Assignment No.4

Title : To add two large vectors using CUDA (Compute Unified Device Architecture), you'll need to perform the addition on the GPU. Below is a CUDA program that adds two large vectors: B) Matrix multiplication using CUDA

C++ code :

%%cu

#include <iostream> using namespace std;

// CUDA code to multiply matrices

global void multiply(int\* A, int\* B, int\* C, int size) { // Uses thread indices and block indices to compute each element int row = blockIdx.y \* blockDim.y + threadIdx.y; int col = blockIdx.x \* blockDim.x + threadIdx.x;

if (row < size && col < size) { int sum = 0;

for (int i = 0; i < size; i++) { sum += A[row \* size + i] \* B[i \* size + col];

}

C[row \* size + col] = sum;

}

}

void initialize(int\* matrix, int size) { for (int i = 0; i < size \* size; i++) { matrix[i] = rand() % 10;

}

}

void print(int\* matrix, int size) { for (int row = 0; row < size; row++) { for (int col = 0; col < size; col++) { cout << matrix[row \* size + col] << " ";

}

cout << '\n';

}

cout << '\n';

}

int main() { int\* A, \* B, \* C; int N = 2;

int blockSize = 16;

int matrixSize = N \* N;

size\_t matrixBytes = matrixSize \* sizeof(int);

1. = new int[matrixSize];
2. = new int[matrixSize];
3. = new int[matrixSize];

initialize(A, N); initialize(B, N); cout << "Matrix A: \n"; print(A, N);

cout << "Matrix B: \n"; print(B, N);

int\* X, \* Y, \* Z; // Allocate space cudaMalloc(&X, matrixBytes); cudaMalloc(&Y, matrixBytes); cudaMalloc(&Z, matrixBytes);

// Copy values from A to X

cudaMemcpy(X, A, matrixBytes, cudaMemcpyHostToDevice);

// Copy values from A to X and B to Y

cudaMemcpy(Y, B, matrixBytes, cudaMemcpyHostToDevice);

// Threads per CTA dimension

int THREADS = 2;

// Blocks per grid dimension (assumes THREADS divides N evenly)

int BLOCKS = N / THREADS;

// Use dim3 structs for block and grid dimensions dim3 threads(THREADS, THREADS);

dim3 blocks(BLOCKS, BLOCKS);

// Launch kernel

multiply<<<blocks, threads>>>(X, Y, Z, N);

cudaMemcpy(C, Z, matrixBytes, cudaMemcpyDeviceToHost);

cout << "Multiplication of matrix A and B: \n"; print(C, N);

delete[] A; delete[] B;

delete[] C;

cudaFree(X);

cudaFree(Y); cudaFree(Z);

return 0;

}

Output :