2025.1 Multicore Computing Project #4

-problem 2-

20223961 김수아

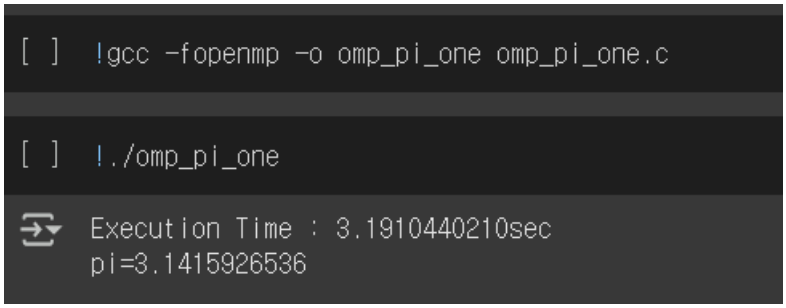
1. Envorinment
   A. Google Colab(T4 GPU)
2. Source Code
   A. thrust_ex.cu

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/transform.h>
#include <thrust/reduce.h>
#include <thrust/functional.h>
#include <thrust/iterator/counting_iterator.h>
#include <thrust/iterator/transform_iterator.h>
#include <stdio.h>
#include <time.h>
#include <cuda_runtime.h>

struct pi_functor {
    const double step;

    pi_functor(double _step) : step(_step) {}

    __host__ __device__
    double operator()(const long& i) const {
        double x = (i + 0.5) * step;
        return 4.0 / (1.0 + x * x);
    }
};

int main() {
    const long num_steps = 1000000000L;
```

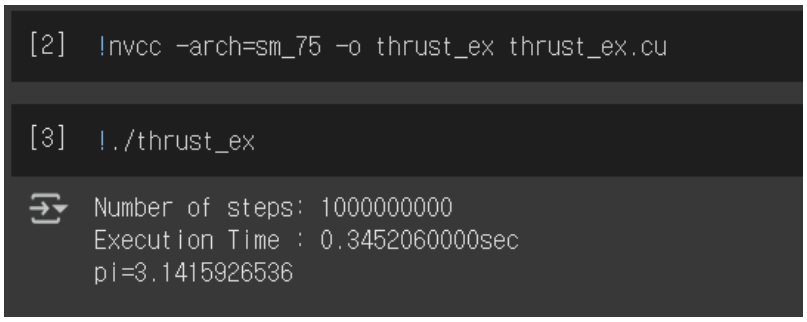
```
    const double step = 1.0 / (double)num_steps; //width

    printf("Thrust Pi Calculation Started...₩n");
    printf("Number of steps: %ld₩n", num_steps);

    // calculate execution time
    clock_t start_time = clock();
    // make counting iterator 0 ~ (num steps -1)
    thrust::counting_iterator<long> first(0);
    thrust::counting_iterator<long> last = first + num_steps;
    //make transform iterator that apply pi functor to each index
    thrust::transform_iterator<pi_functor, thrust::counting_iterator<long>>
        transform_first(first, pi_functor(step));
    thrust::transform_iterator<pi_functor, thrust::counting_iterator<long>>
        transform_last(last, pi_functor(step));
    // add all transformed values
    double sum = thrust::reduce(transform_first, transform_last, 0.0,
thrust::plus<double>());

    double pi = step * sum;

    clock_t end_time = clock();
    double elapsed_time = ((double)(end_time - start_time)) /
CLOCKS_PER_SEC;

    printf("Execution Time : %.10lfsec₩n", elapsed_time);
    printf("pi=%.10lf₩n", pi);

    return 0;
}
```

3. Compilation and Execution
   - %%writefile thrust_ex.cu
     [total code]
   - !nvcc -arch=sm_75 -o thrust_ex thrust_ex.cu
   - !./thrust_ex

4. Result
   A. capture image
      i. omp_pi_one



      ii. thrust_ex



   B. Performance

Implementation	Execution Time	Performance
1 thread (CPU)	3.191 sec	0.313
Thrust GPU	0.345 sec	2.899

   C. Result Analysis

The Thrust implementation was better than 1 thread version. The improvement was achieved because GPU can process thousands of calculations simultaneously, and Thrust libraries provides optimized parallel algorithms.