//4.2 VectorAdd

#include <cstdlib>

#include <iostream>

#define checkCudaErrors(call) \

do { \

cudaError\_t err = call; \

if (err != cudaSuccess) { \

printf("CUDA error at %s %d: %s\n", \_\_FILE\_\_, \_\_LINE\_\_, cudaGetErrorString(err)); \

exit(EXIT\_FAILURE); \

} \

} while (0)

using namespace std;

// VectorAdd parallel function

\_\_global\_\_ void vectorAdd(int \*a, int \*b, int \*result, int n) {

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

if (tid < n) {

result[tid] = a[tid] + b[tid];

}

}

int main() {

int \*a, \*b, \*c;

int \*a\_dev, \*b\_dev, \*c\_dev;

int n = 1 << 4;

a = new int[n];

b = new int[n];

c = new int[n];

int \*d = new int[n];

int size = n \* sizeof(int);

checkCudaErrors(cudaMalloc(&a\_dev, size));

checkCudaErrors(cudaMalloc(&b\_dev, size));

checkCudaErrors(cudaMalloc(&c\_dev, size));

// Array initialization..You can use Randon function to assign values

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

a[i] = rand() % 1000;

b[i] = rand() % 1000;

d[i] = a[i] + b[i]; // calculating serial addition

}

cout << "Given array A is =>\n";

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

cout << a[i] << ", ";

}

cout << "\n\n";

cout << "Given array B is =>\n";

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

cout << b[i] << ", ";

}

cout << "\n\n";

cudaEvent\_t start, end;

checkCudaErrors(cudaEventCreate(&start));

checkCudaErrors(cudaEventCreate(&end));

checkCudaErrors(cudaMemcpy(a\_dev, a, size, cudaMemcpyHostToDevice));

checkCudaErrors(cudaMemcpy(b\_dev, b, size, cudaMemcpyHostToDevice));

int threads = 1024;

int blocks = (n + threads - 1) / threads;

checkCudaErrors(cudaEventRecord(start));

// Parallel addition program

vectorAdd<<<blocks, threads>>>(a\_dev, b\_dev, c\_dev, n);

checkCudaErrors(cudaEventRecord(end));

checkCudaErrors(cudaEventSynchronize(end));

float time = 0.0;

checkCudaErrors(cudaEventElapsedTime(&time, start, end));

checkCudaErrors(cudaMemcpy(c, c\_dev, size, cudaMemcpyDeviceToHost));

// Calculate the error term.

cout << "CPU sum is =>\n";

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

cout << d[i] << ", ";

}

cout << "\n\n";

cout << "GPU sum is =>\n";

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

cout << c[i] << ", ";

}

cout << "\n\n";

int error = 0;

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

error += d[i] - c[i];

if (0 != (d[i] - c[i])) {

cout << "Error at (" << i << ") => GPU: " << c[i] << ", CPU: " << d[i] << "\n";

}

}

cout << "\nError : " << error;

cout << "\nTime Elapsed: " << time;

return 0;

}