%%writefile hpc42.cu

#include<iostream>

#include<cuda\_runtime.h>

using namespace std;

\_\_global\_\_ void mulmax(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 sum = 0;

        for (int i = 0; i < n; i++)

        {

            //sum += A[row \* n + i] \* B[i \* n + col];

            sum += A[row \* n + i] \* B[col + n \* i];

        }

        C[row \* n + col] = sum;

    }

}

void initialize(int\* matrix, int n)

{

    for (int i = 0; i < n\*n; i++)

    {

        matrix[i] = rand() % 10;

    }

}

void print(int\* matrix, int n)

{

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < n; col++)

        {

            cout << matrix[row \* n + col] << " ";

        }

        cout << '\n';

    }

    cout << '\n';

}

int main()

{

    int n=4;

    int \*A,\*B,\*C;

    int size=n\*n\*sizeof(int);

    cudaMallocHost(&A,size);

    cudaMallocHost(&B,size);

    cudaMallocHost(&C,size);

    initialize(A, n);

    initialize(B, n);

    cout << "Matrix A: "<<endl;

    print(A, n);

    cout << "Matrix B: "<<endl;

    print(B, n);

    int \*da,\*db,\*dc;

    cudaMalloc(&da,size);

    cudaMalloc(&db,size);

    cudaMalloc(&dc,size);

    cudaMemcpy(da,A,size,cudaMemcpyHostToDevice);

    cudaMemcpy(db,B,size,cudaMemcpyHostToDevice);

    int thread=n\*n;

    dim3 dimBlock(thread, thread); // Block dimensions (16x16 threads per block)

    dim3 numBlocks((n + dimBlock.x - 1) / dimBlock.x, (n + dimBlock.y - 1) / dimBlock.y); // Grid dimensions

    mulmax<<<numBlocks,dimBlock>>>(da,db,dc,n);

    cudaMemcpy(C,dc,size,cudaMemcpyDeviceToHost);

    cout << "Addition: "<<endl;

    print(C, n);

    cudaFree(da);

    cudaFree(db);

    cudaFree(dc);

    cudaFreeHost(A);

    cudaFreeHost(B);

    cudaFreeHost(C);

    return 0;

}

Run:

!nvcc -o hpc42 hpc42.cu -Xcompiler -fopenmp

!./hpc42