**MODULE-5**

**IMPLEMENTING MATRIX-MULTIPLICATION WITH HADOOP MAP- REDUCE**

In mathematics, matrix multiplication or the matrix product is a binary operation that produces a matrix from two matrices. The definition is motivated by linear equations and linear transformations on vectors, which have numerous applications in applied mathematics, physics, and engineering. In more detail, if A is an n × m matrix and B is an m × p matrix, their matrix product AB is an n × p matrix, in which the m entries across a row of A are multiplied with the m entries down a column of B and summed to produce an entry of AB. When two linear transformations are represented by matrices, then the matrix product represents the composition of the two transformations.

**input.txt**

**Each line:**

**MATRIX\_NAME ROW COLUMN VALUE**

A 0 0 1

A 0 1 2

A 0 2 3

A 1 0 4

A 1 1 5

A 1 2 6

B 0 0 7

B 0 1 8

B 1 0 9

B 1 1 10

B 2 0 11

B 2 1 12

mapper.py

#!/usr/bin/env python3

import sys

# Set matrix dimensions

M = 2 # rows in A

K = 3 # cols in A / rows in B

N = 2 # cols in B

for line in sys.stdin:

line = line.strip()

parts = line.split()

if len(parts) != 4:

continue

matrix, row, col, value = parts

row = int(row)

col = int(col)

value = float(value)

if matrix == "A":

for j in range(N): # A[i][k] affects all C[i][j]

print("{0},{1}\tA,{2},{3}".format(row, j, col, value))

elif matrix == "B":

for i in range(M): # B[k][j] affects all C[i][j]

print("{0},{1}\tB,{2},{3}".format(i, col, row, value))

reducer.py

#!/usr/bin/env python3

import sys

from collections import defaultdict

current\_key = None

a\_vals = {}

b\_vals = {}

def emit\_result(key, a\_vals, b\_vals):

total = 0

for k in a\_vals:

total += a\_vals.get(k, 0) \* b\_vals.get(k, 0)

print("{0}\t{1}".format(key, int(total)))

for line in sys.stdin:

line = line.strip()

key, value = line.split("\t")

tag, k, val = value.split(",")

k = int(k)

val = float(val)

if key != current\_key:

if current\_key:

emit\_result(current\_key, a\_vals, b\_vals)

current\_key = key

a\_vals = {}

b\_vals = {}

if tag == 'A':

a\_vals[k] = val

else:

b\_vals[k] = val

# Emit last key

if current\_key:

emit\_result(current\_key, a\_vals, b\_vals)

**✅ Run It All Locally**

cat input.txt | python mapper.py | sort -n | python reducer.py

0,0 58

0,1 64

1,0 139

1,1 154

**AT HADOOP**

**avniet@DESKTOP-IOHA5Q9:~$** hdfs dfs -ls /kanak/input

ls: `/kanak/input': No such file or directory

**avniet@DESKTOP-IOHA5Q9:~$** hdfs dfs -ls /kanak/input

ls: `/kanak/input': No such file or directory

**avniet@DESKTOP-IOHA5Q9:~$** hdfs dfs -mkdir -p /kanak/input

**avniet@DESKTOP-IOHA5Q9:~$** hdfs dfs -put input.txt /kanak/input

**avniet@DESKTOP-IOHA5Q9:~$** hdfs dfs -ls /kanak/input

**avniet@DESKTOP-IOHA5Q9:~$** hadoop fs -ls /

Found 2 items

drwxr-xr-x - avniet supergroup 0 2025-04-16 11:07 /kanak

drwxr-xr-x - avniet supergroup 0 2025-04-16 11:09 /tmp

**avniet@DESKTOP-IOHA5Q9:~$** hdfs dfs -ls /kanak/output

Found 2 items

-rw-r--r-- 1 avniet supergroup 0 2025-04-16 11:27 /kanak/output/\_SUCCESS

-rw-r--r-- 1 avniet supergroup 30 2025-04-16 11:27 /kanak/output/part-00000

**RUN THE INPUT FILE**

hadoop jar hadoop-3.4.0/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar -input /kanak/input -output /kanak/output -mapper "python3 mapper.py" -reducer "python3 reducer.py" -file mapper.py -file reducer.py

**OUT PUT**

**avniet@DESKTOP-IOHA5Q9:~$** hdfs dfs -ls /kanak/output

Found 2 items

-rw-r--r-- 1 avniet supergroup 0 2025-04-16 11:27 /kanak/output/\_SUCCESS

-rw-r--r-- 1 avniet supergroup 30 2025-04-16 11:27 /kanak/output/part-00000

**avniet@DESKTOP-IOHA5Q9:~$** hdfs dfs -cat /kanak/output/part-00000

0,0 58

0,1 64

1,0 139

1,1 154