Page :
Date : / /

Assignment NO : 2

- TITLE : Vector and Matrix operations :
- Problem Statement : Design parallel algorithm to
  i] Add two large Vectors
  ii] Multiply Vector and Matrix
  iii] Multiply two NxN arrays using n^2 processors.
- Objective :
  i] To understand Vector and Matrix operations
  ii] To implement parallel Algorithm to perform Matrix and Vector operations.
- Outcomes :
  i] Understand Vector and Matrix operations.
- Software and Hardware Requirements : A System with configuration: 4GB RAM, 500 GB HDD, i3 above CPU, GPU functionalities, CUDA framework C++ framework, Google Colab.
- Date of completion :

- Theory :

While executing the parallel algorithm of Matrix Vector Multiplication it is necessary to distribute not only the Matrix A, but also the Vector b and the result Vector c. If the processor holds the Matrix row and all the elements of Vector b and c the total number of used memory is the same order O(n).

Matrices and Matrix operations are widely used in Mathematical Modeling of Various processes, phenomenon and system. Matrix based on many scientific and engineering calculations. Computational Mathematics, physics, economics are only some of the areas of their applications. The efficiency of carrying out Matrix computation is highly important many standard libraries contain procedure for various Matrix applications.

- Add two large vectors :

when added together in this different order these same three vectors still produce a resultant with the same magnitude and direction as before. The order in which vector are added using the head to tail methods insignificant.

Vector implements a dynamic array. It is similar to Array list but with two differences. Vector is synchronized vector contains many legacy methods that are not part of

Page :
Date : / /

Collection framework :

Two add or substract 2 vectors to corresponding vectors components. Let $U \rightarrow \langle u_1, u_2 \rangle$ and $V \rightarrow (v_1, v_2)$ be two vectors. The sum of two or more vector is called the Resultant. The resultant of two vector can be found using either the parallelogram method for the triangle method.

For ex: vector1 : [10, 20, 80, 40, 50]
vector2 : [2, 3, 4, 5, 6]
Result : [12, 23, 84, 45, 56].

Multiply Vector and Matrix :

This is the same as standard Matrix Multiplication. Let Multiply the row of vector with column of Matrix which is same as normal Matrix Multiplication operations. Here we have to perform vector and Matrix multiplication where vector is a n row vector and Matrix is nxm Matric so the resultant will be a n row vector :

For ex: Matrix : $\begin{bmatrix} 2 & 4 & 3 & 1 & 2 \\ 2 & 3 & 4 & 3 & 4 \\ 4 & 3 & 1 & 3 & 1 \\ 1 & 4 & 1 & 4 & 2 \end{bmatrix}$

vector : [4 3 2 4]

Result : [26 47 30 35 30]

- Multiply two nxn arrays: It is similar to multiply two nxn Matrix where the rows of first Matrix is multiplied with columns of second Matrix and the resultant of first row and n columns is added. Similarly for all the n rows the operation is performed. For this the number of rows of first Matrix should by similar to number of columns of second Matrix.

For ex: M1: $\begin{bmatrix} 4 & 7 & 8 & 6 \\ 4 & 6 & 7 & 3 \\ 10 & 2 & 3 & 8 \\ 1 & 10 & 4 & 7 \\ 1 & 7 & 3 & 7 \end{bmatrix}$ M2: $\begin{bmatrix} 2 & 9 & 8 \\ 10 & 3 & 1 \\ 3 & 4 & 8 \\ 6 & 10 & 3 \end{bmatrix}$

5x4 4x3

$= \begin{bmatrix} 138 & 149 & 121 \\ 107 & 112 & 103 \\ 97 & 188 & 130 \\ 156 & 125 & 71 \\ 123 & 112 & 60 \end{bmatrix}$

5x3

- CUDA Programming:

Let M & N be two input Vector and P be result obtain from M & N.

P = M + N .... adding two vectors

P = M * N .... vector x Matrix

P = M * M .... array x array.

With CUDA Programming each element in P can be

obtained from one thread.

- Test cases.

Operation	Input Size	Sequential Time	Parallel Time
i] Matrix, Vector Multiplication	Vector : 1 x 100 Matrix : 100 x 50 Result : Vector : 1 x 50	0.231	0.0213
ii] Vector Addition	Vector 1: 1 x 100 Vector : 1 x 100 result : 1 x 100	0.1070	0.0179
iii] WxN array Multiplication	Matrix1 : 10 x 30 Matrix 2 : 30 x 10 Result : 10 x 10	0.039	0.01945

- Conclusion : Thus we have successfully performed the Vector and Matrix operations using CUDA programming.