proj4 – problem2

과목명	멀티코어컴퓨팅 02분반
교수명	손봉수 교수님
제출일	2023.06.09
학 과	소프트웨어대학 소프트웨어학부
작성자	20200641 임수현

Source Code

```
[ ]  %%writefile trust_ex.cu
     #include <iostream>
     #include <thrust/device_vector.h>
     #include <thrust/transform_reduce.h>

     struct integral_functor
     {
         const double delta_x;

         integral_functor(double delta_x) : delta_x(delta_x) {}

         __host__ __device__
         double operator()(double x)
         {
             double fx = 4.0 / (1.0 + x * x);
             return fx * delta_x;
         }
     };

     int main()
     {
         const int N = 1000000000;
         const double a = 0.0;
         const double b = 1.0;
         const double delta_x = (b - a) / N;

         // Create a device vector with N+1 elements
         thrust::device_vector<double> x(N + 1);

         // Fill the vector with values from a to b with a step of delta_x
         thrust::sequence(x.begin(), x.end(), a, delta_x);

         // Calculate the sum of f(x_i) * delta_x using transform_reduce
         double integral = thrust::transform_reduce(x.begin(), x.end(), integral_functor(delta_x), 0.0,

         std::cout << "Approximate integral: " << integral << std::endl;

         return 0;
     }
```

Result

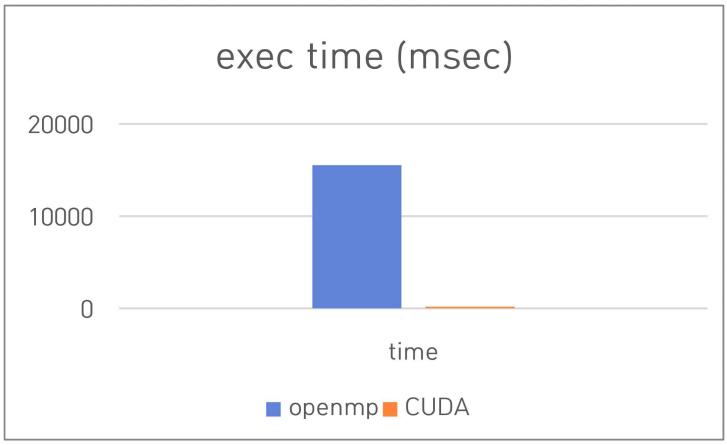
Single Thread

```
Execution Time : 15.5041619000sec
pi=3.1415926536
```

CUDA – Trust

```
1 !./a.out

Calculate time :160 msec
Approximate integral: 3.14159
```



In the case of the single-threaded calculation, it took approximately 150 seconds to compute the result. This longer duration can be attributed to the sequential nature of the computation, where each step must wait for the previous step to complete before proceeding. As a result, the overall execution time is increased.

On the other hand, when using the 'Thrust' library, the computation time significantly reduced to approximately 160 milliseconds. This remarkable improvement in performance is due to the parallel nature of the 'Thrust' library, which leverages the power of GPUs to perform computations concurrently. By distributing the workload across multiple threads and utilizing the GPU's parallel processing capabilities, 'Thrust' achieves faster execution compared to a single-threaded approach.

In conclusion, by utilizing the 'Thrust' library and harnessing the parallel processing capabilities of CUDA, the integration calculation time was significantly reduced from 150 seconds with a single thread to 160 milliseconds. This is because of the advantage of parallel computing and the effectiveness of 'Thrust' in accelerating computations on GPUs.