Artur Balcer 80205

Porównanie szybkości GPU i CPU używając operacji na dużych macierzach.

```
%%cu
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>
#include <time.h>

#define N (1024*1024)
#define M (10000)
#define THREADS 1024

// Funkcja mnozaca i dodajaca wartosci z macierzy a i b.
void cpu_func(double *a, double *b, double *c, int n, int m);

// Wersja GPU
__global__ void gpu_func(double *a, double *b, double *c);


int main()
{
    clock_t start, end;

    double *a, *b, *c;
    int S = N * sizeof(double);

    a = (double*) malloc(S);
    b = (double*) malloc(S);
    c = (double*) malloc(S);


    // Wypelnianie macierzy
    for(int i = 0; i < N; i++ )
    {
        a[i] = b[i] = i;
        c[i] = 0;
    }

    start = clock();
    cpu_func(a, b, c, N, M);
```

```
    end = clock();

    double czasCPU = ((double)(end-start))/CLOCKS_PER_SEC;
    printf("CPU: %lf sekund\n", czasCPU);

    start = clock();
    double *gpu_a, *gpu_b, *gpu_c;


    cudaMalloc((void **) &gpu_a, S);
    cudaMalloc((void **) &gpu_b, S);
    cudaMalloc((void **) &gpu_c, S);

    //Kopiowane danych z tablic a, b
    cudaMemcpy(gpu_a, a, S, cudaMemcpyHostToDevice);
    cudaMemcpy(gpu_b, b, S, cudaMemcpyHostToDevice);

    //Uruchomienie kernela
    gpu_func<<<(N + (THREADS-1)) / THREADS, THREADS>>>(gpu_a, gpu_b, gpu_c);

    //Kopiowanie wartosci z GPU
    cudaMemcpy(c, gpu_c, S, cudaMemcpyDeviceToHost);

    //Dealokacja pamieci
    free(a);
    free(b);
    free(c);
    cudaFree(gpu_a);
    cudaFree(gpu_b);
    cudaFree(gpu_c);

    end = clock();
    double czasGPU = ((double)(end-start))/CLOCKS_PER_SEC;
    printf("GPU: %f sekund, Przyspieszenie: %lf\n",czasGPU, czasCPU/czasGPU);

    return 0;
}


// Funkcja mnozaca i dodajaca wartosci z macierzy a i b.
void cpu_func(double *a, double *b, double *c, int n, int m)
{
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < m; j++)
        {
```

```
            c[i] = a[i] * a[i] + b[i] * b[i];
        }
    }
}

// Wersja GPU
__global__ void gpu_func(double *a, double *b, double *c)
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;

        for(int j = 0; j < M; j++)
        {
            c[i] = a[i] * a[i] + b[i] * b[i];
        }
}
```

Wynik :
CPU: 53.379714 sekund
GPU: 0.778577 sekund,
Przyspieszenie: 68.560610

W przypadku dużego problemu (operacji na sporych danych) GPU radzi sobie o wiele lepiej. Widać to na przykładzie operacji na macierzach. Występuje tam ponad sześćdziesięciokrotne przyśpieszenie w przypadku wersji GPU. Natomiast w pewnych przypadkach, gdy powiedzmy rozmiar problemu jest mały i zrównoleglenie nie jest aż tak wymagane to CPU poradzi sobie z tym lepiej, w krótszym czasie. Ciekawą analogią, o której przeczytałem i która tutaj pasuję jest: ”Piechotą łatwo pokonać szybki samochód w 1 metrowym wyścigu, ponieważ przekręcenie kluczyków, włączenie silnika i nacisnięcie pedału trochę trwa. Nie znaczy to jednak, że jestem szybszy od tego samochodu”.