

**Laboratory Manual**

**Subject: Machine Learning – IV(PCCS7010T) Semester: VII Class: B. Tech Experiment No. : 1**

**Title :** Execute Matrix Multiplication using MapReduce.

**Pre-requisite:** Basics of Machine Learning

Hardware Requirements (if any): Windows 64 bit processor

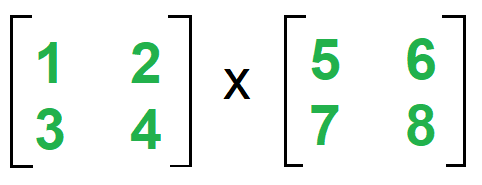
Theory/Concept Explanation (Compulsory):

Introduction:

MapReduce is a technique in which a huge program is subdivided into small tasks and run parallel to make computation faster, save time, and mostly used in distributed systems. It has 2 important parts:

* **Mapper:** It takes raw data input and organizes into key, value pairs. For example, In a dictionary, you search for the word “Data” and its associated meaning is “facts and statistics collected together for reference or analysis”. Here the Key is *Data* and the **Value** associated with is *facts and statistics collected together for reference or analysis.*
* **Reducer:** It is responsible for processing data in parallel and produce final output.

Let us consider the matrix multiplication example to visualize MapReduce. Consider the following matrix:



Here matrix A is a 2×2 matrix which means the number of rows(i)=2 and the number of columns(j)=2. Matrix B is also a 2×2 matrix where number of rows(j)=2 and number of columns(k)=2. Each cell of the matrix is labelled as Aij and Bij. Ex. element 3 in matrix A is called A21 i.e. 2nd-row 1st column. Now One step matrix multiplication has 1 mapper and 1 reducer. The Formula is:

*Mapper for Matrix A (k, v)=((i, k), (A, j, Aij)) for all k   
Mapper for Matrix B (k, v)=((i, k), (B, j, Bjk)) for all i*

Therefore computing the mapper for Matrix A:

# k, i, j computes the number of times it occurs.

# Here all are 2, therefore when k=1, i can have

# 2 values 1 & 2, each case can have 2 further

# values of j=1 and j=2. Substituting all values

# in formula

k=1 i=1 j=1 ((1, 1), (A, 1, 1))

j=2 ((1, 1), (A, 2, 2))

i=2 j=1 ((2, 1), (A, 1, 3))

j=2 ((2, 1), (A, 2, 4))

k=2 i=1 j=1 ((1, 2), (A, 1, 1))

j=2 ((1, 2), (A, 2, 2))

i=2 j=1 ((2, 2), (A, 1, 3))

j=2 ((2, 2), (A, 2, 4))

Computing the mapper for Matrix B

i=1 j=1 k=1 ((1, 1), (B, 1, 5))

k=2 ((1, 2), (B, 1, 6))

j=2 k=1 ((1, 1), (B, 2, 7))

k=2 ((1, 2), (B, 2, 8))

i=2 j=1 k=1 ((2, 1), (B, 1, 5))

k=2 ((2, 2), (B, 1, 6))

j=2 k=1 ((2, 1), (B, 2, 7))

k=2 ((2, 2), (B, 2, 8))

**The formula for Reducer is:**

*Reducer(k, v)=(i, k)=>Make sorted Alist and Blist   
(i, k) => Summation (Aij \* Bjk)) for j   
Output =>((i, k), sum)*

Therefore computing the reducer:

# We can observe from Mapper computation

# that 4 pairs are common (1, 1), (1, 2),

# (2, 1) and (2, 2)

# Make a list separate for Matrix A &

# B with adjoining values taken from

# Mapper step above:

(1, 1) =>Alist ={(A, 1, 1), (A, 2, 2)}

Blist ={(B, 1, 5), (B, 2, 7)}

Now Aij x Bjk: [(1\*5) + (2\*7)] =19 -------(i)

(1, 2) =>Alist ={(A, 1, 1), (A, 2, 2)}

Blist ={(B, 1, 6), (B, 2, 8)}

Now Aij x Bjk: [(1\*6) + (2\*8)] =22 -------(ii)

(2, 1) =>Alist ={(A, 1, 3), (A, 2, 4)}

Blist ={(B, 1, 5), (B, 2, 7)}

Now Aij x Bjk: [(3\*5) + (4\*7)] =43 -------(iii)

(2, 2) =>Alist ={(A, 1, 3), (A, 2, 4)}

Blist ={(B, 1, 6), (B, 2, 8)}

Now Aij x Bjk: [(3\*6) + (4\*8)] =50 -------(iv)

From (i), (ii), (iii) and (iv) we conclude that

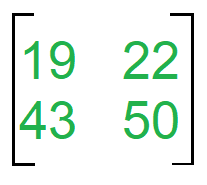
((1, 1), 19)

((1, 2), 22)

((2, 1), 43)

((2, 2), 50)

Therefore the Final Matrix is:



Implementation:

Input:

A,0,0,1

A,0,1,0

A,1,0,0

A,1,1,1

B,0,0,2

B,0,1,3

B,1,0,4

B,1,1,5

Code:

cache\_info = open("cache.txt").readlines()[0].split(",")

row\_a, col\_b, col\_a = map(int, cache\_info)

mapperOutput = open("mapperOutput.txt", "w")

for line in open("input.txt"):

matrix\_index, row, col, value = line.rstrip().split(",")

if matrix\_index == "A":

for i in range(0, col\_b):

key = row + "," + str(i)

mapperOutput.write("%s\t%s\t%s" % (key, col, value) + "\n")

else:

for j in range(0, row\_a):

key = str(j) + "," + col

mapperOutput.write("%s\t%s\t%s" % (key, row, value) + "\n")

mapperOutput.close()

numbers1 = list()

for line in open("mapperOutput.txt"):

curr\_index, index, value = line.rstrip().split("\t")

index, value = map(int, [index, value])

numbers1.append((curr\_index, index, value))

numbers2 = numbers1

initValue1 = list()

initValue2 = list()

for i in numbers1:

checker = 0

for j in numbers2:

if i == j:

if checker == 0:

checker += 1

continue

if i[0] == j[0]:

if i[1] == j[1]:

initValue1.append([i[0],str(i[1]),i[2]\*j[2]])

initValue2 = initValue1

myOut = dict()

counter = 0

for i in initValue1:

if counter > (row\_a\*col\_b\*col\_a):

break

if i[0] in myOut.keys():

counter += 1

continue

counter += 1

myOut[i[0]] = i[2]

inercounter = 0

for j in initValue2:

if i[0] == j[0]:

if i[1] != j[1]:

inercounter += 1

if inercounter > col\_b - 1:

continue

myOut[i[0]] += j[2]

for key,value in myOut.items():

print(key,value)

Output:

0,0 2

0,1 3

1,0 4

1,1 5