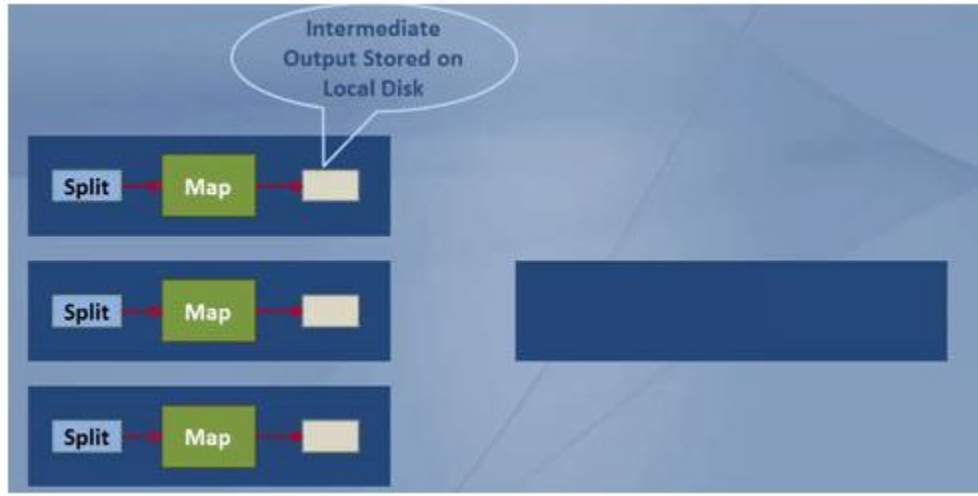
➢ **Input data** given to mapper is processed through **user defined function** written at **mapper**. All the required complex **business logic** is implemented at the mapper level so that heavy processing is done by the mapper in parallel as the number of mappers is much more than the number of reducers. Mapper generates an output which is intermediate data and this output goes as input to reducer.

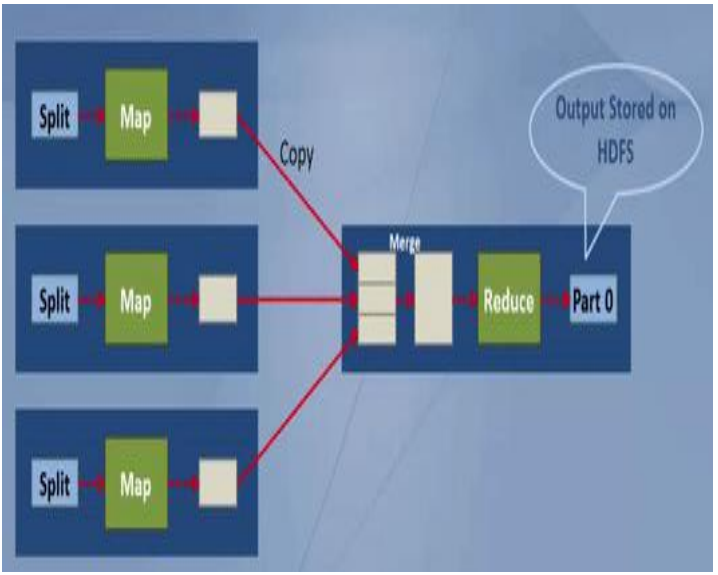
➢ This **intermediate result** is then processed by user defined function written at **reducer** and final output is generated. Usually, in reducer very light processing is done. This final output is stored in HDFS and replication is done as usual.

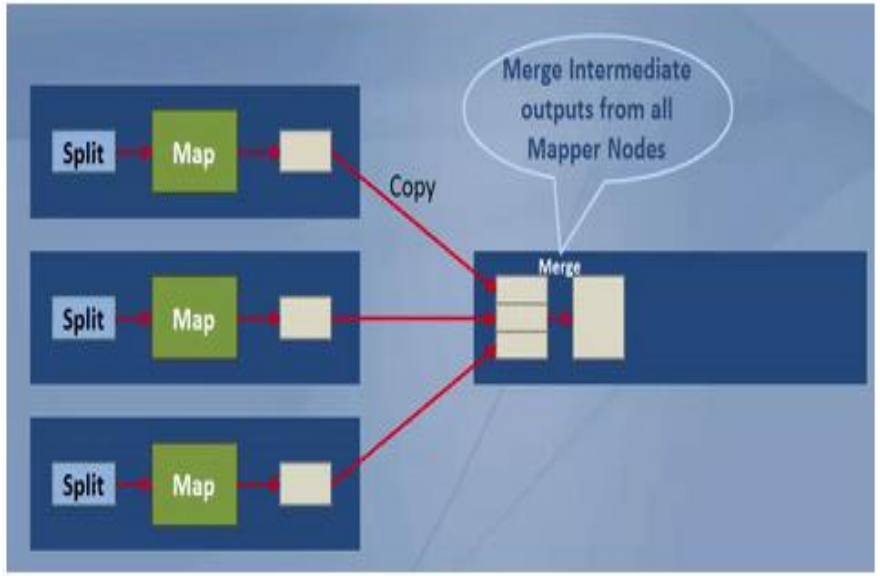
Hadoop MapReduce

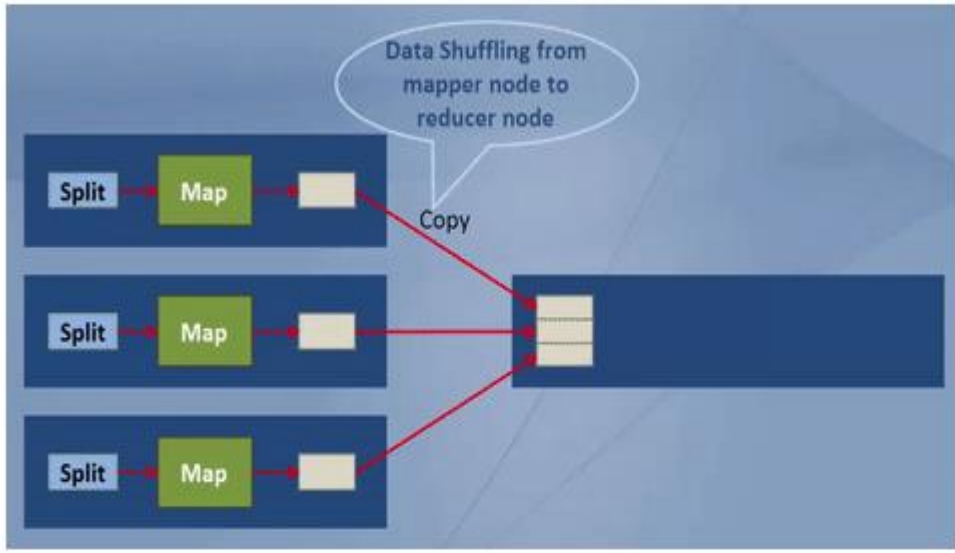












Hadoop MapReduce

- As seen from the diagram of mapreduce workflow in Hadoop, the **square block** is a **slave**. There are 3 slaves in the figure. On all 3 slaves mappers will run, and then a reducer will run on any 1 of the slave. For simplicity of the figure, the reducer is shown on a different machine but it will run on mapper node only.
- Let us now discuss the map phase: An input to a mapper is 1 block at a time. (Split = block by default)
- An **output of mapper** is written to a **local disk** of the machine on which **mapper** is running. Once the map finishes, this intermediate output travels to reducer nodes (node where reducer will run).
- **Reducer** is the second phase of processing where the user can again write his custom business logic. Hence, an output of reducer is the final **output written to HDFS**.

Hadoop MapReduce

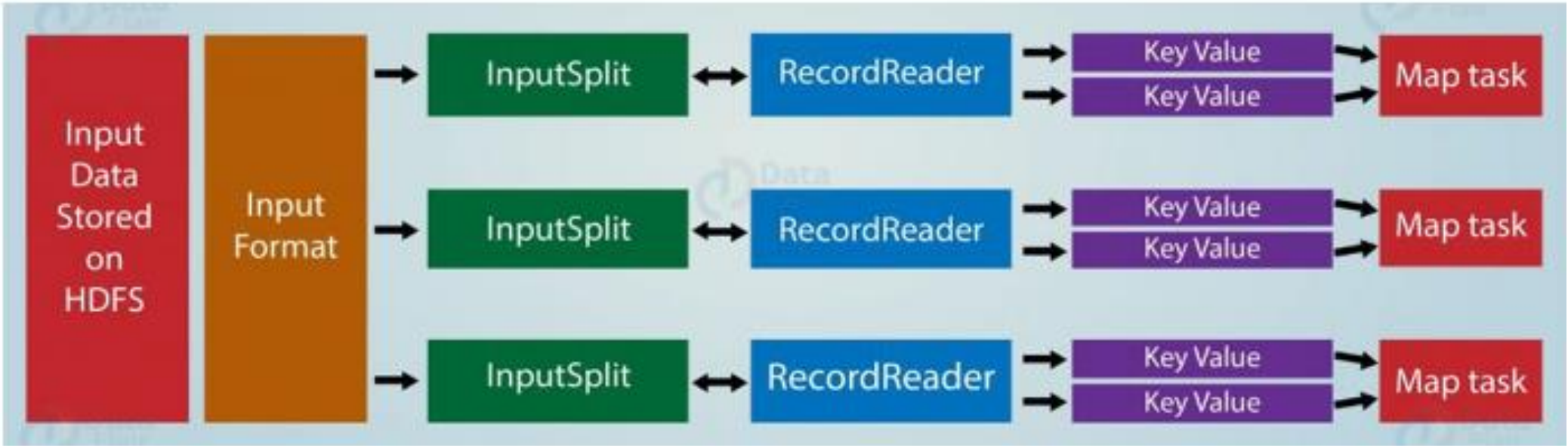
- By default on a slave, 2 mappers run at a time which can also be increased as per the requirements. It depends again on factors like data node hardware, block size, machine configuration etc. We should not increase the number of mappers beyond the certain limit because it will decrease the performance.
- **Mapper** in Hadoop Mapreduce writes the output to the local disk of the machine it is working. This is the temporary data. An output of mapper is also called intermediate output. All mappers are writing the output to the local disk. As First mapper finishes, data (output of the mapper) is traveling from mapper node to reducer node. Hence, this **movement of output from mapper node to reducer node** is called **shuffle**.

Hadoop MapReduce

- **Reducer** is also deployed on any one of the datanode only. An output from all the mappers goes to the reducer. All these **outputs** from different **mappers** are **merged** to form input for the reducer. This input is also on local disk. Reducer is another processor where you can write custom business logic. It is the second stage of the processing. Usually to reducer we write **aggregation, summation** etc. type of functionalities. Hence, Reducer gives the final output which it writes on HDFS.
- Map and reduce are the stages of processing. They run one after other. After all, mappers complete the processing, then only reducer starts processing.

Hadoop MapReduce

- Concept of Key-Value Pair in Hadoop MapReduce



- In MapReduce process, before passing the data to the mapper, data should be first converted into key-value pairs as mapper only understands key-value pairs of data.

Hadoop MapReduce

- key-value pairs in Hadoop MapReduce is generated as follows:

➢ **InputSplit** – It is the logical representation of data. The data to be processed by an individual Mapper is presented by the InputSplit.

➢ **RecordReader** – It communicates with the InputSplit and it converts the Split into records which are in form of key-value pairs that are suitable for reading by the mapper. By default, RecordReader uses TextInputFormat for converting data into a key-value pair. RecordReader communicates with the InputSplit until the file reading is not completed.

Hadoop MapReduce

- In MapReduce, map function processes a certain key-value pair and emits a certain number of key-value pairs and the Reduce function processes values grouped by the same key and emits another set of key-value pairs as output. The output types of the Map should match the input types of the Reduce as shown below:
- Map: (K1, V1) -> list (K2, V2)
- Reduce: {(K2, list (V2 }) -> list (K3, V3)

Hadoop MapReduce

- **On what basis is a key-value pair generated in Hadoop?**

➢ Generation of a key-value pair in Hadoop depends on the data set and the required output. In general, the key-value pair is specified in 4 places: Map input, Map output, Reduce input and Reduce output.

1. Map Input

➢ Map-input by default will take the line offset as the key and the content of the line will be the value as Text. By using custom InputFormat we can modify them.

2. Map Output

➢ Map basic responsibility is to filter the data and provide the environment for grouping of data based on the key.

➢ Key – It will be the field/ text/ object on which the data has to be grouped and aggregated on the reducer side.

➢ Value – It will be the field/ text/ object which is to be handled by each individual reduce method.

Hadoop MapReduce

3. **Reduce Input**

- The output of Map is the input for reduce, so it is same as Map-Output.

4. **Reduce Output**

- It depends on the required output.

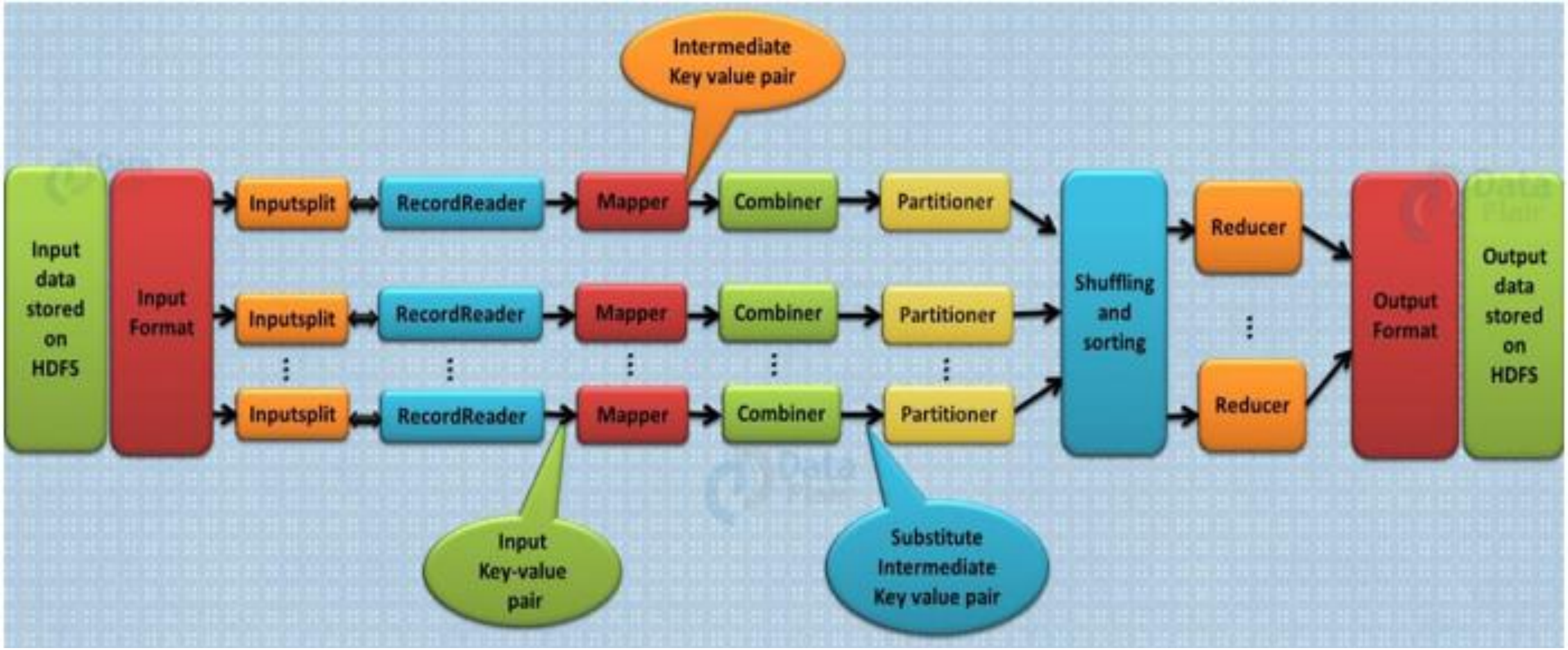
- **MapReduce key-value pair Example**

- Suppose, the content of the file which is stored in HDFS is **John is Mark Joey is John**. Using InputFormat, we will define how this file will split and read. By default, RecordReader uses TextInputFormat to convert this file into a key-value pair.
- **Key –** It is offset of the beginning of the line within the file.
- **Value –** It is the content of the line, excluding line terminators.

- From the above content of the file-

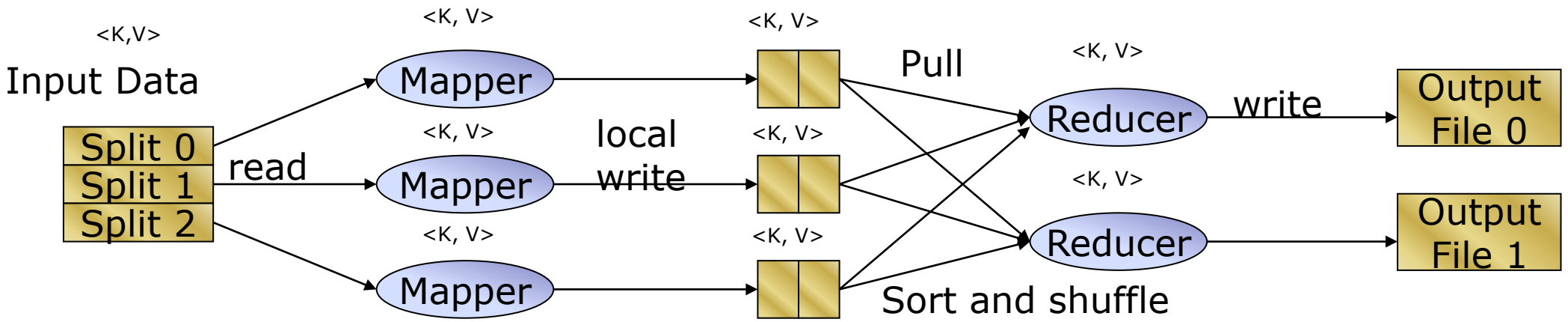
- **Key** is 0
- **Value** is John is Mark Joey is John.

Hadoop MapReduce

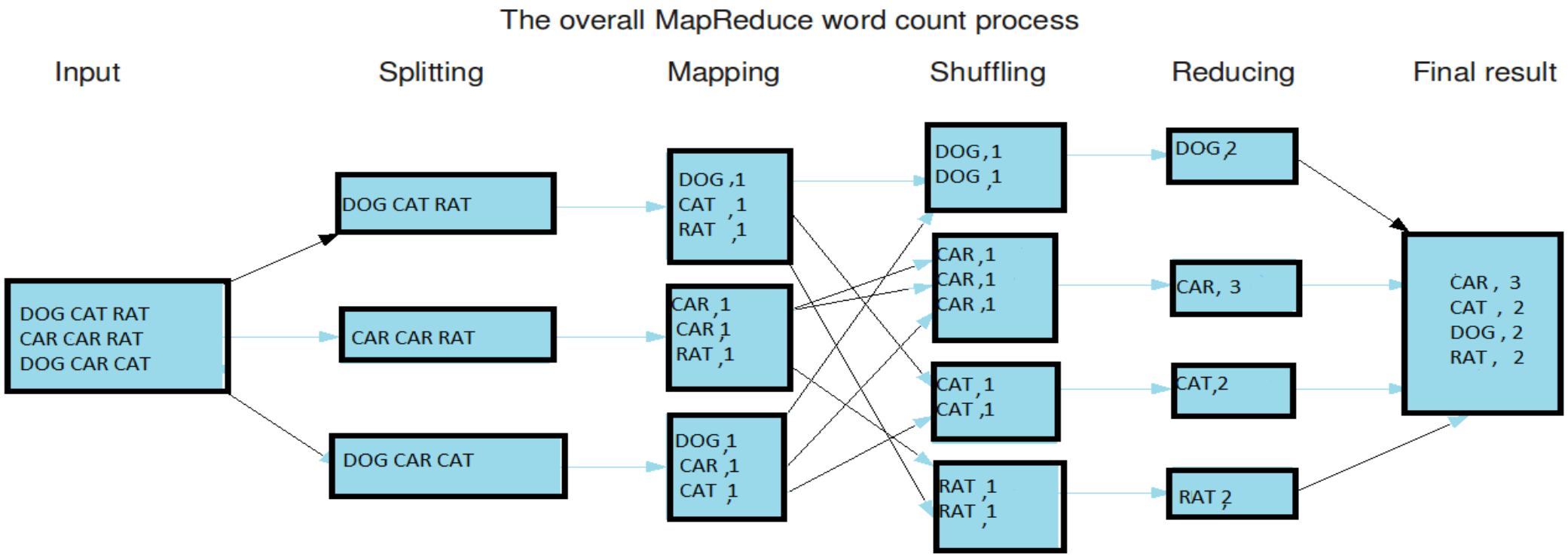


MapReduce

- MapReduce



WordCount


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
DOG CAT RAT
CAR CAR RAT
DOG CAR CAT
DOG CAT RAT
CAR CAR RAT
DOG CAR CAT
DOG ,1
CAT ,1
RAT ,1
CAR ,1
CAR ,1
RAT ,1
DOG ,1
CAR ,1
CAT ,1
DOG ,1
DOG ,1
CAR ,1
CAR ,1
CAR ,1
CAT ,1
CAT ,1
RAT ,1
RAT ,1
DOG ,2
CAR, 3
CAT,2
RAT ,2
CAR , 3
CAT , 2
DOG , 2
RAT , 2

Thank you.