



Introduction to Data Science

(Lecture 23)

Dr. Mohammad Pourhomayoun

Assistant Professor

Computer Science Department

California State University, Los Angeles





Map Reduce

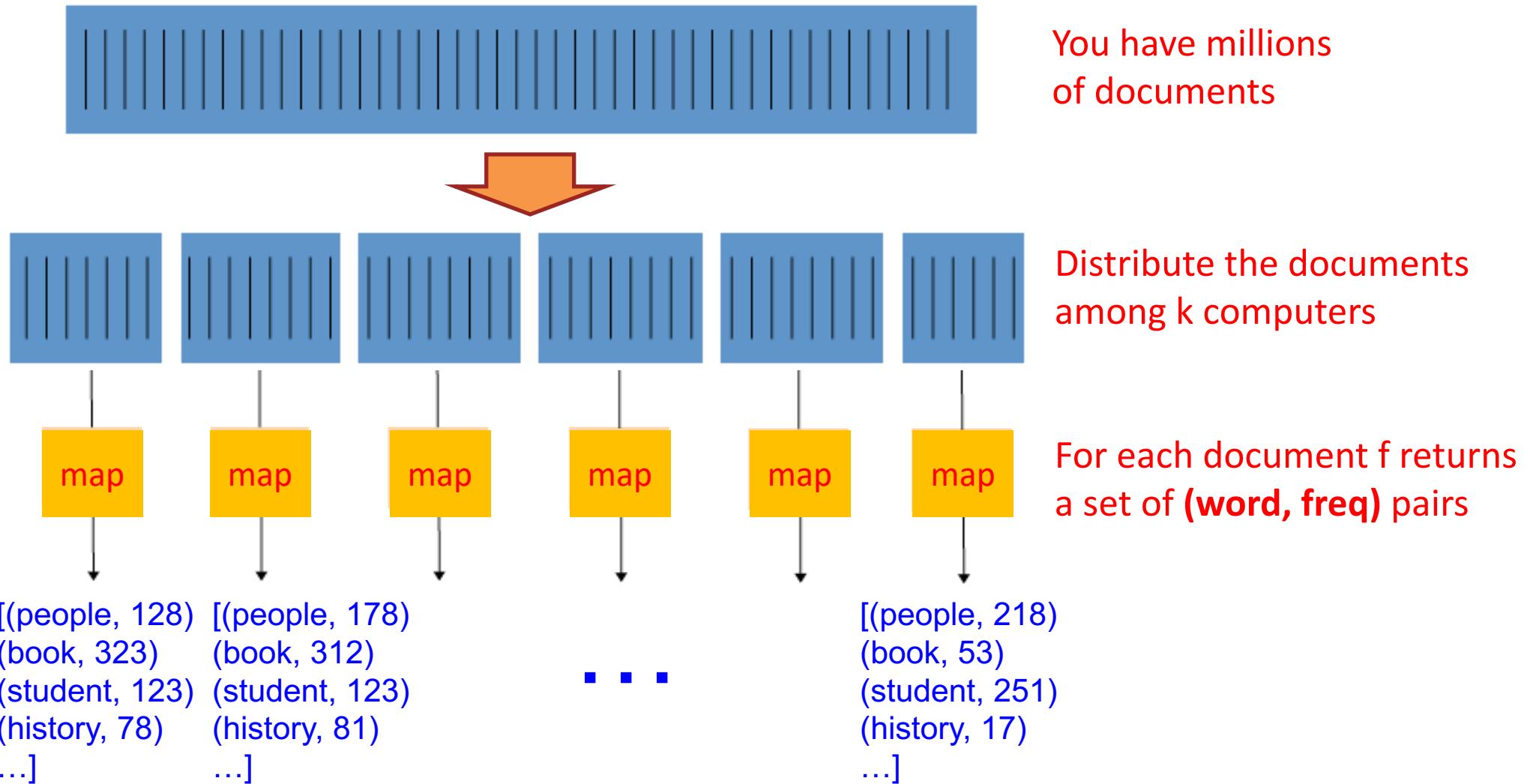
Map-Reduce

- **Map-Reduce** is a **programming model** for processing and generating **big data sets** with a **parallel and distributed** algorithm.
- **map function** processes input **key/value pairs** to generate a set of intermediate **key/value pairs**.
- **reduce function** merges all intermediate values associated with the same intermediate key.

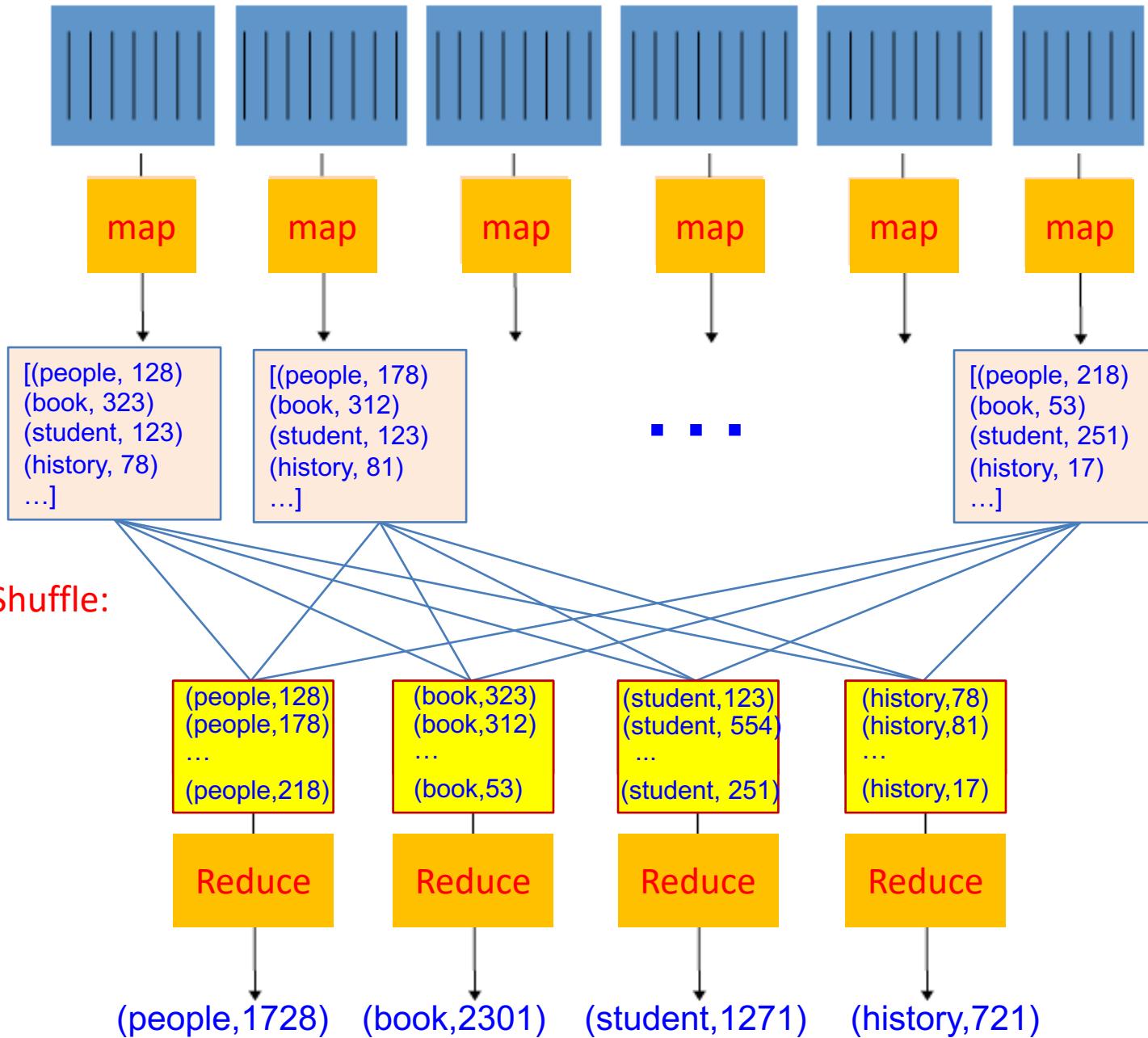
[Ref]: Dean, Jeffrey & Ghemawat, Sanjay. (2004). MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM.



Example: Compute overall word frequency across 5M docs



MAP



Map: Count All words in Each chunk of data

Reduce: Count the occurrences of Each word in the Entire data

MAP-REDUCE

- In this example:
 - **Map:** Counts All words in Each chunk of data
 - **Reduce:** Counts the occurrences of Each word in the Entire dataset





Map Reduce for Relational Database Operations

Example: Relational Join

- **Relational Join:** Stick the tuples of two relations together when they agree on common attributes (column names).
- Consider two database tables $R(A, B)$ and $S(B, C)$.
- **R JOIN S :** $T(A, B, C)$ formed by joining rows $(a, b) \in R$ and $(b, c) \in S$ with **matching b**.
- **Example:** $R(A,B) \text{ JOIN } S(B,C) = \{a,b,c \mid a,b \text{ is in } R \text{ and } b,c \text{ is in } S\}$.

A	B
6	2
12	2
7	5

R

B	C
2	9
5	11
5	3
9	5

S

A	B	C
6	2	9
12	2	9
7	5	11
7	5	3

R JOIN S



The Map Function for Join

- Each tuple (a,b) in R is mapped to:

key = b, value = (R,a).

- Note: “R” in the value is just a bit to indicate “this value represents a tuple in R, not S.”

- Each tuple (b,c) in S is mapped to:

key = b, value = (S,c).

- After grouping by keys (shuffle), each reducer gets a key-list that looks like:

$(b, [(R,a_1), (R,a_2), \dots, (S,c_1), (S,c_2), \dots]).$



The Map Function for Join

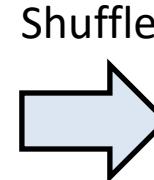
A	B
6	2
12	2
7	5

R =

B	C
2	9
5	11
5	3
9	5

S =

Map on R: $\begin{cases} (2, (R,6)) \\ (2, (R,12)) \\ (5, (R,7)) \end{cases}$



(2, [(R,6), (R,12), (S,9)])
(5, [(R,7), (S,11), (S,3)])
(9, [(S,5)])

Map on S: $\begin{cases} (2, (S,9)) \\ (5, (S,11)) \\ (5, (S,3)) \\ (9, (S,5)) \end{cases}$

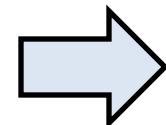
The Reduce Function for Join

- After grouping by keys, each reducer gets a key-list that looks like:
$$(b, [(R,a_1), (R,a_2), \dots, (S,c_1), (S,c_2), \dots]).$$
- Reducer generates a tuple (a,b,c) for **each** pair of (R,a_i) and (S,c_j) on the list with key b .



The Reduce Function for Join

(2, [(R,6), (R,12), (S,9)])
(5, [(R,7), (S,11), (S,3)])
(9, [(S,5)])



A	B	C
6	2	9
12	2	9
7	5	11
7	5	3



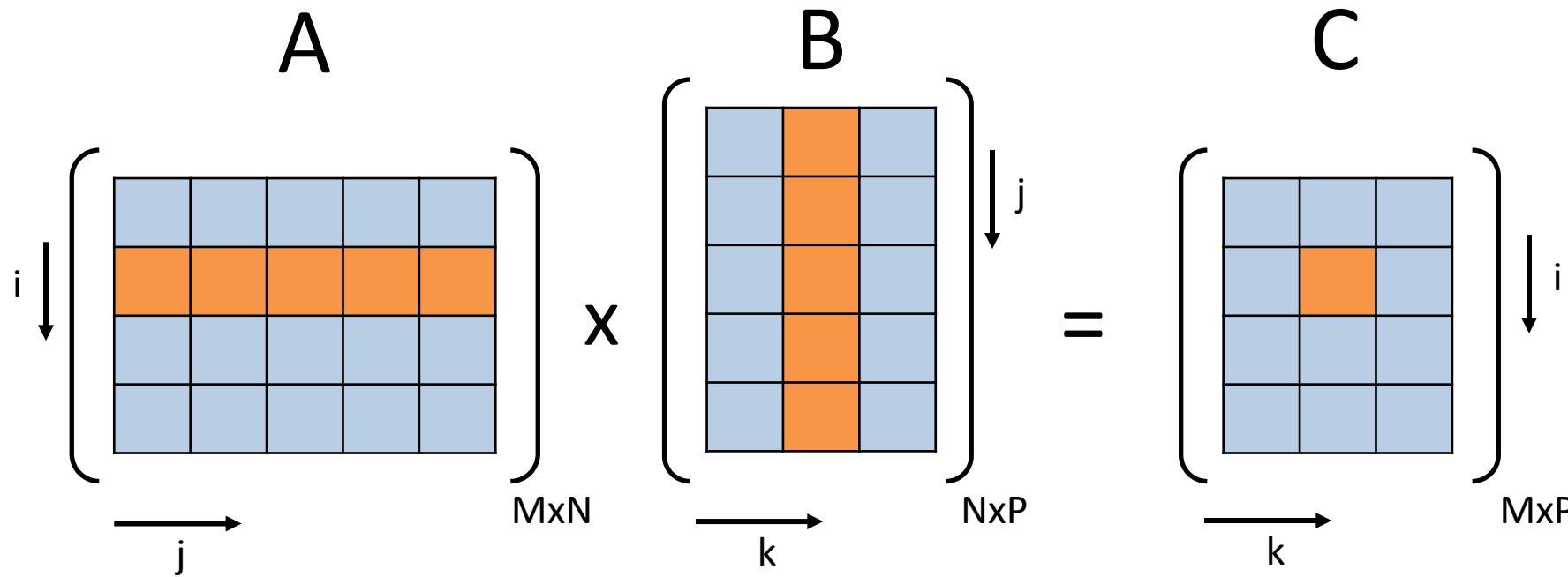
Map Reduce for Matrix and Vector Multiplication

MapReduce for Matrix Multiplication

The diagram illustrates the multiplication of two matrices, A and B, resulting in matrix C. Matrix A is a 4x5 grid of light blue squares, labeled 'A' above and 'MxN' below. Matrix B is a 5x3 grid of light blue squares, labeled 'B' above and 'NxP' below. The multiplication is shown as $A \times B = C$, where C is a 4x3 grid of light blue squares, labeled 'C' above and 'MxP' below. Brackets on the left and right sides group the matrices A and B, and the result C, respectively.



Matrix Multiplication



$$c_{ik} = \sum_j a_{ij} \times b_{jk}$$

Simple Pseudo Code for Matrix Multiplication (general approach)

```
for i = 1 to n do
    for j = 1 to n do
        for k = 1 to n do
            C[i,j] = C[i,j] + A[i,k] x B[k,j]
        endfor
    endfor
endfor
```



Special Case: Matrix-Vector Multiplication

- Map-Reduce originally developed by **Google** in order to compute the **PageRank** vector.

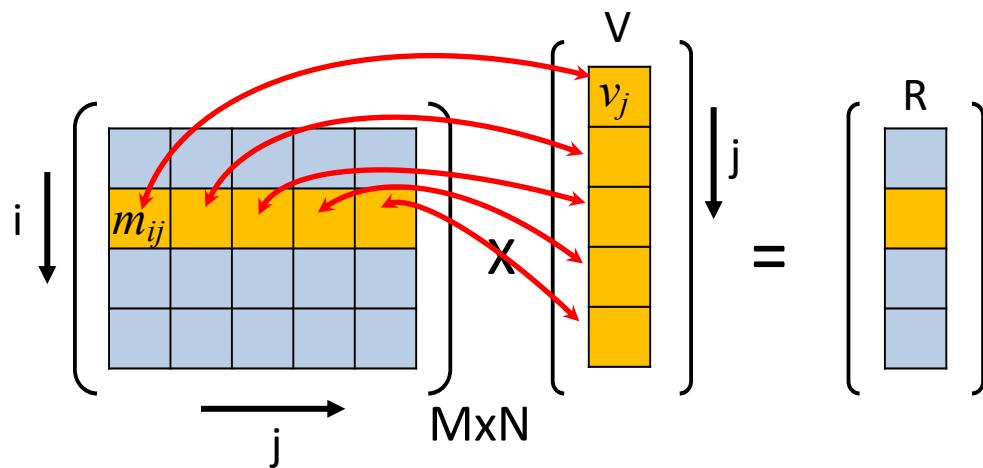
$$\begin{matrix} M \\ \times \\ V \\ = \\ R \end{matrix} \quad \begin{array}{c} \text{M} \\ \left(\begin{array}{|c|c|c|c|c|} \hline & m_{ij} & & & \\ \hline \end{array} \right) \\ \times \\ \left(\begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline \end{array} \right) \\ \text{Nx1} \\ \text{Mx1} \end{array}$$

The diagram illustrates matrix-vector multiplication. On the left, a matrix M of size $M \times N$ is shown as a grid of N columns. A specific element m_{ij} is highlighted in yellow. An arrow labeled i points to the i -th row, and an arrow labeled j points to the j -th column. To the right, a vector V of size $N \times 1$ is represented as a vertical stack of N components, all colored yellow. The result of the multiplication is a scalar R of size $M \times 1$, represented as a vertical stack of M components, where the i -th component is the sum of the i -th row of M and the corresponding component of V .

Map for Matrix-Vector Multiplication

- **Map Function:**

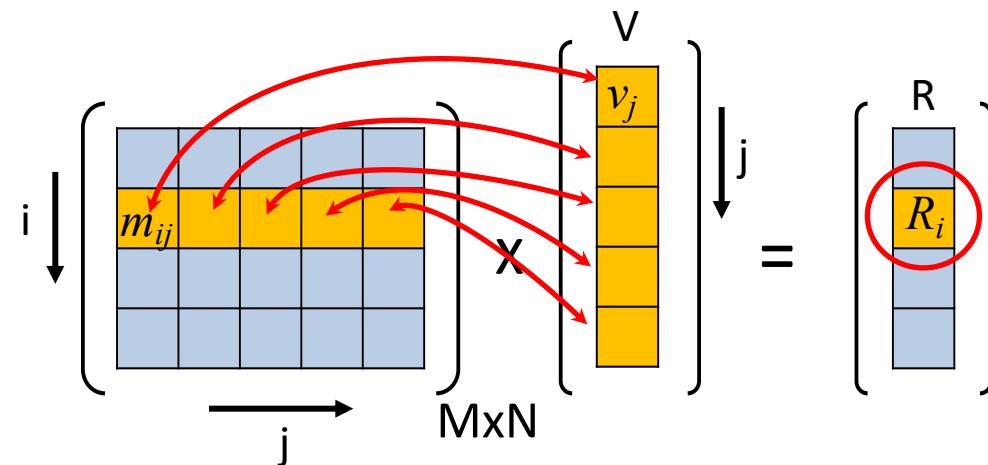
- N mappers process row i of the matrix at a time
- Mapper j Maps $((i, j), m_{ij})$ to $(i, m_{ij} v_j)$
- Note that We assumed that each mapper can load vector v .



Reduce for Matrix-Vector Multiplication

- **Reduce function:**
 - M reducers calculate elements of vector R
 - Reducer i receives $(i, [m_{i1} v_1, \dots, m_{iN} v_N])$, sums all values of the list of a key i , and produces (i, R_i) :

$$R_i = \sum_j m_{ij} \times v_j$$





Map Reduce for Matrix to Matrix Multiplication (Optional)

MapReduce for Matrix-Matrix Multiplication

(Optional: Not in Final Exam)

- Multiply matrix $M = [m_{ij}]$ by $N = [n_{jk}]$.

- $P = M \times N$:

$$P = [p_{ik}], \text{ where } p_{ik} = \sum_j m_{ij} \times n_{jk}.$$

- In many applications, Large matrices are very **sparse** (mostly 0's). Thus, to save memory, we can represent a sparse matrix only using its nonzero elements m_{ij} in the form of tuples (i, j, m_{ij}) . zero elements are not represented at all!
- There are two main approaches for Matrix Multiplication using MapReduces:
Single Pass and **Double Pass**.



Single-Pass MapReduce for Matrix Multiplication

(Optional: Not in Final Exam)

- **Map function:**
 - maps each m_{ij} to **key = (i,k) , value = (M,j,m_{ij})** , for all k
 - maps each n_{jk} to **key = (i,k) , value = (N,j,n_{jk})** , for all i
 - Similar to join, M and N here are bits indicating which matrix the value comes from.



Single-Pass MapReduce for Matrix Multiplication

(Optional: Not in Final Exam)

- **Reduce function:**

- For each (M, j, m_{ij}) on the list for key (i, k) find the (N, j, n_{jk}) with the **same j** .
- Multiply m_{ij} by n_{jk} and then sum up to generate $((i, k), p_{ik})$:

$$p_{ik} = \sum_j m_{ij} \times n_{jk}$$

- $((i, k), p_{ik})$ means that the element (i, k) of the output matrix has value of p_{ik} !



Double-Pass MapReduce for Matrix Multiplication (Optional): The First Pass

- **Map function:**

- maps each m_{ij} to **key = j , value = (M, i, m_{ij})**
- maps each n_{jk} to **key = j , value = (N, k, n_{jk})**

- **Reduce function:**

- for key j , pair each (M, i, m_{ij}) on its list with each (N, k, n_{jk}) , and generate: **key = (i, k) , value = $m_{ij} \times n_{jk}$.**



Double-Pass MapReduce for Matrix Multiplication (Optional): The Second Pass

- **Map function:**
 - Pair each key (i,k) with the list of products $m_{ij} \times n_{jk}$ for all j :
$$((i,k), [m_{i1} \times n_{1k}, m_{i2} \times n_{2k}, \dots, m_{iN} \times n_{Nk}])$$
- **Reduce function:**
 - For each (i,k) , add up all the elements on the list, and generates $((i,k), p_{ik})$:

$$p_{ik} = \sum_j m_{ij} \times n_{jk}$$





Thank You!

Questions?