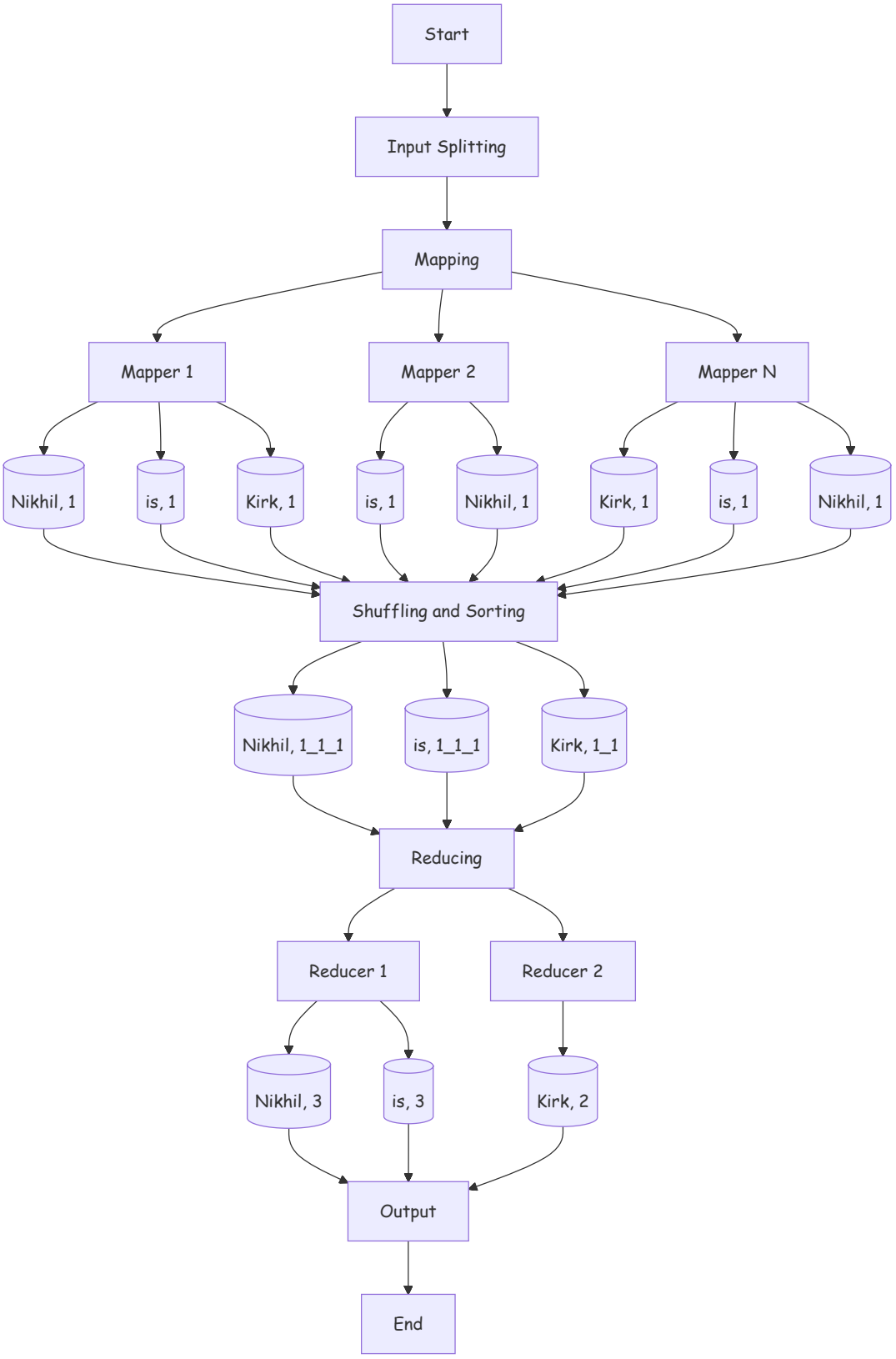
04 Map Reduce

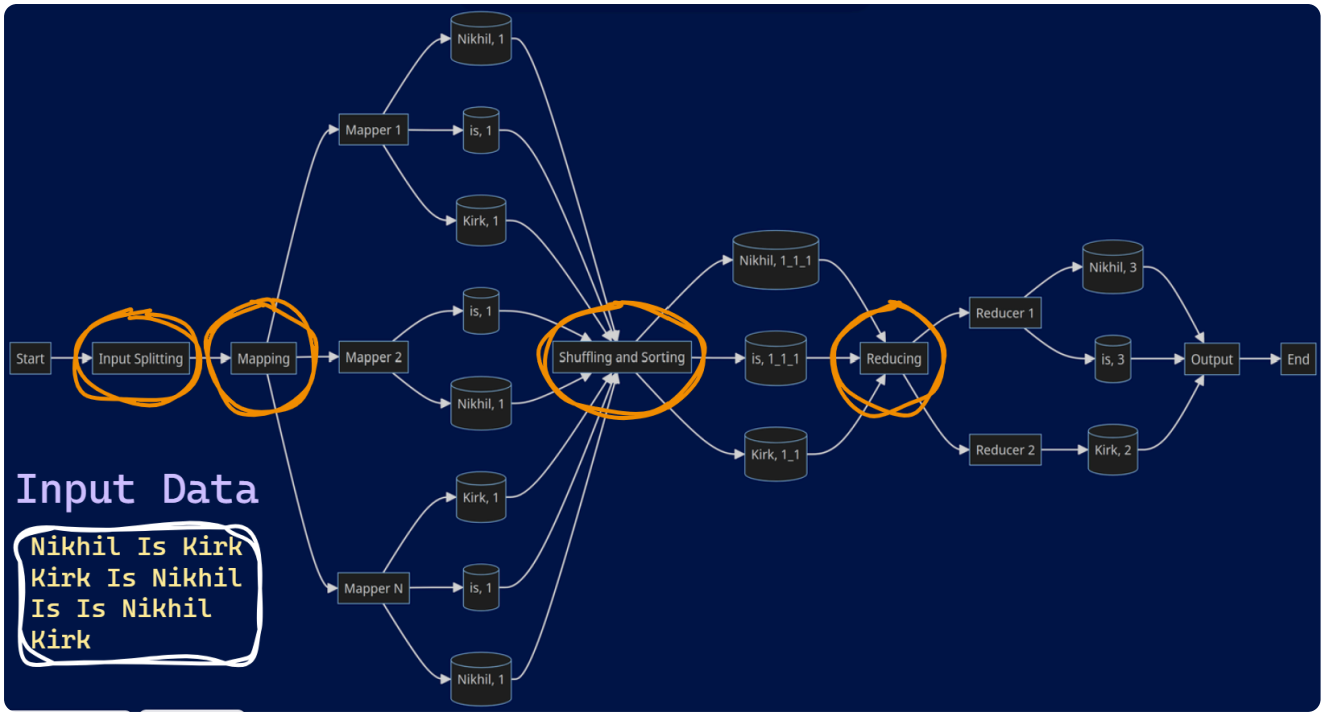
What is MapReduce?

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.

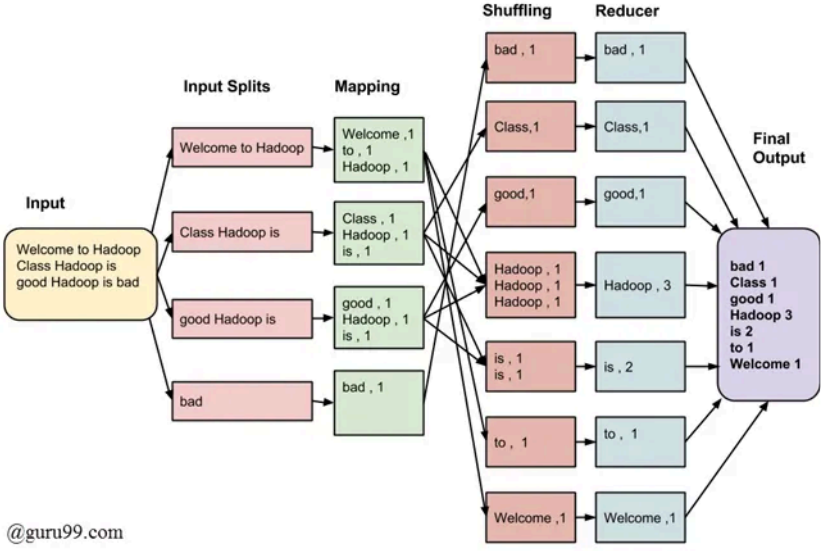
- The Map task job is to read and process a certain amount of data and produce key-value pairs.
- The output of Map job (key values) becomes the input for the Reducer task.
- The Reducer task collects and aggregates the key-value pairs and gives the final output.

Instead of moving data to the processing unit, we are moving the processing unit to the data in the MapReduce Framework.


Start
Input Splitting
Mapping
Mapper 1
Mapper 2
Mapper N
Nikhil, 1
is, 1
Kirk, 1
is, 1
Nikhil, 1
Kirk, 1
is, 1
Nikhil, 1
Shuffling and Sorting
Nikhil, 1_1_1
is, 1_1_1
Kirk, 1_1
Reducing
Reducer 1
Reducer 2
Nikhil, 3
is, 3
Kirk, 2
Output
End



Mapper is responsible for splitting and mapping the data while the reducer for shuffling and reducing the data.



1. Splitting:

Large datasets is divided into smaller chunks, `input splits`.
Each split can be processed independently, this helps in achieving parallel processing as compute nodes work simultaneously on each input.

2. Mapping:

Each split is subjected to `UD_Mapper`, transforms the input data to key-value pairs.
Examples:

1. **Tokenization**: For text analysis, each input record (e.g., text document) is split into tokens (words, phrases, etc.), and each token is emitted as a key-value pair with the token as the key and a count (usually 1) as the value.
2. **Filtering**: only certain records that meet specific criteria are emitted as key-value pairs, while others are discarded.
3. **Parsing**: Parsing map operations involve parsing input data into structured formats (e.g., JSON, XML) and extracting relevant fields or attributes. Each parsed record is emitted as a key-value pair with the extracted information.

3. Shuffling:

All values with same same key should go to the same node. It involves sorting, partitioning, and moving data between nodes, incurring network and I/O overhead.
It is a resource intensive step.
Performance of reducing phase is greatly enhanced by the shuffling phase.

4. Reducing:

In the final *"reducing"* phase, data with the same key is grouped together, and a UD-Reduce function is applied to these key-value pairs. This reduce function is responsible for aggregating, filtering etc. It produces the final output.

Advantages of MapReduce

1. Parallel Processing

In MapReduce, we are dividing the job among multiple nodes and each node works with a part of the job simultaneously. So, MapReduce is based on Divide and Conquer paradigm which helps us to process the data using different machines. As the data is processed by multiple machines instead of a single machine in parallel, the time taken to process the data gets reduced by a tremendous amount.

2. Data Locality

Instead of moving data to the processing unit, we are moving the processing unit to the data in the MapReduce Framework.

In the traditional system, we used to bring data to the processing unit and process it. But, as the data grew and became very huge, bringing this huge amount of data to the processing unit posed the following issues:

- Moving huge data to processing is costly and deteriorates the network performance.
- Processing takes time as the data is processed by a single unit which becomes the bottleneck.
- Master node can get over-burdened and may fail.

Now, MapReduce allows us to overcome the above issues by bringing the processing unit to the data. So, as you can see in the above image that the data is distributed among multiple nodes where each node processes the part of the data residing on it. This allows us to have the following advantages:

- It is very cost effective to move the processing unit to the data.
- The processing time is reduced as all the nodes are working with their part of the data in parallel.
- Every node gets a part of the data to process and therefore, there is no chance of a node getting overburdened.

```
year_max_temp_dict = {}

for line in open('/tmp/weather.csv', 'r').readlines():
  full_date, zip, temperature = line.split(",")
  temperature = int(temperature)
  year, month, day = [int(field) for field in full_date.split("-")]
  if year not in year_max_temp_dict:
    curr_max_temp = temperature
  else:
    existing_temp = year_max_temp_dict[year]
    curr_max_temp = max(temperature, existing_temp)
  year_max_temp_dict[year] = curr_max_temp

for year in year_max_temp_dict:
  print("Maximum temperature for the year {} is {}".format(year, year_max_temp_dict[year]))
```

Extra Readings:

1. https://medium.com/@abhisheksukhwal9/hadoop-map-reduce-in-detail-ac1c8429b03b
2. https://medium.com/edureka/mapreduce-tutorial-3d9535ddbe7c