Department of Computer Science and Engineering (Data Science)

Machine Learning – IV

Experiment 2

Name: Umang Kirit Lodaya **SAP ID: 60009200032**

Batch: D11

Aim:

To implement and comprehend the process of matrix multiplication using the MapReduce programming model, a scalable and parallelized approach for processing large datasets.

Theory:

Introduction to MapReduce:

- MapReduce is a programming model designed for processing and generating large datasets in parallel across distributed clusters.
- Comprises two main steps: Map and Reduce.

Matrix Multiplication Overview:

- **Input Matrices:**
    - *A* - Matrix of size *m*×*p*
    - *B* - Matrix of size *p*×*n*
    - Result *C* - Matrix of size *m*×*n*
- **Map Step:**
    - Mapper Function:
        - For each element *A*[*i*][*j*] or *B*[*j*][*k*], emit key-value pairs where the key is the resulting cell's coordinates (i, k).
        - Value is a tuple indicating the source (A or B), row (i or k), and value.

- **Shuffling and Sorting:**
    - The MapReduce framework shuffles and sorts the emitted key-value pairs, grouping them by the resulting cell's coordinates.
- **Reduce Step:**
    - **Reducer Function:**
        - For each group of values with the same key, perform the matrix multiplication and sum the products to get the result $C[i][k]$.

Step-by-Step Implementation:

- **Step 1: Mapper Function**
    - Implement the mapper function to emit key-value pairs.
- **Step 2: Shuffling and Sorting**
    - Understand the automatic shuffling and sorting mechanism of the MapReduce framework.
- **Step 3: Reducer Function**
    - Implement the reducer function to perform matrix multiplication.
- **Step 4: Input Data Formatting**
    - Prepare input data in a format suitable for MapReduce processing.
- **Step 5: Experimentation with Input Sizes**
    - Experiment with different sizes of input matrices to observe the scalability of the MapReduce approach.

Implementation Tips:

- Optimize the mapper and reducer functions for efficient data processing.
- Consider partitioning strategies to optimize data distribution.

Lab Experiments to be Performed in This Session:

Execute the Matrix Multiplication using Map Reduce to gain insights into its functionality and operation.

NAME: UMANG KIRIT LODAYA

SAP ID: 60009200032

BATCH: K - K1 / D11

MATRIX MULTIPLICATION USING MAP-REDUCE

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
def INPUT():
    done = False
    I, J, K = None, None, None

    while not done:
        Sa = input('ENTER SHAPE OF MATRIX A (eg. 2,2): ')
        Sb = input('ENTER SHAPE OF MATRIX B (eg. 2,2): ')

        i, j = Sa.split(',')
        jj, k = Sb.split(',')
        if j == jj:
            I = i
            J = j
            K = k
            done = True

        else:
            print('Number of Columns in A is not equal to Number of Rows in

    A = [[None for j in range(J)] for i in range(I)]
    B = [[None for k in range(K)] for j in range(J)]

    totalInputs = I * J + J * K

    while i < totalInputs:
        inp = input('Enter Values for Matrix (eg. A,0,0,4): ').split()
        if len(inp) != 4:
            print('Invalid Values. Please follow format!')
            continue
        if inp[0] == 'A':
            i, j = inp[1], inp[2]
            A[i][j] = inp[3]
        else:
            k, j = inp[1], inp[2]
            A[j][k] = inp[3]

        i += 1

    return A, B
```

In [3]:

```
A = np.array([[1, 2],
              [2, 1],
              [3, 4]])

B = np.array([[1, 2],
              [1, 3]])
```

In [4]:

```
A = np.array([[1, 1, 3],
              [5, 2, 6],
              [2, 3, 4]])

B = np.array([[1, 1, 3],
              [5, 2, 6],
              [-2, -1, -3]])
```

In [5]:

```
# A, B = INPUT()
```

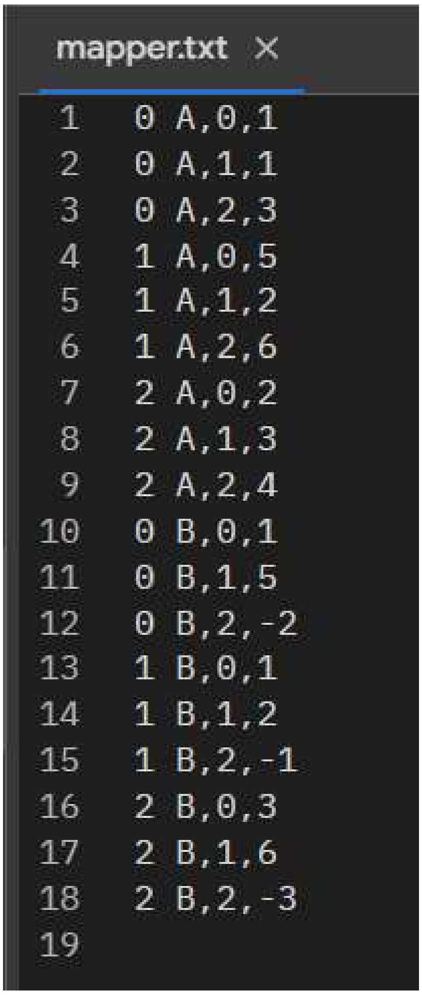
In [6]:

```
def mapper(A, B):
    I, J = A.shape
    J, K = B.shape
    f = open('mapper.txt', 'w')
    for i in range(I):
        for j in range(J):
            f.write(f"{i} A,{j},{A[i][j]}\n")

    for k in range(K):
        for j in range(J):
            print()
            f.write(f"{k} B,{j},{B[j][k]}\n")

    f.close()
```

In [ ]:

```
mapper(A, B)
```

OUTPUT OF MAPPER



```
0 A,0,1
0 A,1,1
0 A,2,3
1 A,0,5
1 A,1,2
1 A,2,6
2 A,0,2
2 A,1,3
2 A,2,4
0 B,0,1
0 B,1,5
0 B,2,-2
1 B,0,1
1 B,1,2
1 B,2,-1
2 B,0,3
2 B,1,6
2 B,2,-3

```

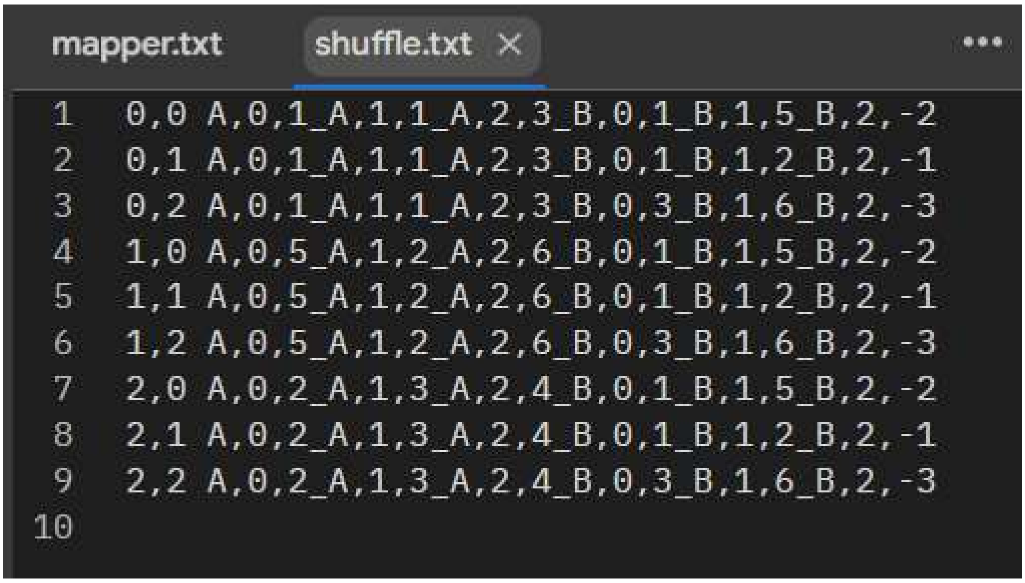
In [8]:

```
def shuffle():
    DATA = None
    with open('mapper.txt', 'r') as f:
        DATA = f.readlines()

    I = 0; K = 0

    for row in DATA:
        k, v = row[:-1].split()
        k = int(k)
        v = v.split(',')
        if v[0] == 'A':
            I = max(k, I)
        else:
            K = max(k, K)

    I += 1; K += 1

    GROUPS = {}
    for i in range(I):
        for k in range(K):
            for row in DATA:
                key, val = row.split()
                key = int(key)
                KEY = f"{i},{k}"
                if (i == key and val[0] == 'A') or (k == key and val[0] == '
                    GROUPS[KEY] = GROUPS.get(KEY, "") + " " + val

    f = open('shuffle.txt', 'w')
    for key in GROUPS:
        f.write(f"{key} {'_'.join(GROUPS[key].split())}\n")
    f.close()
```

In [9]:

```
shuffle()
```

OUTPUT OF SHUFFLER



```
0,0 A,0,1_A,1,1_A,2,3_B,0,1_B,1,5_B,2,-2
0,1 A,0,1_A,1,1_A,2,3_B,0,1_B,1,2_B,2,-1
0,2 A,0,1_A,1,1_A,2,3_B,0,3_B,1,6_B,2,-3
1,0 A,0,5_A,1,2_A,2,6_B,0,1_B,1,5_B,2,-2
1,1 A,0,5_A,1,2_A,2,6_B,0,1_B,1,2_B,2,-1
1,2 A,0,5_A,1,2_A,2,6_B,0,3_B,1,6_B,2,-3
2,0 A,0,2_A,1,3_A,2,4_B,0,1_B,1,5_B,2,-2
2,1 A,0,2_A,1,3_A,2,4_B,0,1_B,1,2_B,2,-1
2,2 A,0,2_A,1,3_A,2,4_B,0,3_B,1,6_B,2,-3
```

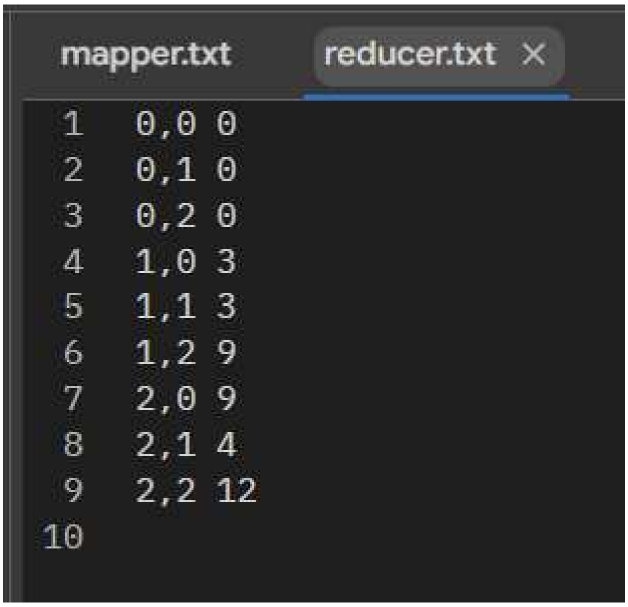
In [10]:

```
def reducer():
    DATA = None
    with open('shuffle.txt', 'r') as f:
        DATA = f.readlines()

    f = open('reducer.txt', 'w')
    j = int(DATA[-1].split()[-1].split('_')[-1].split(',')[1]) + 1

    for row in DATA:
        key, val = row[:-1].split()
        val = val.split('_')
        val = list(map(lambda x: x.split(','), val))
        n = len(val)

        value = 0
        for i in range(0, n - j):
            value += int(val[i][-1]) * int(val[i + j][-1])

        f.write(f"{key} {value}\n")

    f.close()
```

In [11]:

```
reducer()
```

OUTPUT OF REDUCER



```
0,0 0
0,1 0
0,2 0
1,0 3
1,1 3
1,2 9
2,0 9
2,1 4
2,2 12
```

In [12]:

```
def result():
    DATA = None
    with open('reducer.txt', 'r') as f:
        DATA = f.readlines()

    I = 0
    C = []
    temp = []

    for row in DATA:
        k, v = row[:-1].split()
        k = list(map(int, k.split(',')))
        v = int(v)

        if k[0] == I:
            temp.append(v)
        else:
            C.append(temp)
            I = k[0]
            temp = [v]

    C.append(temp)
    for row in C:
        print(row)
```

In [13]:

```
result()
```

```
[0, 0, 0]
[3, 3, 9]
[9, 4, 12]
```