UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Experiment No. 2.3

Student Name: Rishav Kumar **UID: 22MCC20039**

Branch: MCA - CCD **Section/Group: 22MCD-1/ Grp A**

Semester: IV **Date of Performance: 21st Mar 2024**

Subject Name: Big Data & Analytics Lab **Subject Code: 22CAH-782**

1. Aim/Overview of the practical:

Write a Map Reduce Program that implements Matrix Multiplication.

2. Code/Steps for practical:

We assume that the input files for A and B are streams of (key,value) pairs in sparse matrix format, where each key is a pair of indices (i,j) and each value is the corresponding matrix element value. The output files for matrix C=A*B are in the same format.

We have the following input parameters:

The path of the input file or directory for matrix A. The path of
the input file or directory for matrix B.
The path of the directory for the output files for matrix C. strategy = 1, 2,
3 or 4.
R = the number of reducers.
I = the number of rows in A and C.
K = the number of columns in A and rows in B. J = the
number of columns in B and C.
IB = the number of rows per A block and C block.
KB = the number of columns per A block and rows per B block. JB = the
number of columns per B block and C block.

- **Steps:**

1. setup ()
2. var NIB = (I-1)/IB+1

3. var NKB = (K-1)/KB+1
4. var NJB = (J-1)/JB+1
5. map (key, value)
6. if from matrix A with key=(i,k) and value=a(i,k)
7. for 0 <= jb < NJB
8. emit (i/IB, k/KB, jb, 0), (i mod IB, k mod KB, a(i,k))
9. if from matrix B with key=(k,j) and value=b(k,j)
10. for 0 <= ib < NIB emit (ib, k/KB, j/JB, 1), (k mod KB, j mod JB, b(k,j))

Intermediate keys (ib, kb, jb, m) sort in increasing order first by ib, then by kb, then by jb, then by m. Note that m = 0 for A data and m = 1 for B data.

- **The partitioner maps intermediate key (ib, kb, jb, m) to a reducer r as follows:**

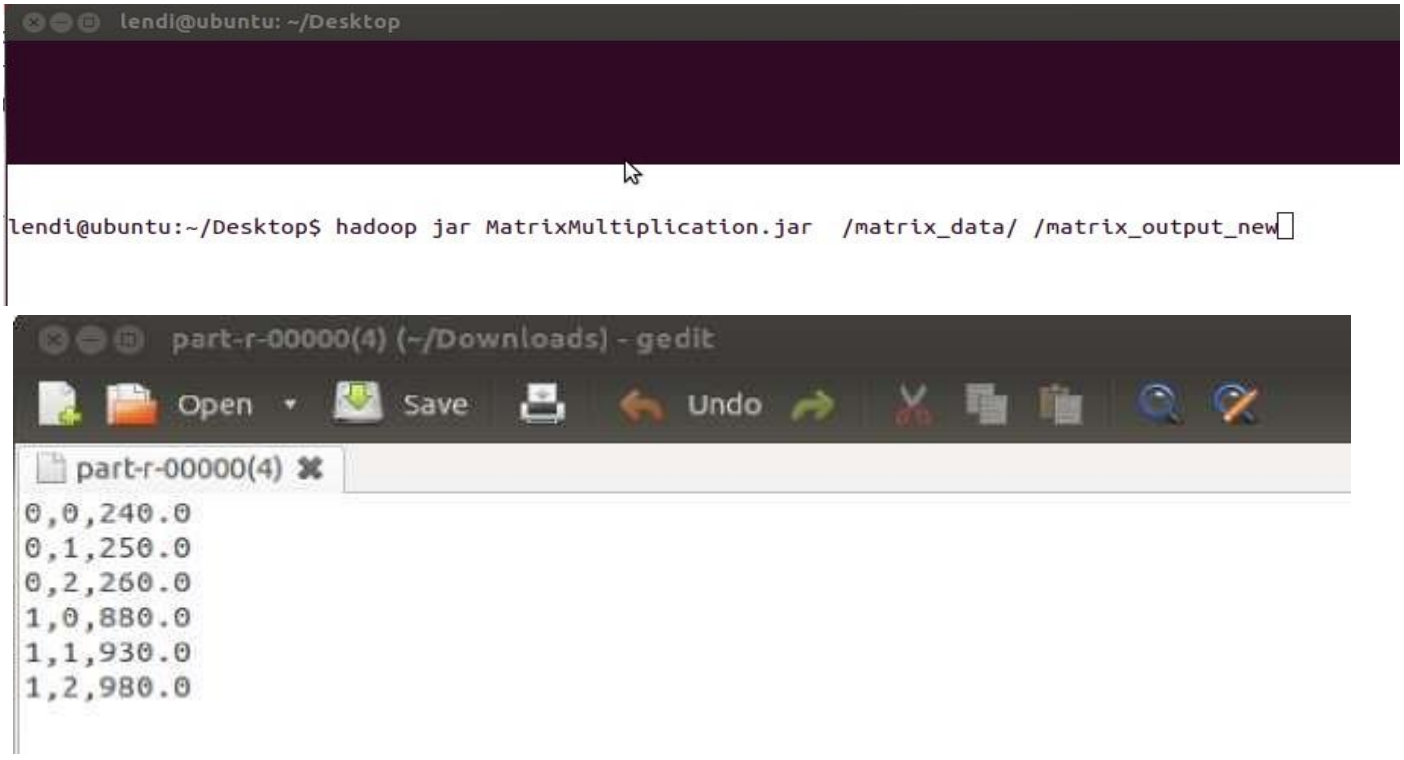
11. r = ((ib*JB + jb)*KB + kb) mod R
12. These definitions for the sorting order and partitioner guarantee that each reducer R[ib,kb,jb] receives the data it needs for blocks A[ib,kb] and B[kb,jb], with the data for the A block immediately preceding the data for the B block.
13. var A = new matrix of dimension IBxKB
14. var B = new matrix of dimension KBxJB
15. var sib = -1
16. var skb = -1

- **Reduce (key, valueList)**

17. if key is (ib, kb, jb, 0)
18. // Save the A block.
19. sib = ib
20. skb = kb
21. Zero matrix A
22. for each value = (i, k, v) in valueList A(i,k) = v
23. if key is (ib, kb, jb, 1)

24. if ib != sib or kb != skb return // A[ib,kb] must be zero!
25. // Build the B block.
26. Zero matrix B
27. for each value = (k, j, v) in valueList B(k,j) = v
28. // Multiply the blocks and emit the result.
29. ibase = ib*IB
30. jbase = jb*JB
31. for 0 <= i < row dimension of A
32. for 0 <= j < column dimension of B
33. sum = 0
34. for 0 <= k < column dimension of A = row dimension of B
    a. sum += A(i,k)*B(k,j)
35. if sum != 0 emit (ibase+i, jbase+j), sum

3. Result/Output/Writing Summary:



4. Learning outcomes (What I have learned):

Map Reduce Program that implements Matrix Multiplication.