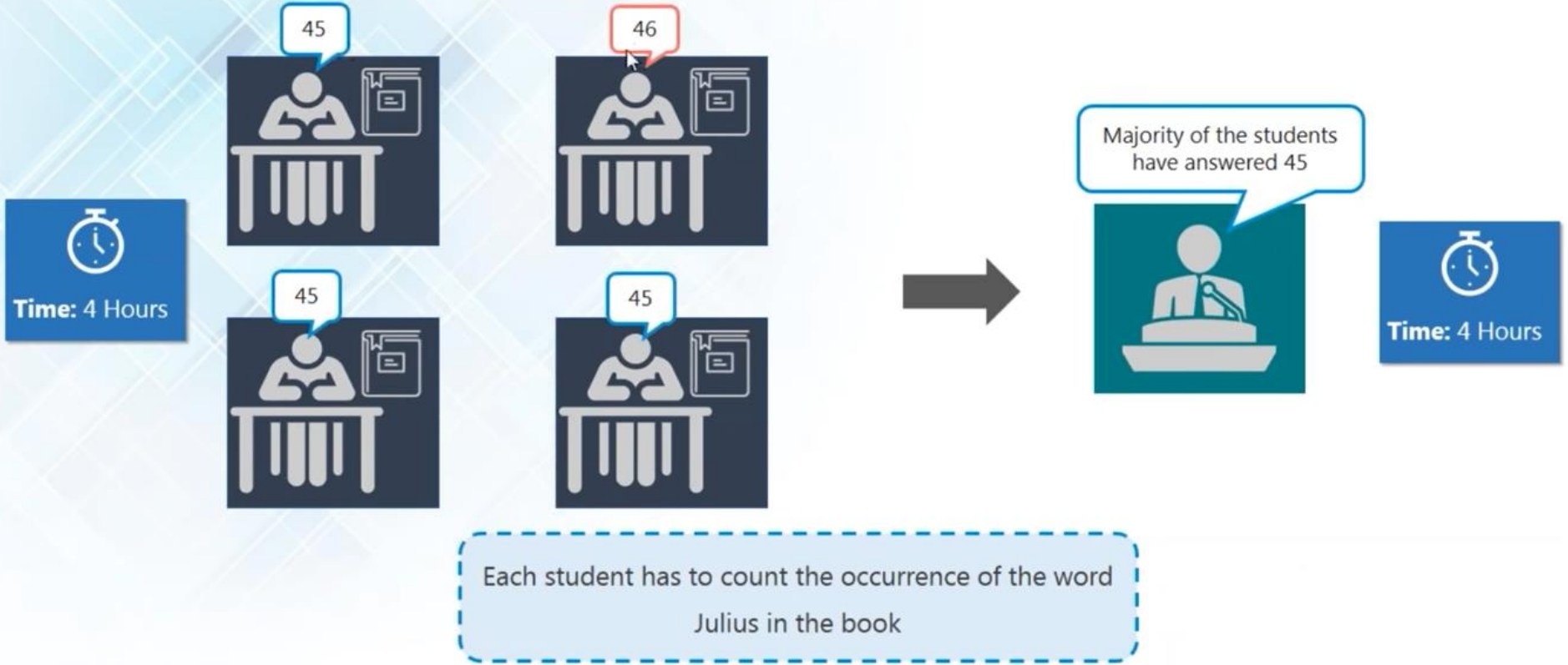
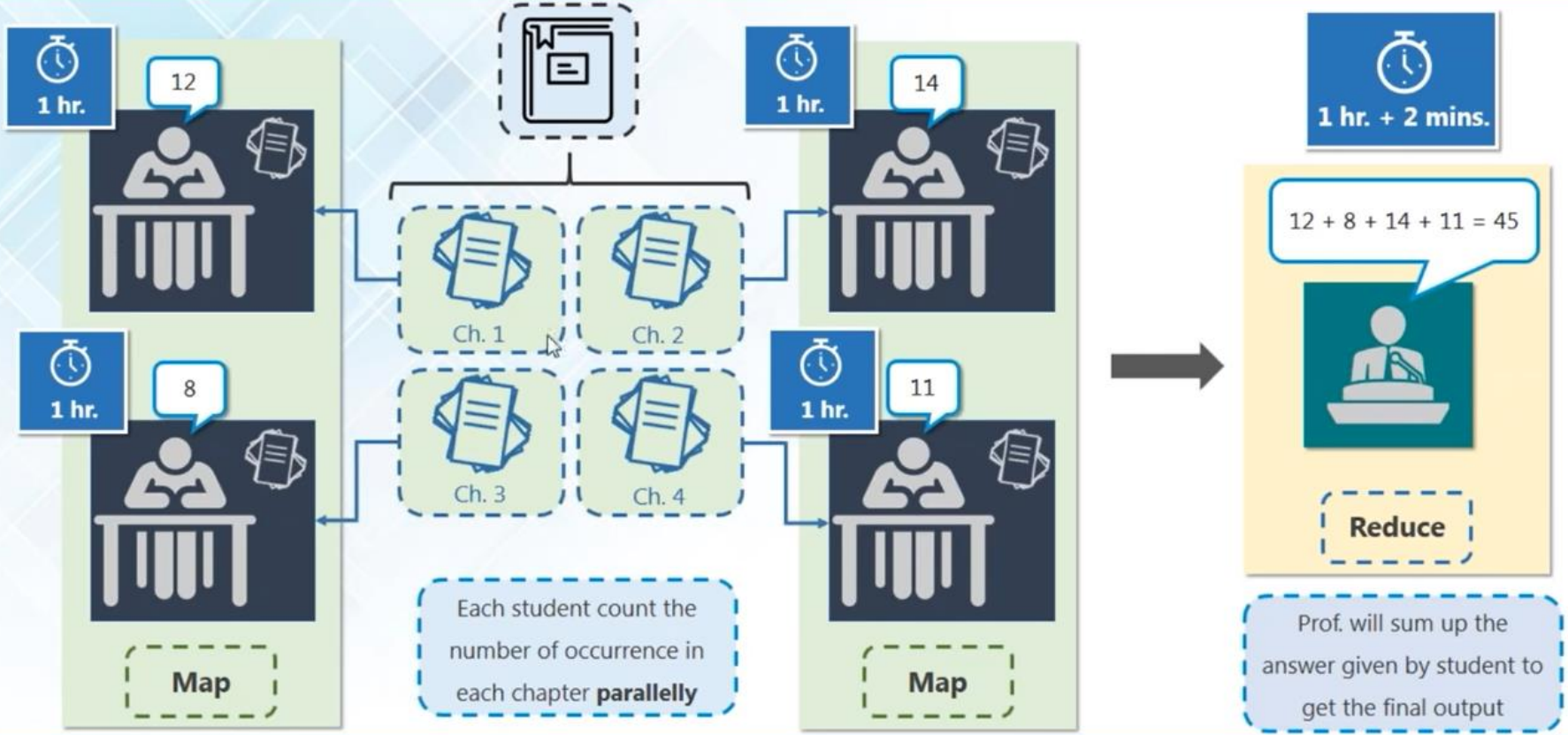
Unit 4: MAPREDUCE

Story of MapReduce



Story of MapReduce

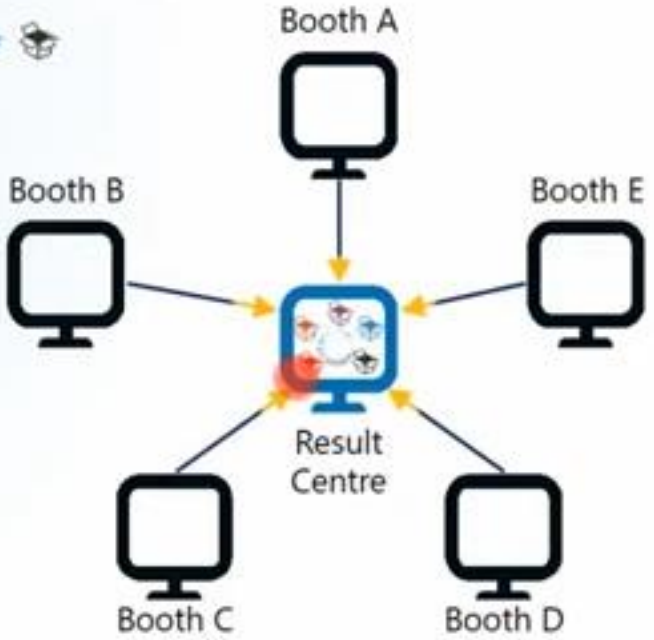


Election Votes Counting – Traditional Way

Data →

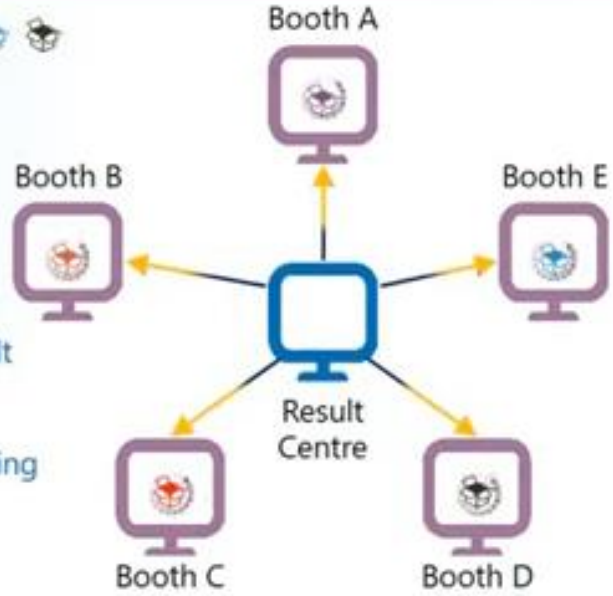
Counting – Traditional Approach

- Votes are moved to Result Centre for counting
- Moving all the votes to Centre is costly
- Result Centre is over-burdened
- Counting takes time



Election Votes Counting – MapReduce Way

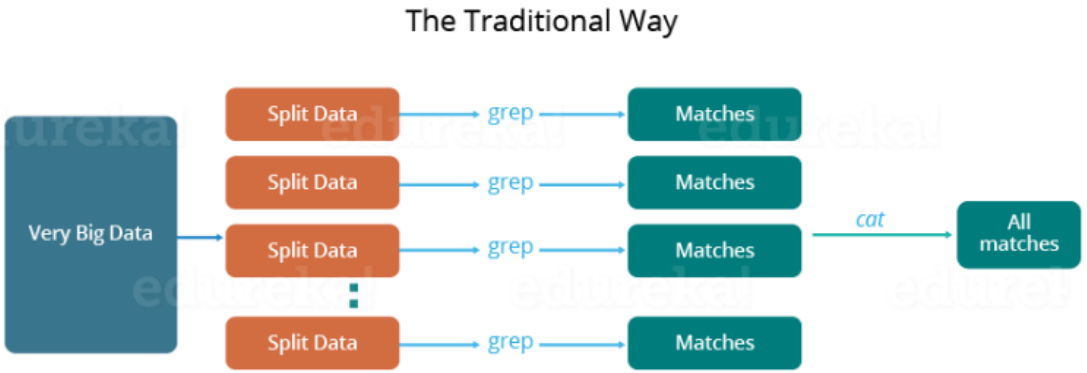
Votes→

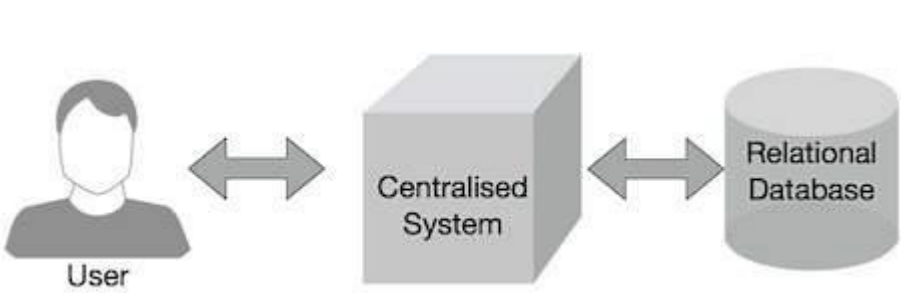


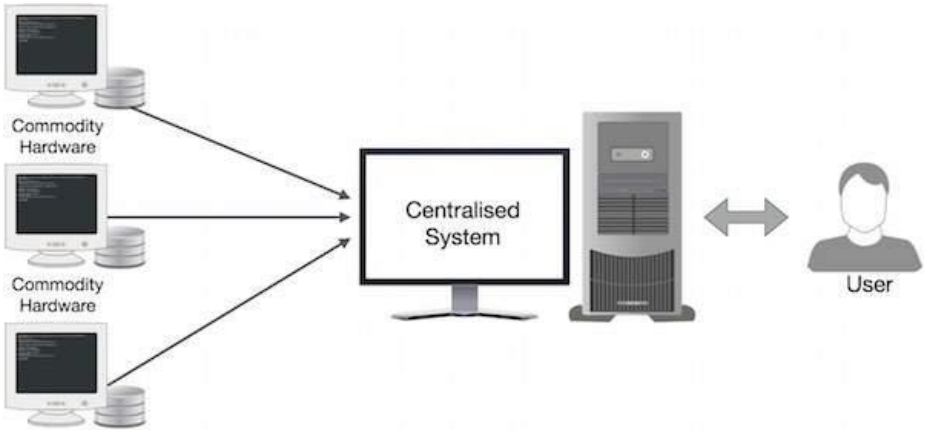
Counting – MapReduce Approach

- Votes are counted at individual booths
- Booth-wise results are sent back to the result centre
- Final Result is declared easily and quickly using this way

Traditional Way



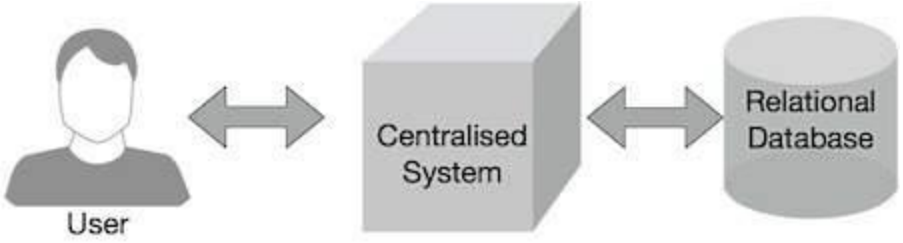


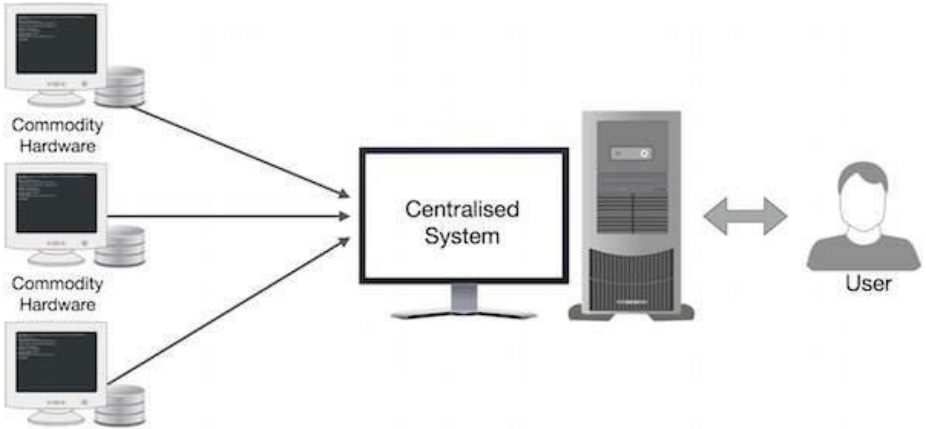


Traditional Way

- let us take an example where I have a weather log containing the daily average temperature of the years from 2000 to 2015. Here, I want to calculate the day having the highest temperature in each year.
- So, just like in the traditional way, I will split the data into smaller parts or blocks and store them in different machines. Then, I will find the highest temperature in each part stored in the corresponding machine. At last, I will combine the results received from each of the machines to have the final output.

Traditional Way





Challenges in traditional approach

- **Critical path problem: delay problem**
- **Reliability problem:** What if, any of the machines which are working with a part of data fails?
- **Equal split issue:** how to equally divide the data such that no individual machine is overloaded or underutilized.
- **The single split may fail: no combined result**
- **Aggregation of the result:**
- To overcome these issues, we have the MapReduce framework which allows us to perform such parallel computations without bothering about the issues like reliability, fault tolerance etc. Therefore, MapReduce gives you the flexibility to write code logic without caring about the design issues of the system.

challenges in traditional approach

1. **Critical path problem:** It is the amount of time taken to finish the job without delaying the next milestone or actual completion date. So, if, any of the machines delay the job, the whole work gets delayed.
2. **Reliability problem:** What if, any of the machines which are working with a part of data fails? The management of this failover becomes a challenge.
3. **Equal split issue:** How will I divide the data into smaller chunks so that each machine gets even part of data to work with. In other words, how to equally divide the data such that no individual machine is overloaded or underutilized.
4. **The single split may fail:** If any of the machines fail to provide the output, I will not be able to calculate the result. So, there should be a mechanism to ensure this fault tolerance capability of the system.
5. **Aggregation of the result:** There should be a mechanism to aggregate the result generated by each of the machines to produce the final output.

MapReduce

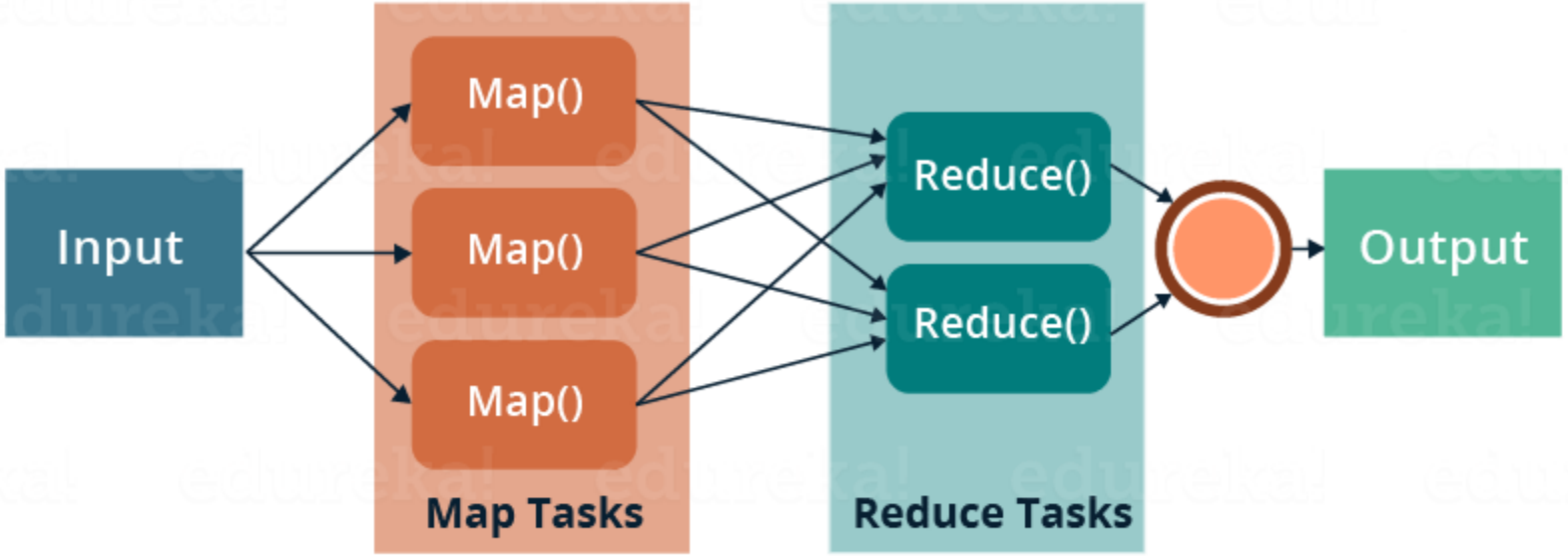
- A MapReduce is a data processing tool which is used to process the data parallelly in a distributed form.
- It was developed in 2004, on the basis of paper titled as **"MapReduce: Simplified Data Processing on Large Clusters,"** published by Google
- Inspired by Google's paper, **Doug Cutting and Mike Cafarella** created **Hadoop**, an open-source implementation of MapReduce.
- MapReduce is a programming model and an associated implementation for processing and generating large data sets.
- Google developed **GFS (Google File System)** and **MapReduce** to efficiently process **web search indexes** across massive clusters.
- Many real world tasks are expressible in this model
- Apache Spark , Apache Flink, Apache Beam, Google Cloud Dataflow

MapReduce

- Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines.
- MapReduce runs on a large cluster of commodity machines and is highly scalable:a typical MapReduce computation processes many terabytes of data on thousands of machines.
- Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

What is MapReduce?

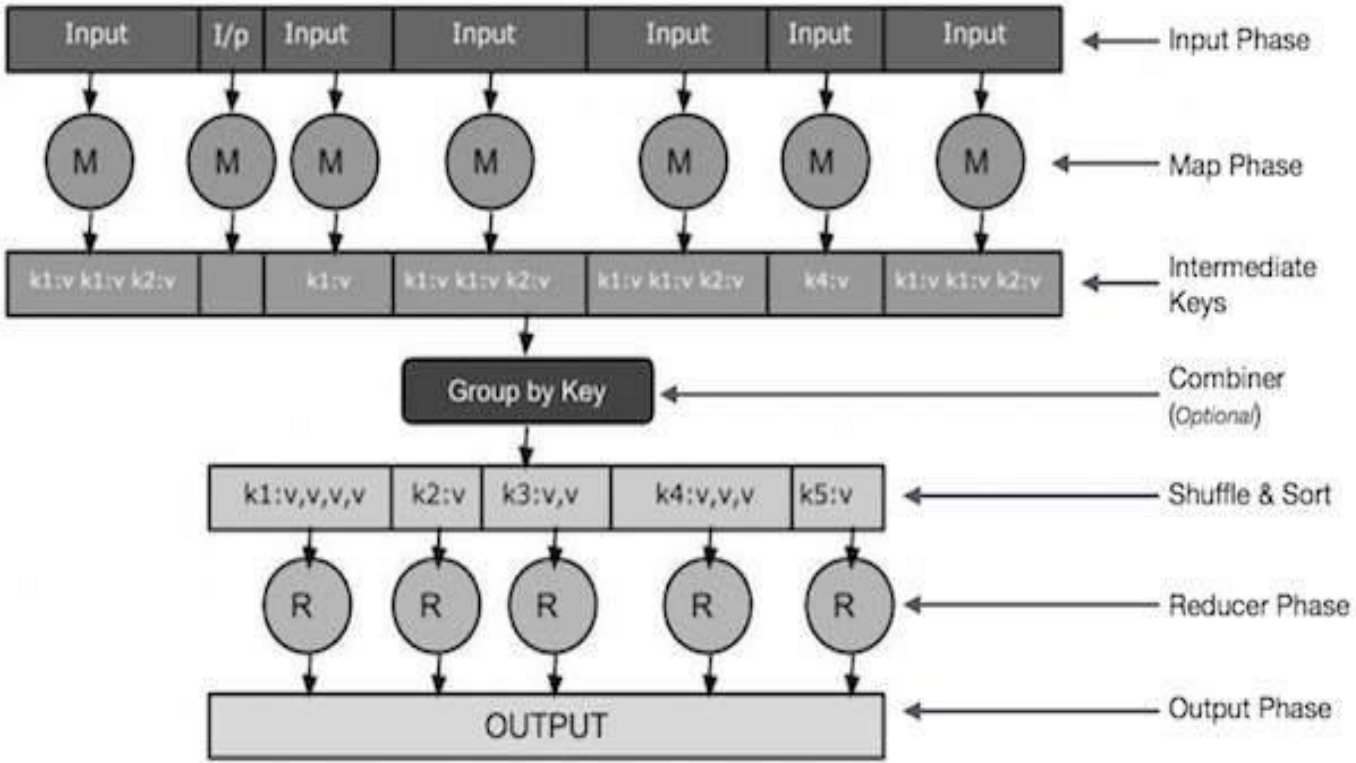
MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment



How MapReduce Works?

- 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).**
- The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.
- The reduce task is always performed after the map job.
- The reducer receives the key-value pair from multiple map jobs.
- Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair) into a smaller set of tuples or key-value pairs which is the final output.

How MapRed uce Works?



Input Splitting:

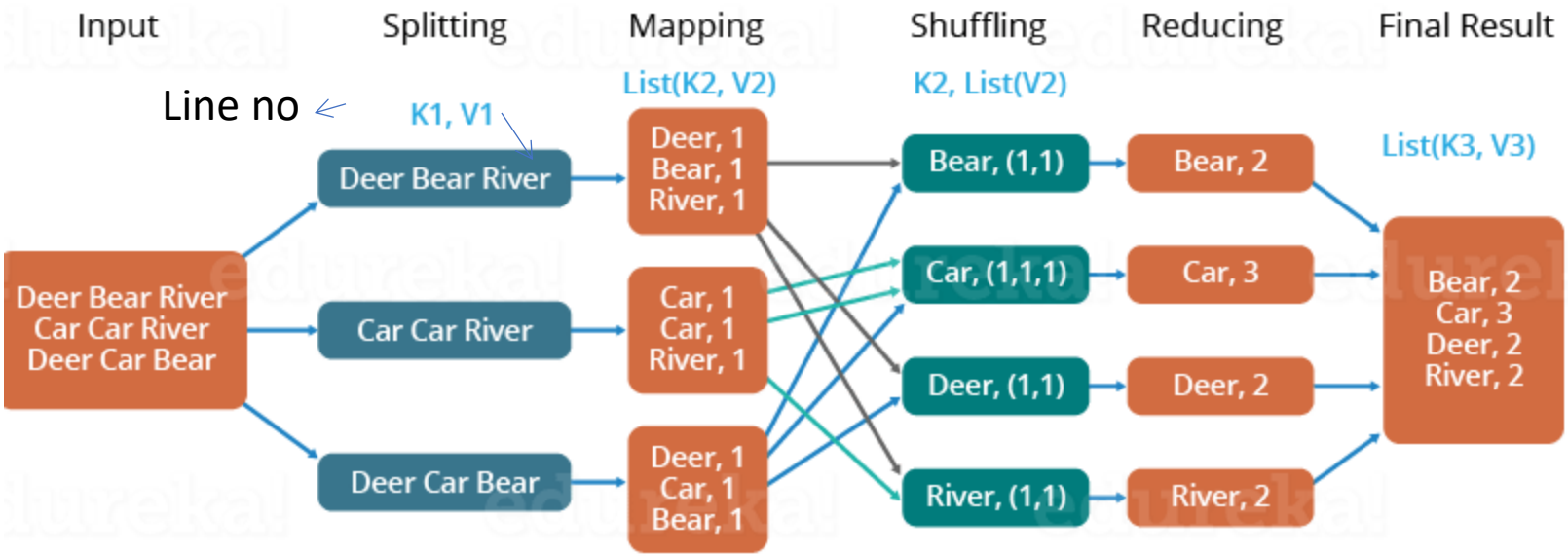
Mapping

- **Input Phase** −The input reader reads the upcoming data and splits it into the data blocks of the appropriate size (64 MB to 128 MB). Here we have a Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-value pairs.
- **Map** − Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-value pairs.
- **Intermediate Keys** − They key-value pairs generated by the mapper are known as intermediate keys.
- **Combiner** − A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.
- **Shuffle and Sort** − The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.
- **Reducer** − The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the final step.
- **Output Phase** − In the output phase, we have an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.

Example:

- Suppose a text file called example.txt whose contents are as follows:
  - **Dear, Bear, River, Car, Car, River, Deer, Car and Bear**

Example


Input
Splitting
Mapping
Shuffling
Reducing
Final Result
Line no
K1, V1
List(K2, V2)
K2, List(V2)
List(K3, V3)
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, (1,1)
Car, (1,1,1)
Deer, (1,1)
River, (1,1)
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

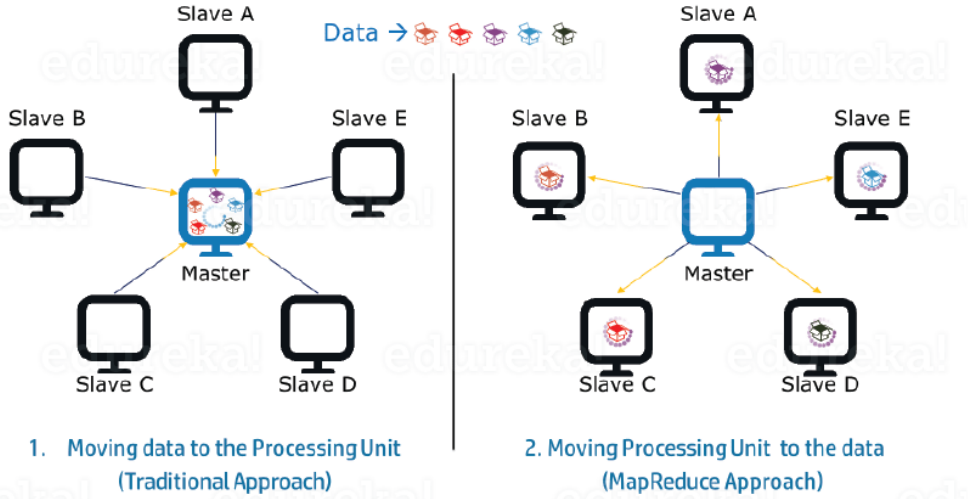
- First, divide the input into three splits as shown in the figure. This will distribute the work among all the map nodes.
- Then, tokenize the words in each of the mappers and give a hardcoded value (1) to each of the tokens or words. The rationale behind giving a hardcoded value equal to 1 is that every word, in itself, will occur once.
- Now, a list of key-value pair will be created where the key is nothing but the individual words and value is one. So, for the first line (Dear Bear River) we have 3 key-value pairs – Dear, 1; Bear, 1; River, 1. The mapping process remains the same on all the nodes.
- After the mapper phase, a partition process takes place where sorting and shuffling happen so that all the tuples with the same key are sent to the corresponding reducer.
- So, after the sorting and shuffling phase, each reducer will have a unique key and a list of values corresponding to that very key. For example, Bear, [1,1]; Car, [1,1,1].., etc.
- Now, each Reducer counts the values which are present in that list of values. As shown in the figure, reducer gets a list of values which is [1,1] for the key Bear. Then, it counts the number of ones in the very list and gives the final output as – Bear, 2.
- Finally, all the output key/value pairs are then collected and written in the output file.

MapReduce algorithm

- MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiple systems.
- These mathematical algorithms may include the following −
- Sorting
- Searching
- Indexing
- TF-IDF(Term Frequency-Inverse Document Frequency)

Advantages of MapReduce

- **Parallel Processing:I**n 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



- **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.
  - The 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.
- 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 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.

Usage of MapReduce

- It can be used in various application like document clustering, distributed sorting, and web link-graph reversal.
- It can be used for distributed pattern-based searching.
- use MapReduce in machine learning.
- Used by Google to regenerate Google's index of the World Wide Web.
- It can be used in multiple computing environments such as multi-cluster, multi-core, and mobile environment.

spiceworks

USES OF MAPREDUCE

1. Search Engines (Google, Bing, Yahoo!)

- **Use Case:** Indexing billions of web pages for search results.

1. Entertainment: Assists end users in finding the most popular movies based on their preferences and previous viewing history. It primarily concentrates on their **clicks and logs.**

Various OTT services, including Netflix, regularly release many web series and movies. It may have happened to you that you couldn't pick which movie to watch, so you looked at Netflix's recommendations and decided to watch one of the suggested series or films. Netflix uses Hadoop and MapReduce to indicate to the user some well-known movies based on what they have watched and which movies they enjoy. MapReduce can examine user clicks and logs to learn how they watch movies.

2. E-commerce: Several e-commerce companies, including Flipkart, Amazon, and eBay, employ MapReduce to evaluate consumer buying patterns based on customers' interests or historical purchasing patterns. For various e-commerce businesses, it provides product suggestion methods by analyzing data, purchase history, and user interaction logs.

Many e-commerce vendors use the MapReduce programming model to identify popular products based on customer preferences or purchasing behavior. Making item proposals for e-commerce inventory is part of it, as is looking at website records, purchase histories, user interaction logs, etc., for product recommendations.

3. **Social Media Analytics (Facebook, Twitter, LinkedIn)**

- **Use Case:** Analyzing user behavior, trending hashtags, and ad targeting.

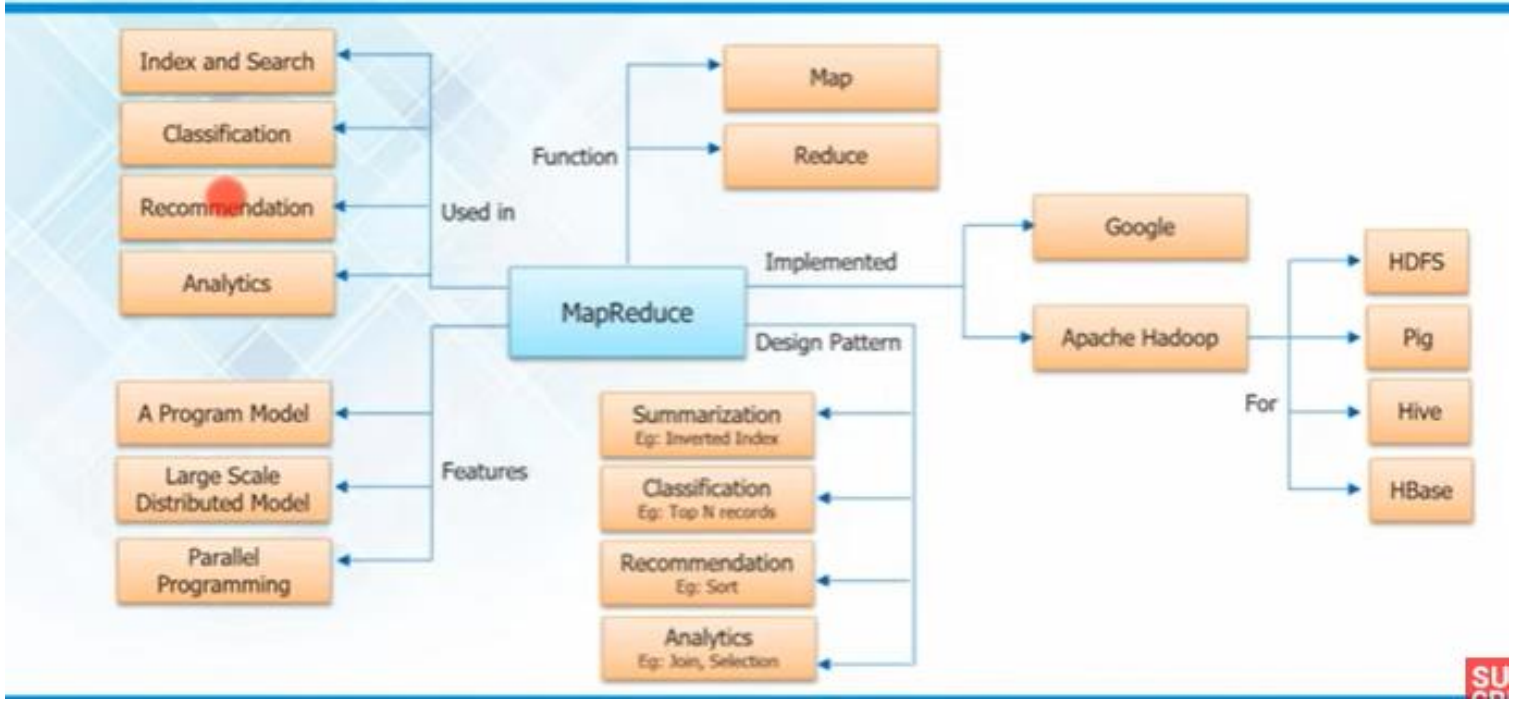
5. Fraud detection: **Use Case:** Detecting anomalies in financial transactions and system logs.

Financial businesses, including banks, insurance companies, and payment locations, use Hadoop and MapReduce for fraud detection, pattern recognition evidence, and business analytics through transaction analysis.

4. Data warehouse: Systems that handle enormous volumes of information are known as data warehouse systems. The star schema, which consists of a fact table and several dimension tables, is the most popular data warehouse model. In a shared-nothing architecture, storing all the necessary data on a single node is impossible, so retrieving data from other nodes is essential.

This results in network congestion and slow query execution speeds. If the dimensions are not too big, users can replicate them over nodes to get around this issue and maximize parallelism. Using MapReduce, we may build specialized business logic for data insights while analyzing enormous data volumes in data warehouses.

MapReduce In Nutshell



- https://www.spiceworks.com/tech/big-data/articles/what-is-map-reduce/
- https://www.edureka.co/blog/mapreduce-tutorial/