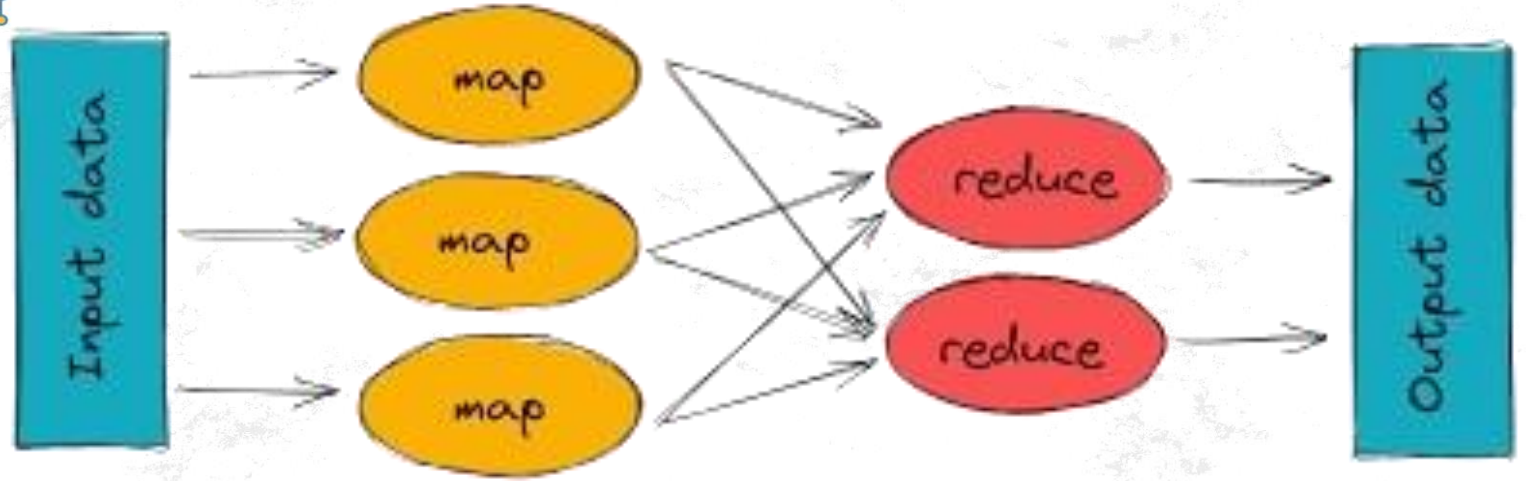


Introduction to **MAPREDUCE**

NIELIT Chandigarh/Ropar

"Big Data is at the foundation of all the megatrends that are happening today."

MapReduce - What?

- **MapReduce** is a programming model used for processing large amounts of data in parallel across many computers, or nodes, in a distributed system like Hadoop. It helps break down complex data processing tasks into smaller, more manageable parts and makes the whole process faster and more efficient.

Here's a breakdown of how **MapReduce** works:

- **Map Phase:**

Goal: The "Map" function breaks down the input data into smaller chunks and processes each chunk independently.

For example, imagine you have a huge list of words, and you want to count how often each word appears.

The **Map** step would go through the list and create pairs of each word and the number 1 (e.g., ("apple", 1), ("banana", 1), etc.).

The output of the Map phase is a list of key-value pairs.

MapReduce - What?

- **Shuffle and Sort Phase (Automatic):**

After the Map phase, all the key-value pairs are sorted by the key (in this case, the word) and grouped together.

For example, all occurrences of "apple" would be grouped together, all occurrences of "banana" together, and so on.

This step is handled automatically by the MapReduce system.

- **Reduce Phase:**

Goal: The "Reduce" function takes these grouped key-value pairs and processes them further.

For example, it sums up the numbers for each word, so for "apple", it would add all the 1s to get the total count of "apple".

The Reduce phase outputs the final result, which in this case would be something like ("apple", 10) if the word "apple" appeared 10 times.

Why is it Efficient?

- **Parallel Processing**: Instead of processing data in one big chunk, MapReduce splits the task into smaller pieces (Map), processes them in parallel across many computers, and then combines them (Reduce).
- **Scalability**: As the amount of data grows, you can simply add more computers to the system to handle the increased load, making MapReduce scalable.
- MapReduce allows you to:

1. **Map**: Break large data into smaller chunks, process them in parallel, and create intermediate key-value pairs.
2. **Shuffle and Sort**: Automatically group and organize these pairs based on keys.
3. **Reduce**: Aggregate or process the grouped data to produce the final output.

- *MapReduce is widely used in big data processing systems (like Hadoop) because it efficiently handles huge datasets that wouldn't fit on a single machine. It's a powerful way to process data in parallel and scale across many computers!*

MapReduce - What?

- MapReduce is a programming model for efficient distributed computing
- It works like a Unix pipeline
  - cat input | grep | sort | uniq -c | cat > output
  - **Input | Map |** Shuffle & Sort **| Reduce | Output**
- Efficiency from
  - Streaming through data, reducing seeks
  - Pipelining
- A good fit for a lot of applications
  - Log processing
  - Web index building

MapReduce

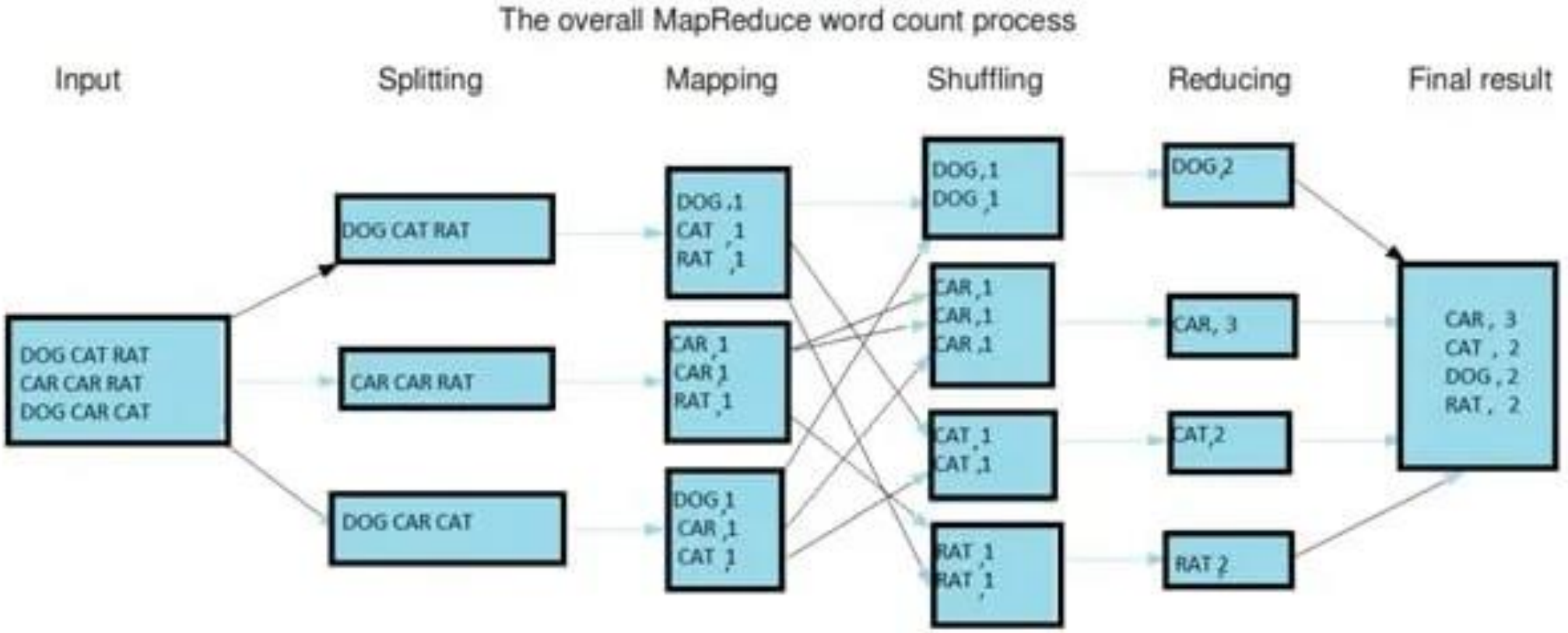
- MapReduce is one of the core components of the Hadoop framework and plays an important role in processing huge amounts of structured and unstructured data. The issues associated with traditional ways of parallel and distributed processing have laid roots for the development of the MapReduce framework. In this session we are going to cover the below topics:
- What is MapReduce?
- Different stages available in MapReduce
- MapReduce architecture & its working style
- Core components of Hadoop architecture
- Role of Job Tracker and Task Tracker in MapReduce

What is MapReduce?

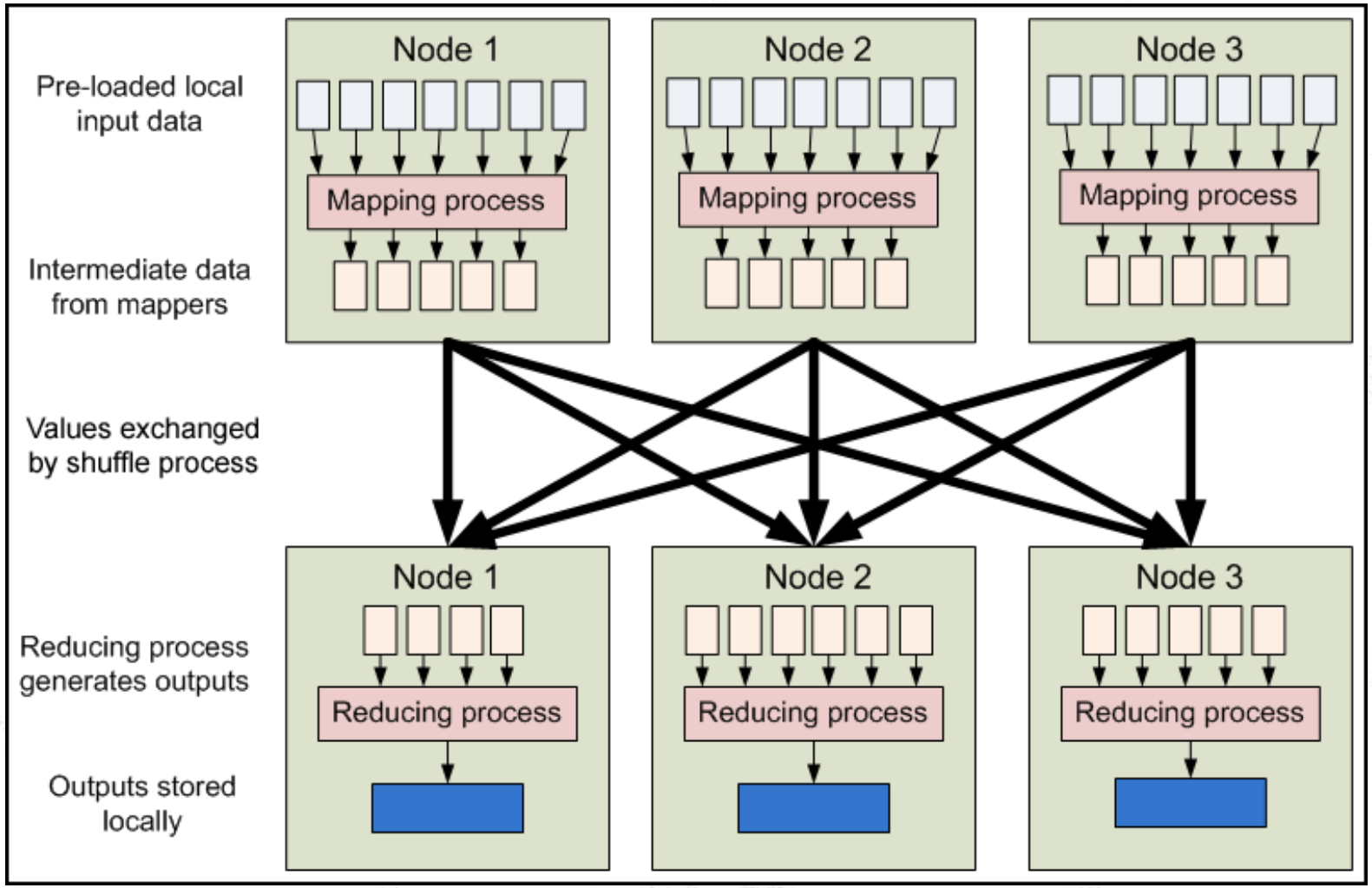
MapReduce is a programming model that simplifies the task of data processing by allowing users to perform parallel and distributed processing on huge volumes of data. As the name depicts it consists of two important tasks which are Map and Reduce. The Reducer task begins once the mapper task is over. The Map task job is to read and process a certain amount of data and produce key-value pairs. The output of Map job (key values) becomes the input for the Reducer task. The Reducer task collects and aggregates the key-value pairs and gives the final output. Hadoop framework is highly flexible and supports the MapReduce programs written in multiple languages which includes Python, Java, Ruby, and C++. The Parallel nature of Mapreduce framework streamlines the procedure to perform analysis on large-scale data analysis by using multiple machines in a cluster.

Stages in MapReduce MapReduce performs two functions namely Map and Reduce. In order to accomplish these tasks, it goes through four stages which are **splitting, mapping, shuffling, and reducing**.

MapReduce


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

MapReduce - Dataflow



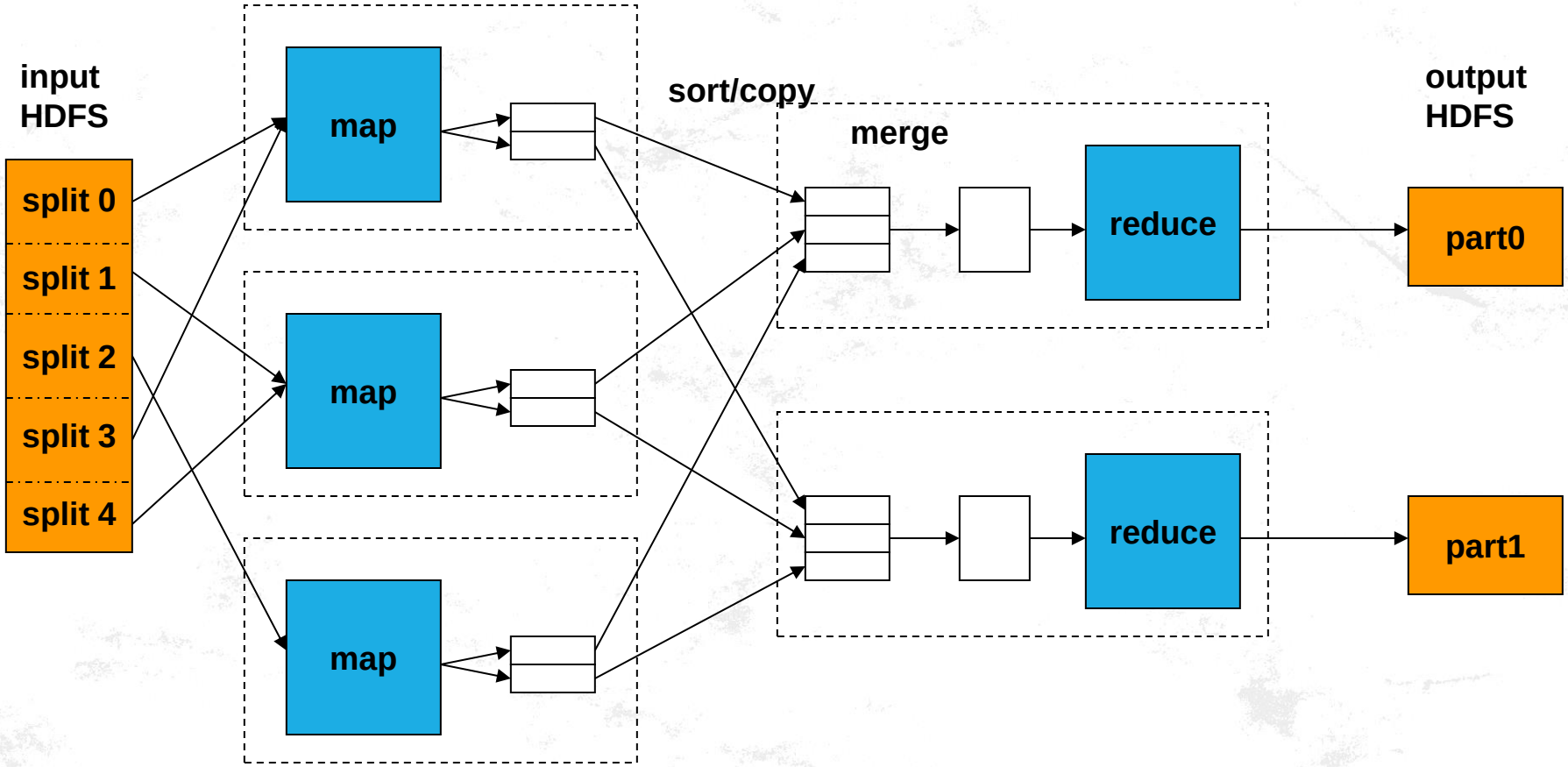
MapReduce - Features

- Fine grained Map and Reduce tasks
  - Improved load balancing
  - Faster recovery from failed tasks
- Automatic re-execution on failure
  - In a large cluster, some nodes are always slow or flaky
  - Framework re-executes failed tasks
- Locality optimizations
  - With large data, bandwidth to data is a problem
  - Map-Reduce + HDFS is a very effective solution
  - Map-Reduce queries HDFS for locations of input data
  - Map tasks are scheduled close to the inputs when possible

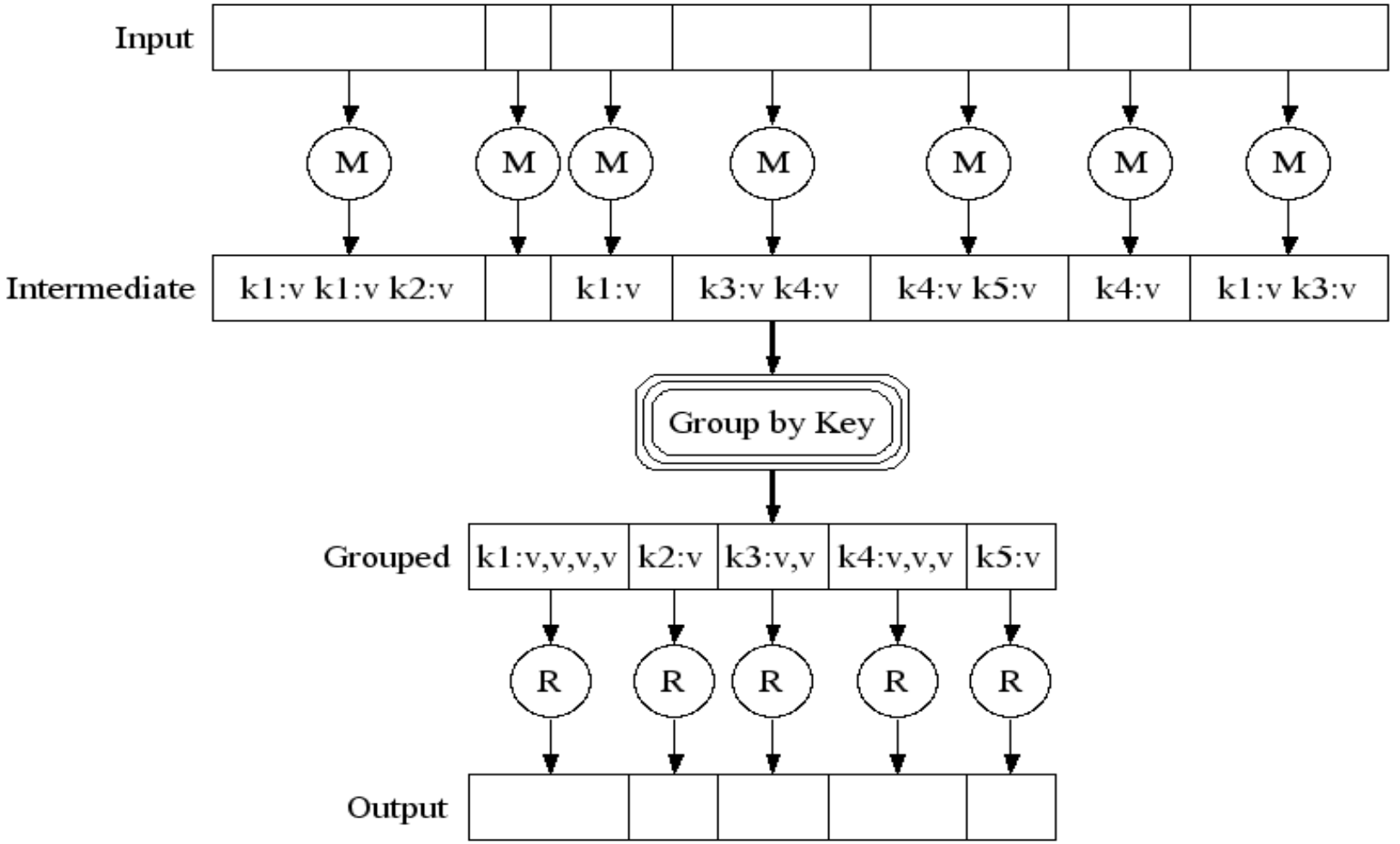
Word Count Example

- Mapper
  - Input: value: lines of text of input
  - Output: key: word, value: 1
- Reducer
  - Input: key: word, value: set of counts
  - Output: key: word, value: sum
- Launching program
  - Defines this job
  - Submits job to cluster

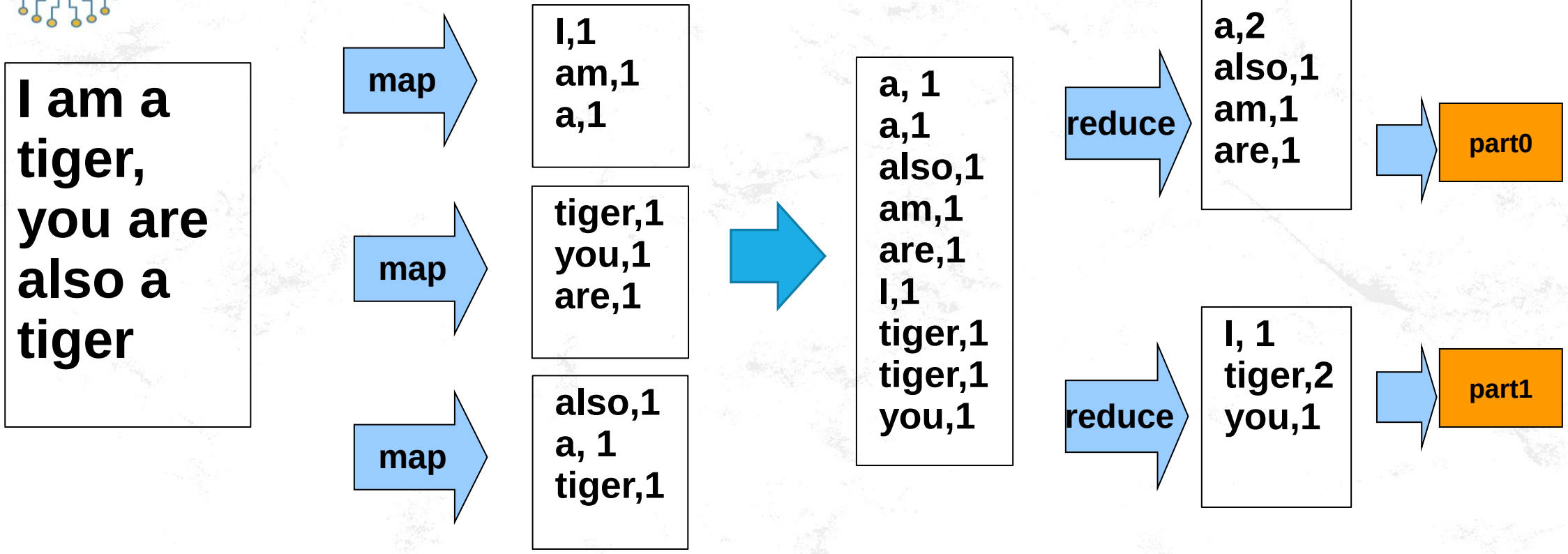
Hadoop-MapReduce Workflow



MapReduce Dataflow



Example



JobTracker generates three TaskTrackers for map tasks

Hadoop sorts the intermediate data

JobTracker generates two TaskTrackers for map tasks

MapReduce

Let us understand each stage in detail:

1) Input Splits An input to the MapReduce is divided into smaller units called input Splits. An input Split is a part of the input and consumed by a single Map.

2) Mapping Mapping is the first phase in the MapReduce algorithm. At this stage data in each split is transferred to a mapping function to produce output values. In the above diagram, the Map function segregates each word from input splits.

3)Shuffling The Shuffling stage takes the output from the map Stage and combines the relevant records. In the above diagram, you can clearly see that all the related items are grouped together with their respective frequencies.

4) Reducing This is the final stage in the MapReduce algorithm. At the Reduce stage, it takes the output from the Shuffle stage and combines the values in order to produce the final output value. In the above diagram, the Reducer task gives the number of a specific item and produces the final result.

MapReduce Framework

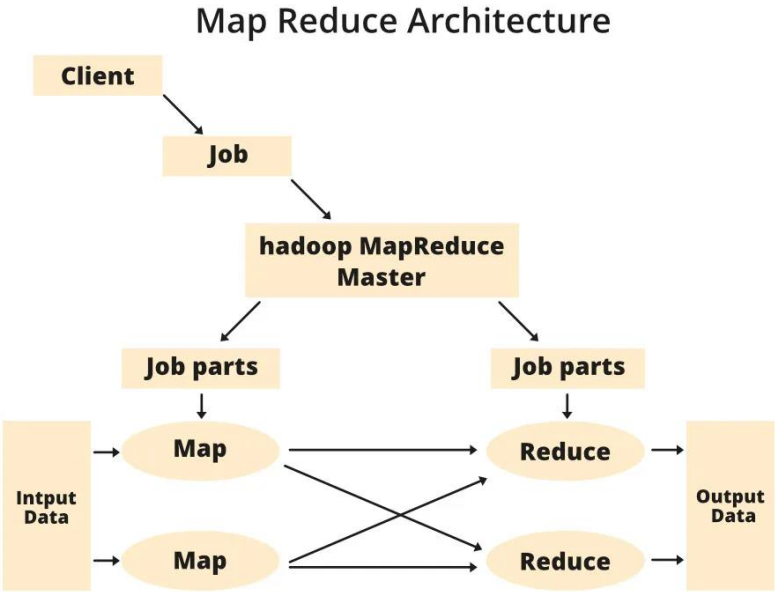
MapReduce is a powerful programming model within Hadoop designed to process and analyze large-scale data in a distributed and parallel manner. By dividing a job into smaller tasks, MapReduce ensures efficiency and scalability while reducing space and time complexities.

How MapReduce Works:

- **Job Submission:** A Client submits a job of a specific size to the MapReduce Master. The job is divided into smaller, manageable job-parts.
- **Task Execution:** Each job-part is processed using Map and Reduce tasks. Developers write logical programs for these tasks to solve specific problems or generate the desired output.
- **Data Flow:** Input Data is consumed by the Map task, which produces key-value pairs as intermediate output. These key-value pairs serve as the input for the Reducer, which processes them and generates the final output stored in the Hadoop Distributed File System (HDFS).
- **Optimized Design:** The architecture is designed to process multiple Map and Reduce tasks efficiently. The algorithms minimize both space and time complexities, ensuring optimal performance.

MapReduce Components

Client: The source of jobs, responsible for assigning work to the MapReduce framework.
Multiple clients can submit jobs simultaneously.
Job: Represents the actual work that needs to be accomplished, consisting of many small tasks.
Hadoop MapReduce Master: Divides the main job into smaller job-parts for parallel processing.
Job Parts: The smaller tasks derived from the main job; their results are combined to produce the final output.
Input Data: The raw data provided to MapReduce for processing.
Output Data: The processed and finalized data stored in HDFS.



Role of Job Tracker and Task Tracker in MapReduce

- **Job Tracker:**
  - Manages resources and jobs across the cluster.
  - Divides the job into multiple tasks and schedules them on different data nodes.
  - Monitors the execution and ensures task completion.
- **Task Tracker:**
  - Executes Map and Reduce tasks as per the instructions from the Job Tracker.
  - Deployed on every node in the cluster to handle local task execution.

Conclusion : MapReduce revolutionizes Big Data processing by enabling the efficient handling of massive unstructured datasets. Its ability to process data in parallel across distributed nodes makes it indispensable in the world of Big Data.

Execute the Word Count program