Big Data Analytics

Vidyalankar Institute of Technology
Accredited A+ by NAAC

Academic Year 2022-23

Index

1. MapReduce
2. MapReduce Workflow
3. MapReduce Word Count

Subject Teacher: **Prof. Prakash Parmar, Assistant Professor, CMPN**

MapReduce: Data Processing using Programming

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

- MapReduce is a framework used for writing application that processes large data sets in a distributed manner with parallels algorithms.
- It is core component of the Apache Hadoop ecosystem
- MapReduce have main two function
  1. Map
  2. Reduce
- Both these Map & Reduce works only on key-value pairs

  Ex. (Roll_No, Name)

  (1, Sohan)

  (2, Mohan)
- In both function, input is Key-value pair and output also Key, Value pair

  (k, v) -> Map -> (k, v)   -> Reduce -> (k, v)

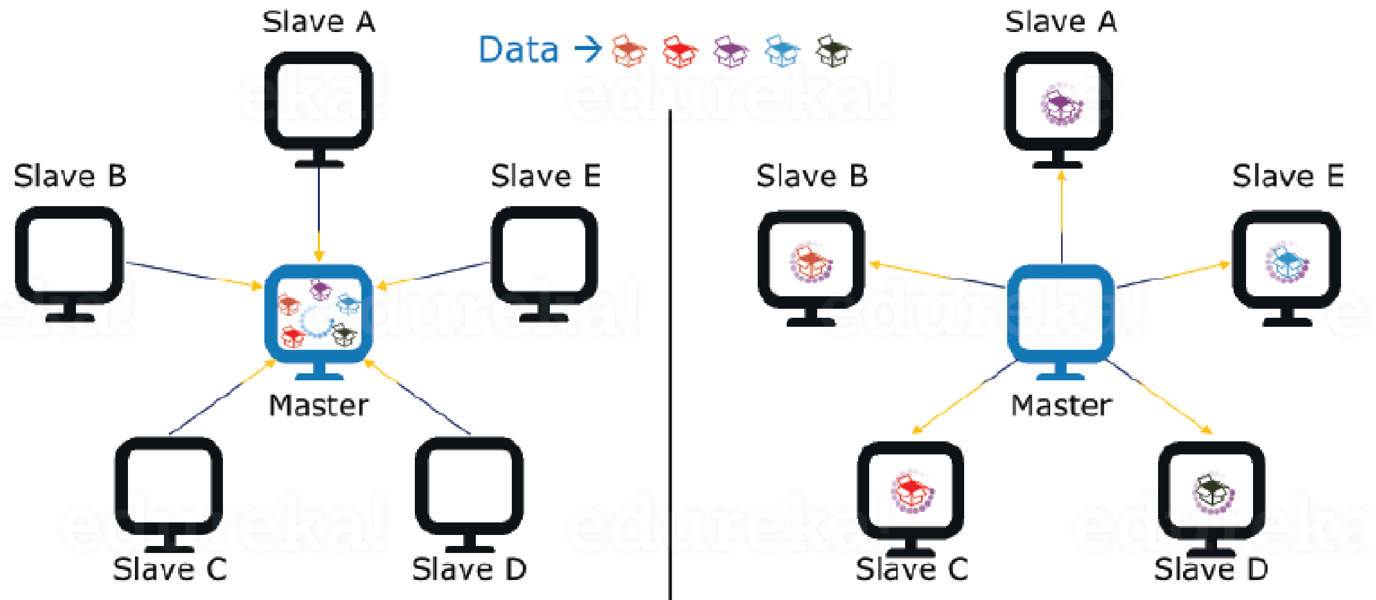
MapReduce, Why?

- MapReduce is a programming paradigm
- Traditional programming models work only when data is kept on a single machine and if data kept on multiple machine in a distributed manner than we require a new programming model to solve the problems.

- ***MapReduce is a computing paradigm for processing data that resides on many machines.***

MapReduce: Working Mechanism

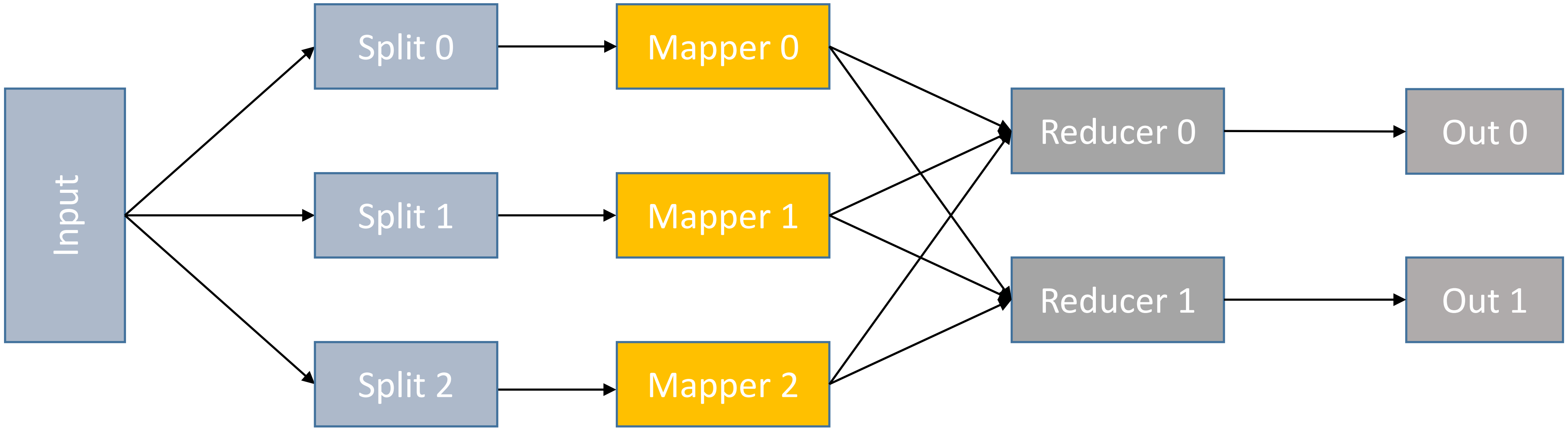
- Hadoop works on principle of data Locality, i.e., the data is processed where it is kept.
- In MapReduce code is move to data, data is not coming towards code, it makes fast processing.
- Data Block – 128 MB but code around 20 KB.



1. Moving data to the Processing Unit (Traditional Approach)

2. Moving Processing Unit to the data (MapReduce Approach)

MapReduce: Work Flow



MapReduce: Word Count

Vidyalankar Institute of Technology
Accredited A+ by NAAC

Input

Car Bike Car Motor
Car Bike Motor
Car Bike Car
Bike Car
Bike Car Bike

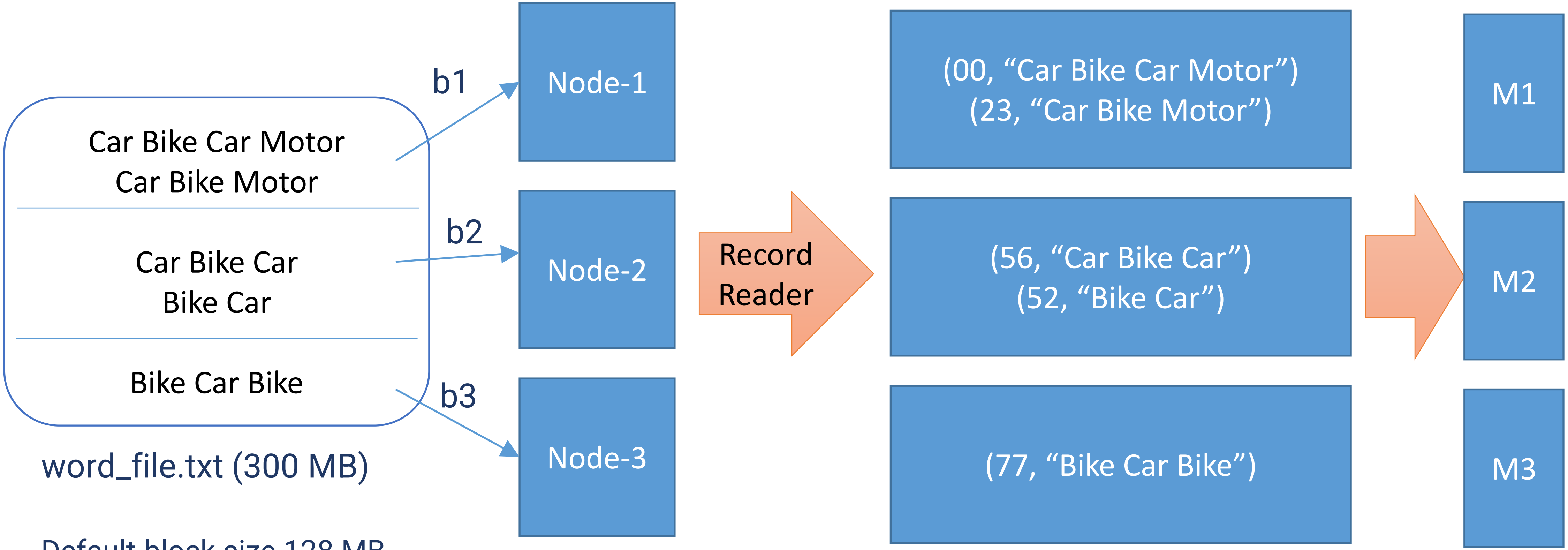
word_file.txt (300 MB)

Output

'Car': 7,
'Bike': 6,
'Motor': 2

Expected output

MapReduce: Word Count



Default block size 128 MB
So file divided into 3 block

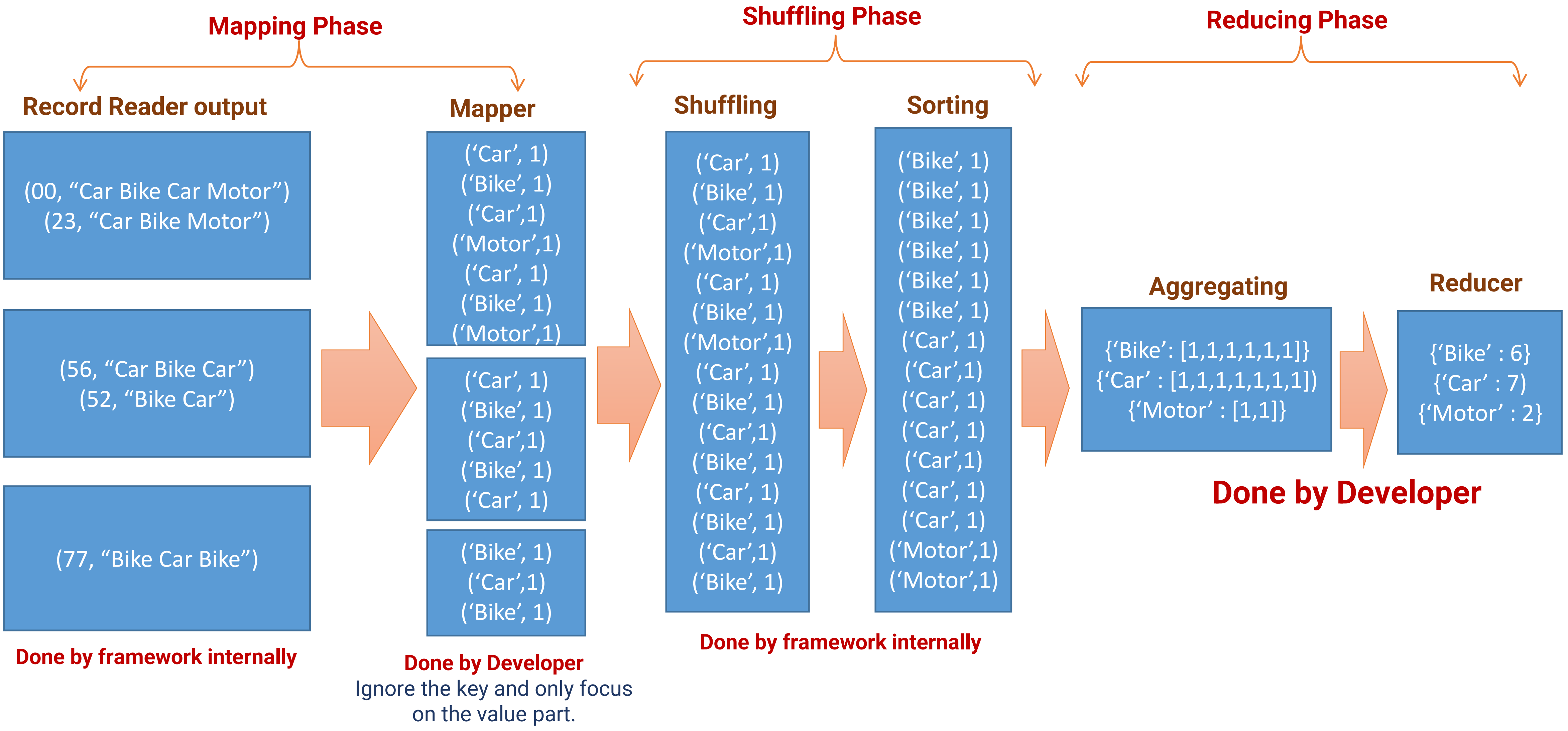
Record reader takes the each line as input, add dummy key and return key-value pairs as output.
It is done by framework internally.

Mapper

Mapper Program /logic has to be written by the developer.

MapReduce: Word Count

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

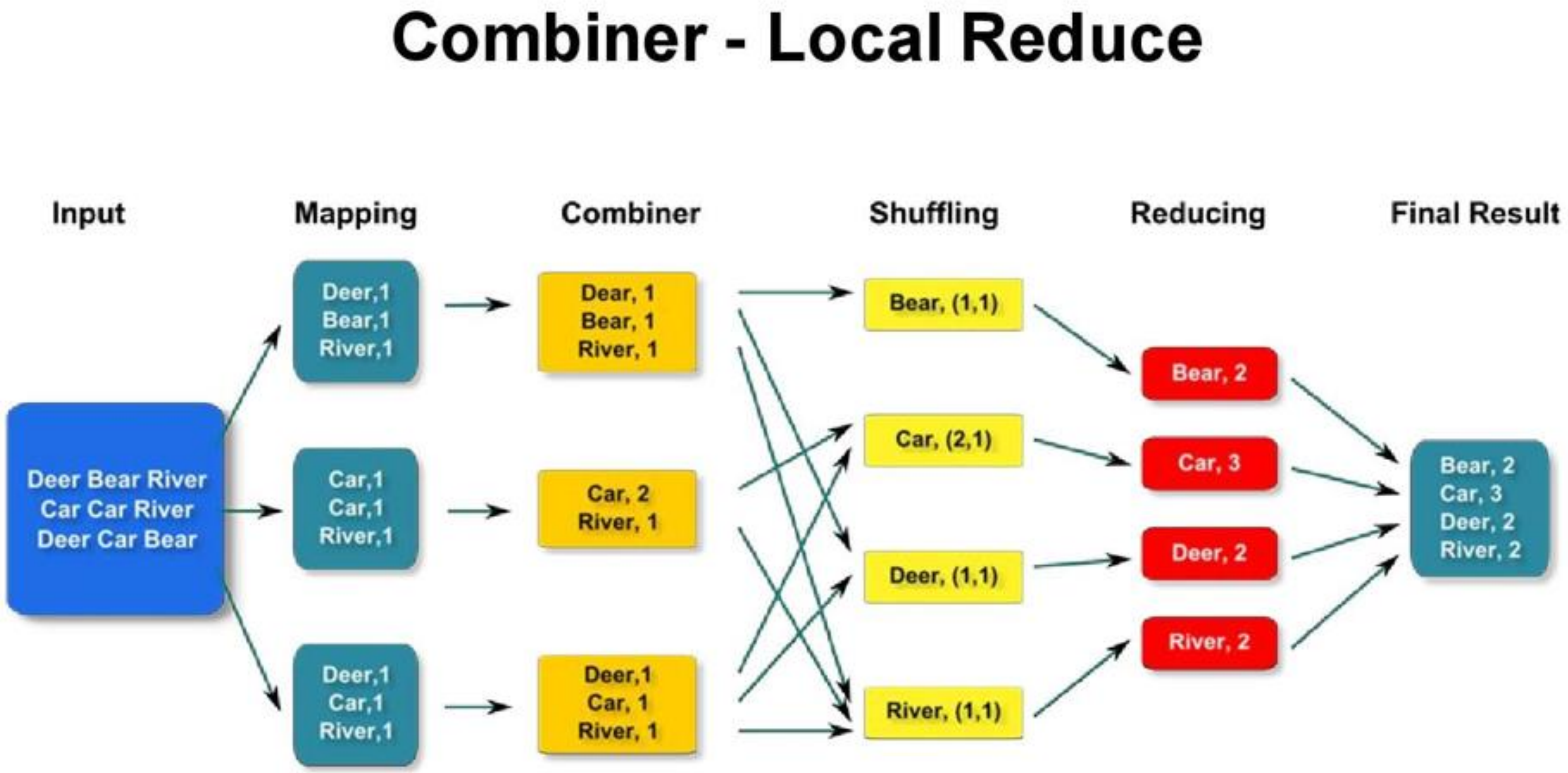


MapReduce with Combiner: Work Flow

Vidyalankar Institute of Technology
Accredited A+ by NAAC

- Combiner is also known as "**Mini-Reducer**" that synopsizes the Mapper output record with the same Key before passing to the Reducer.
- On a huge dataset when we run a MapReduce job. So Mapper creates large chunks of intermediate data. Then the framework passes this intermediate data on the Reducer for further handling. This leads to huge network congestion. The Hadoop framework offers a function known as Combiner that plays a key role in reducing network congestion.
- The main job of Combiner a "Mini-Reducer is to handle the output data from the Mapper, before passing it to Reducer. It works after the mapper and before the Reducer. Its usage is optional.

MapReduce with Combiner: Work Flow



MapReduce with Combiner

Advantages

- Use of combiner decreases the time taken for data transfer between mapper and reducer.
- Combiner increases the overall performance of the reducer.
- It reduces the amount of data that the reducer has to process.

Disadvantages

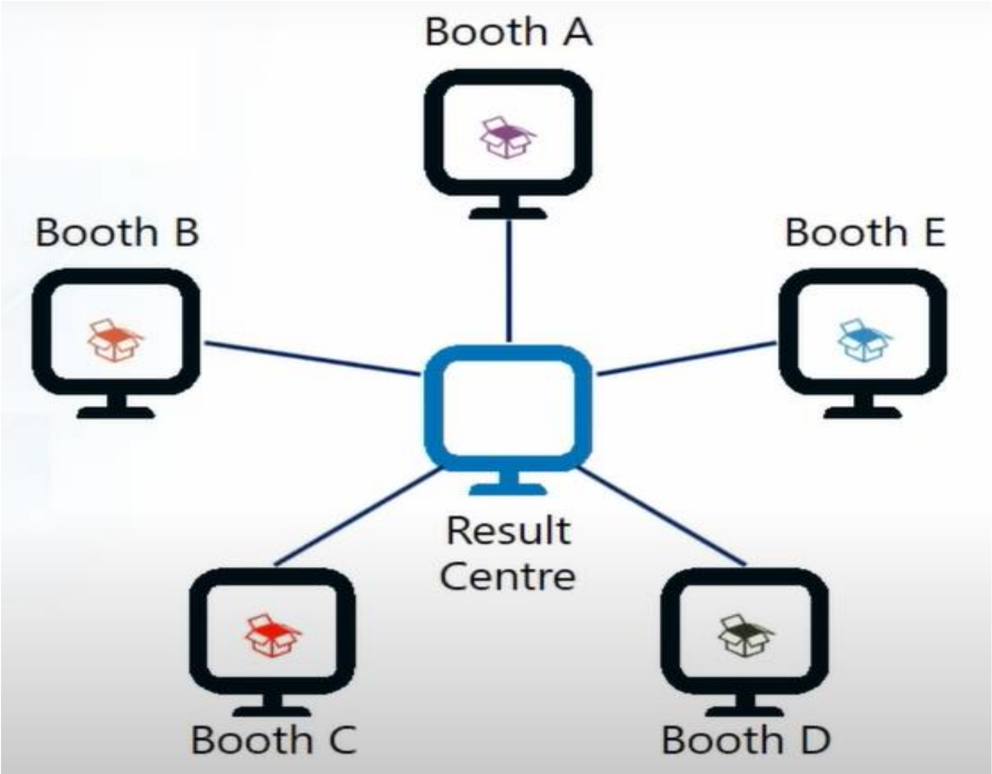
- In the native filesystem, when Hadoop stores the key-value pairs and runs the combiner later this will result in expensive disk IO.
- MapReduce jobs can’t rely on the combiner execution as there is no guarantee in its execution.

MapReduce Use Case: Election Vote Counting

Votes is stored at different Booths and Result centre has the details of all the booths

Counting: Traditional Approach

- Votes are moved to result centre for counting.
- Moving all the votes to centre is costly.
- Results centre is over burdened.
- Counting takes time.

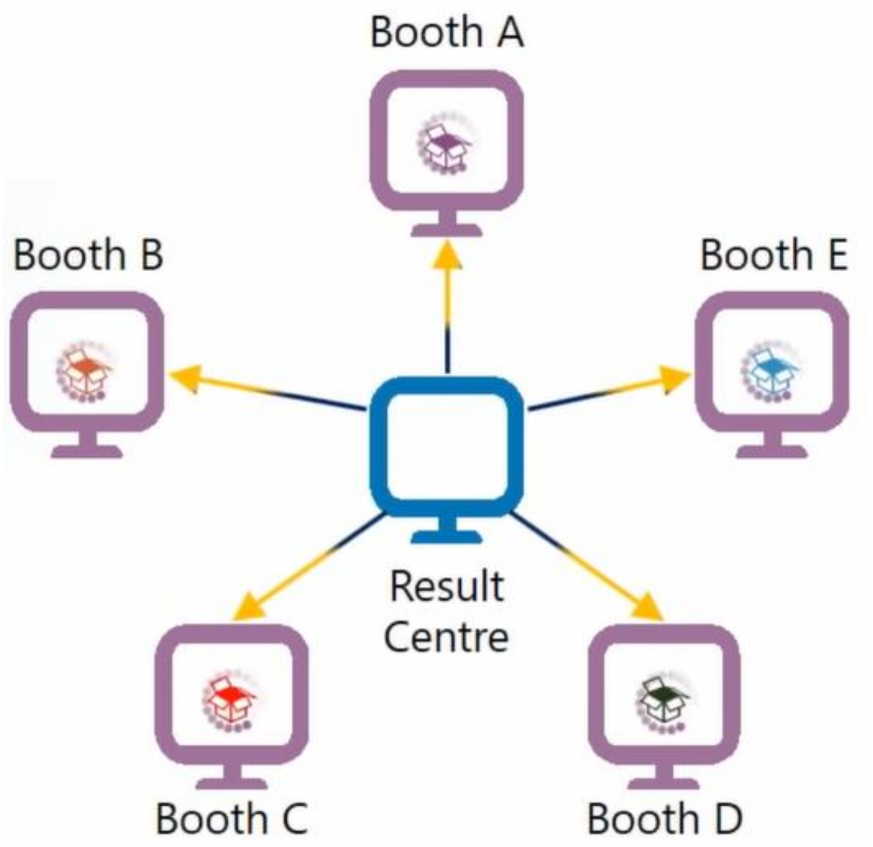


MapReduce Use Case: Election Vote Counting

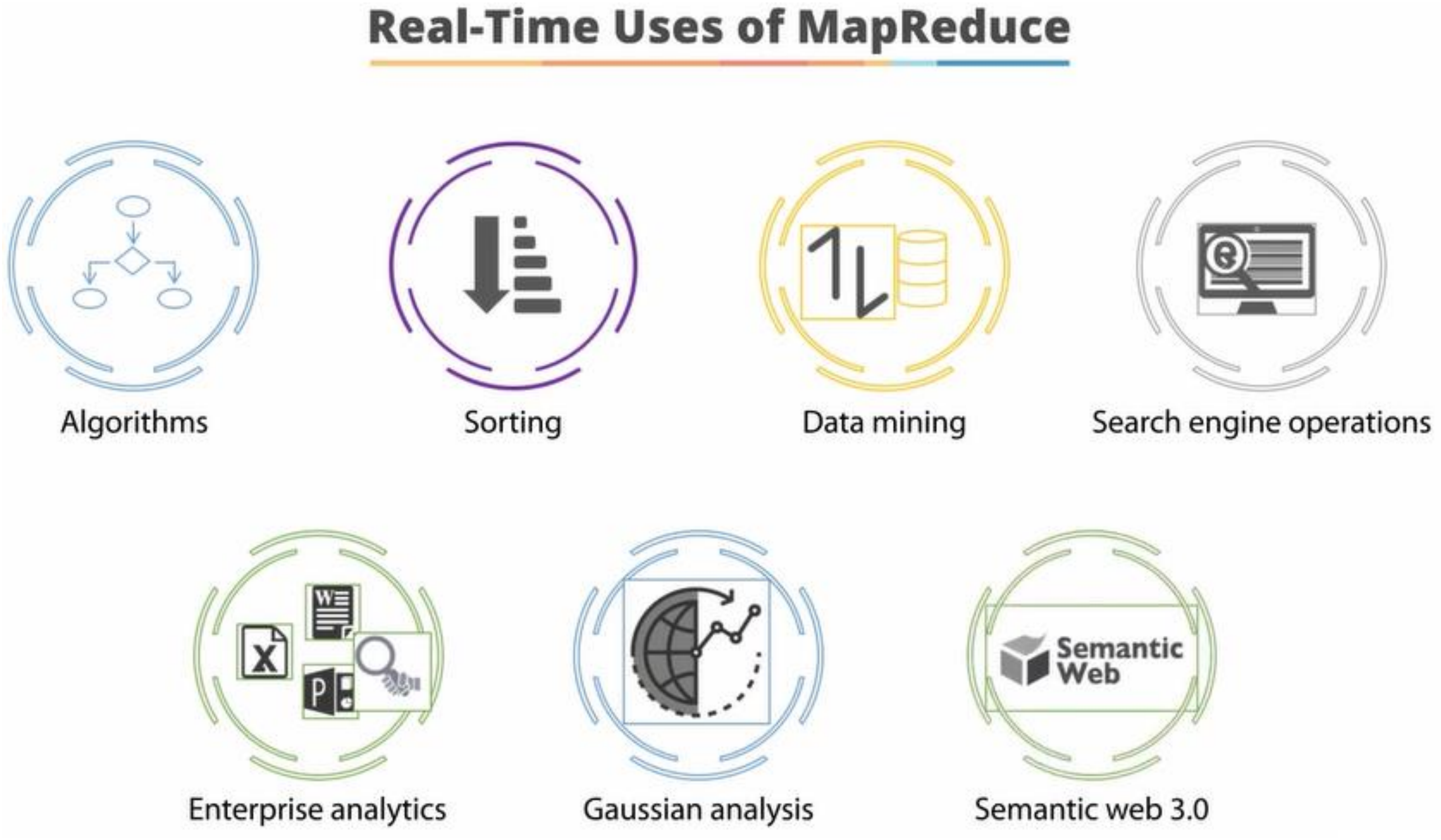
Votes is stored at different Booths and Result centre has the details of all the booths

Counting: MapReduce Approach

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

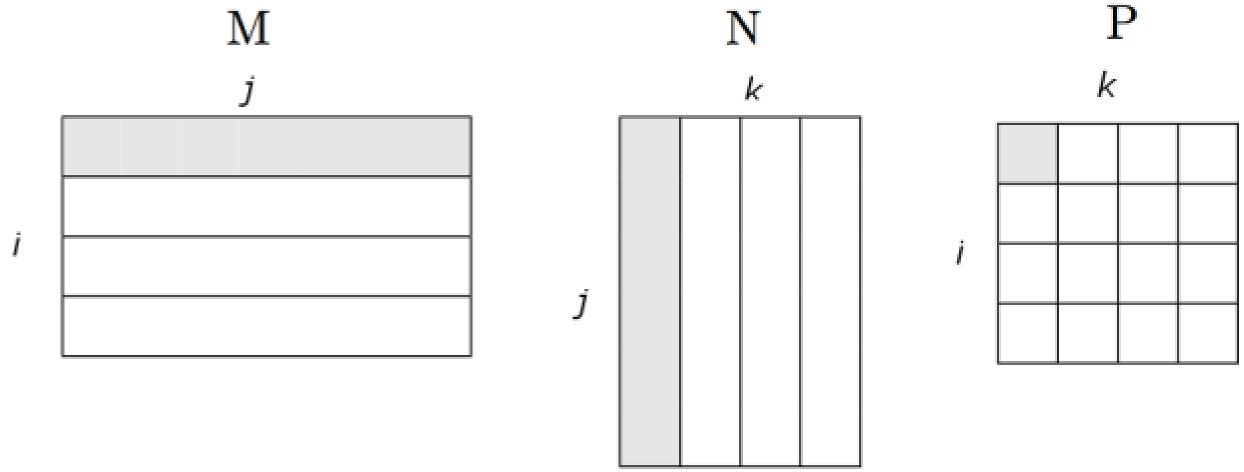


Real-Time Uses of MapReduce



Algorithm uses MapReduce : Matrix Vector Multiplication

Matrix-vector and matrix-matrix calculations fit nicely into the MapReduce style of computing.



Matrix Data Model for MapReduce

Matrix Multiplication using MapReduce

Algorithm 1: The Map Function

for *each element* m_{ij} *of* M **do**
    produce $(key, value)$ pairs as $((i,k),(M,j,m_{ij}))$, for $k = 1,2,3,..$ up to the number of columns of N
for *each element* n_{jk} *of* N **do**
    produce $(key, value)$ pairs as $((i,k),(N,j,n_{jk}))$, for $i = 1,2,3,...$ up to the number of rows of M
return *Set of (key, value) pairs that each key,* (i,k)*, has a list with values* (M,j,m_{ij}) *and* (N,j,n_{jk}) *for all possible values of* j

Algorithm 2: The Reduce Function

for *each key (i,k)* **do**
    sort values begin with M by j in $list_M$
    sort values begin with N by j in $list_N$
    multiply m_{ij} and n_{jk} for j_{th} value of each list
    sum up $m_{ij} * n_{jk}$
return $(i,k), \sum_{j=1} m_{ij} * n_{jk}$

Matrix Multiplication using MapReduce

Matrix Data Model for MapReduce

$$\boldsymbol{M_{ij}} = \begin{bmatrix} \mathbf{10} & \mathbf{20} \\ \mathbf{30} & \mathbf{40} \end{bmatrix} \qquad \boldsymbol{N_{jk}} = \begin{bmatrix} \mathbf{5} & \mathbf{6} \\ \mathbf{7} & \mathbf{8} \end{bmatrix}$$

As input data files, we store matrix M and N on HDFS in following format

(M, i, j, m_{ij}) and (N, i, j, m_{ij})

$(\boldsymbol{M}, \mathbf{1}, \mathbf{1}, \mathbf{10})$, $(\boldsymbol{M}, \mathbf{1}, \mathbf{2}, \mathbf{20})$, $(\boldsymbol{M}, \mathbf{2}, \mathbf{1}, \mathbf{30})$ … …

$(\boldsymbol{N}, \mathbf{1}, \mathbf{1}, \mathbf{5})$, $(\boldsymbol{N}, \mathbf{1}, \mathbf{2}, \mathbf{6})$, $(\boldsymbol{N}, \mathbf{2}, \mathbf{1}, \mathbf{7})$ … …

Note:

Most matrices are sparse so large amount of cells have value zero. When we represent matrices in this form, we do not need to keep entries for the cells that have values of zero to save large amount of disk space.

Matrix Multiplication using MapReduce

for each element m_{ij} of M do

produce (key, value) pairs as $\left((i,k),(M,j,M_{ij})\right)$

for k=1,2, .. Up to column of matrix N

for each element n_{jk} of N do

produce (key, value) pairs as $\left((i,k),(N,j,N_{jk})\right)$

for i=1,2, .. Up to row of matrix M

Mapper code for Matrix M

k=#colums of N	i = row number of M	j = column number of M	formula (key, value) $\left((i,k),(M,j,m_{ij})\right)$
k=1	i=1	j=1	((1,1), (M, 1, 10))
		j=2	((1,1), (M, 2, 20))
	i=2	j=1	((2,1), (M, 1, 30))
		j=2	((2,1), (M, 2, 40))
k=2	i=1	j=1	((1,2), (M, 1, 10))
		j=2	((1,2), (M, 2, 20))
	i=2	j=1	((2,2), (M, 1, 30))
		j=2	((2,2), (M, 2, 40))

Mapper code for Matrix N

i=#colums of M	j = row number of N	k = column number of N	formula (key, value) $\left((i,k),(N,j,n_{ij})\right)$
i=1	j=1	k=1	((1,1), (N, 1, 5))
		k=2	((1,2), (N, 1, 6))
	j=2	k=1	((1,1), (N, 2, 7))
		k=2	((1,2), (N, 2, 8))
i=2	j=1	k=1	((2,1), (N, 1, 5))
		k=2	((2,2), (N, 1, 6))
	j=2	k=1	((2,1), (N, 2, 7))
		k=2	((2,2), (N, 2, 8))

Matrix Multiplication using MapReduce

Return Set of (key, value) pairs that same key has a list with values M & N

((1,1), [(M, 1, 10)), (M, 2, 20)]),

((2,1), [(M, 1, 30)), (M, 2, 40)]),

((1,2), [(M, 1, 10)), (M, 2, 20)]),

((2,2), [(M, 1, 30)), (M, 2, 40)])

((1,1), [(N, 1, 5)), (N, 2, 7)]),

((1,2), [(N, 1, 6)), (N, 2, 8)]),

((2,1), [(N, 1, 5)), (N, 2, 7)]),

((2,2), [(N, 1, 6)), (N, 2, 8)]),

Matrix Multiplication using MapReduce

Algorithm 2: The Reduce Function

```
for each key (i,k) do
    sort values begin with M by j in list_M
    sort values begin with N by j in list_N
    multiply m_ij and n_jk for j_th value of each list
    sum up m_ij * n_jk
return (i,k), Σ_{j=1} m_ij * n_jk
```

Reduce Code:

Sort values in M and N list according to J and perform multiplication.

((1,1), 10*5+20*7) => ((1,1), 190)

((1,2), 10*6+20*8) => ((1,2), 220)

((2,1), 30*5+40*7) => ((2,1), 430)

((2,2), 30*6+40*8) => ((2, 2), 500)

Result :

$$M_{ij} = \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} \quad N_{jk} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \quad M \times N = \begin{bmatrix} 190 & 220 \\ 430 & 500 \end{bmatrix}$$

Failure in MapReduce

- Failures are norm in commodity hardware
- **Worker** failure
  - Detect failure via periodic heartbeats
  - Re-execute in-progress map/reduce tasks
- **Master** failure
  - Single point of failure; Resume from Execution Log
- **Robust**
  - Google's experience: lost 1600 of 1800 machines once!, but finished fine

Learn Fundamentals & Enjoy Engineering

Thank You

Prof. Prakash Parmar
Assistant Professor
Computer Engineering Department
Vidyalankar Classes CSE GATE Faculty