4b Vector Multiplication

#include <cuda\_runtime.h>

#include <iostream>

\_\_global\_\_ void matmul(int\* A, int\* B, int\* C, int N) {

int Row = blockIdx.y \* blockDim.y + threadIdx.y;

int Col = blockIdx.x \* blockDim.x + threadIdx.x;

if (Row < N && Col < N) {

int Pvalue = 0;

for (int k = 0; k < N; k++) {

Pvalue += A[Row \* N + k] \* B[k \* N + Col];

}

C[Row \* N + Col] = Pvalue;

}

}

int main() {

int N = 512;

int size = N \* N \* sizeof(int);

int \*A, \*B, \*C;

int \*dev\_A, \*dev\_B, \*dev\_C;

cudaMallocHost(&A, size);

cudaMallocHost(&B, size);

cudaMallocHost(&C, size);

cudaMalloc(&dev\_A, size);

cudaMalloc(&dev\_B, size);

cudaMalloc(&dev\_C, size);

// Initialize A and B

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

A[i \* N + j] = i \* N + j;

B[i \* N + j] = j \* N + i;

}

}

cudaMemcpy(dev\_A, A, size, cudaMemcpyHostToDevice);

cudaMemcpy(dev\_B, B, size, cudaMemcpyHostToDevice);

dim3 dimBlock(16, 16);

dim3 dimGrid(N / dimBlock.x, N / dimBlock.y);

matmul<<<dimGrid, dimBlock>>>(dev\_A, dev\_B, dev\_C, N);

cudaMemcpy(C, dev\_C, size, cudaMemcpyDeviceToHost);

// Print part of result

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

std::cout << C[i \* N + j] << " ";

}

std::cout << std::endl;

}

cudaFree(dev\_A);

cudaFree(dev\_B);

cudaFree(dev\_C);

cudaFreeHost(A);

cudaFreeHost(B);

cudaFreeHost(C);

return 0;

}

Open a text editor (like Gedit, VS Code, or Nano) and save each code as: matrix\_mul.cu

Command to write to run on terminal

nvcc matrix\_mul.cu -o matrix\_mul

./matrix\_mul