Problem2
Report

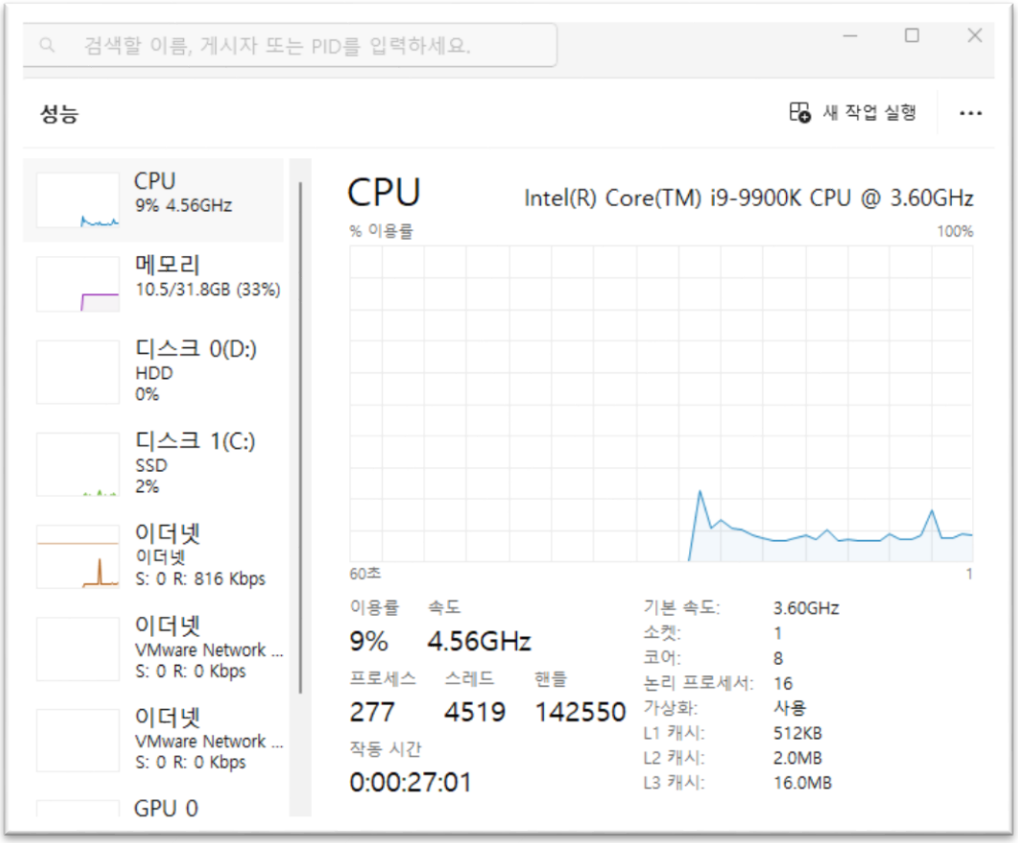
Student NO: 20183784
Student Name: 노현진

[Execution environment]

- **OS**

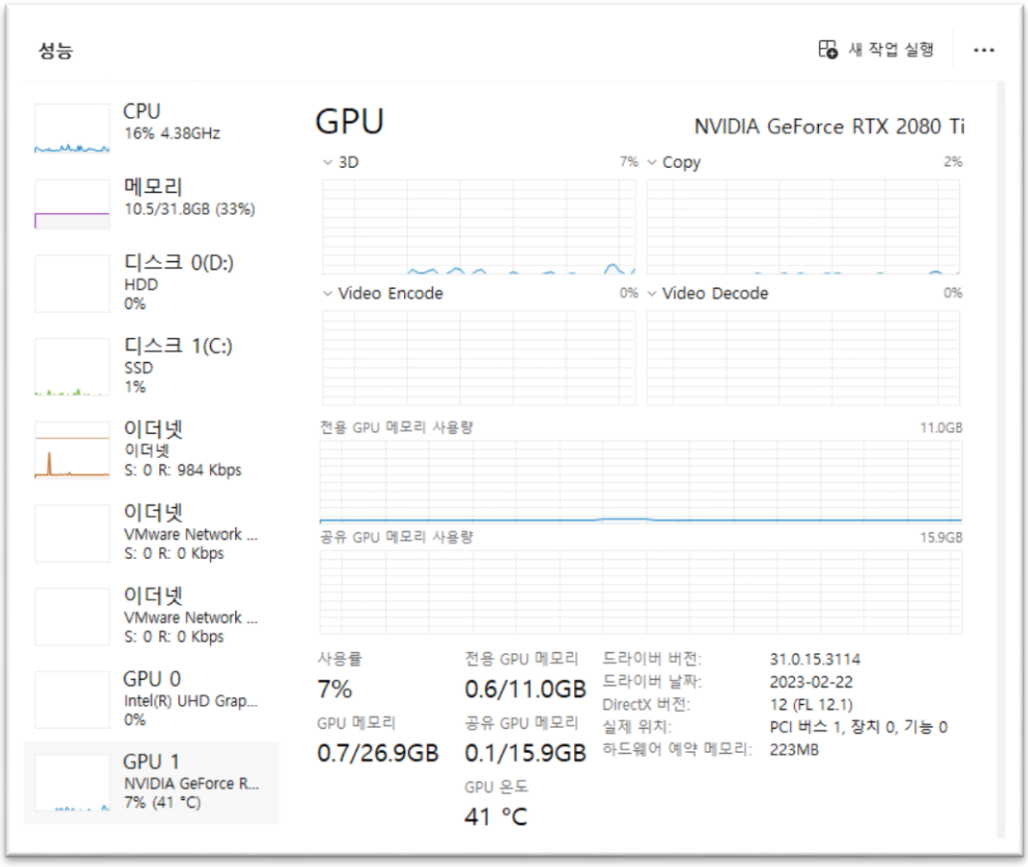
OS type: Windows 11

- **CPU**



CPU type: Intel® Core™ i9-9900K CPU

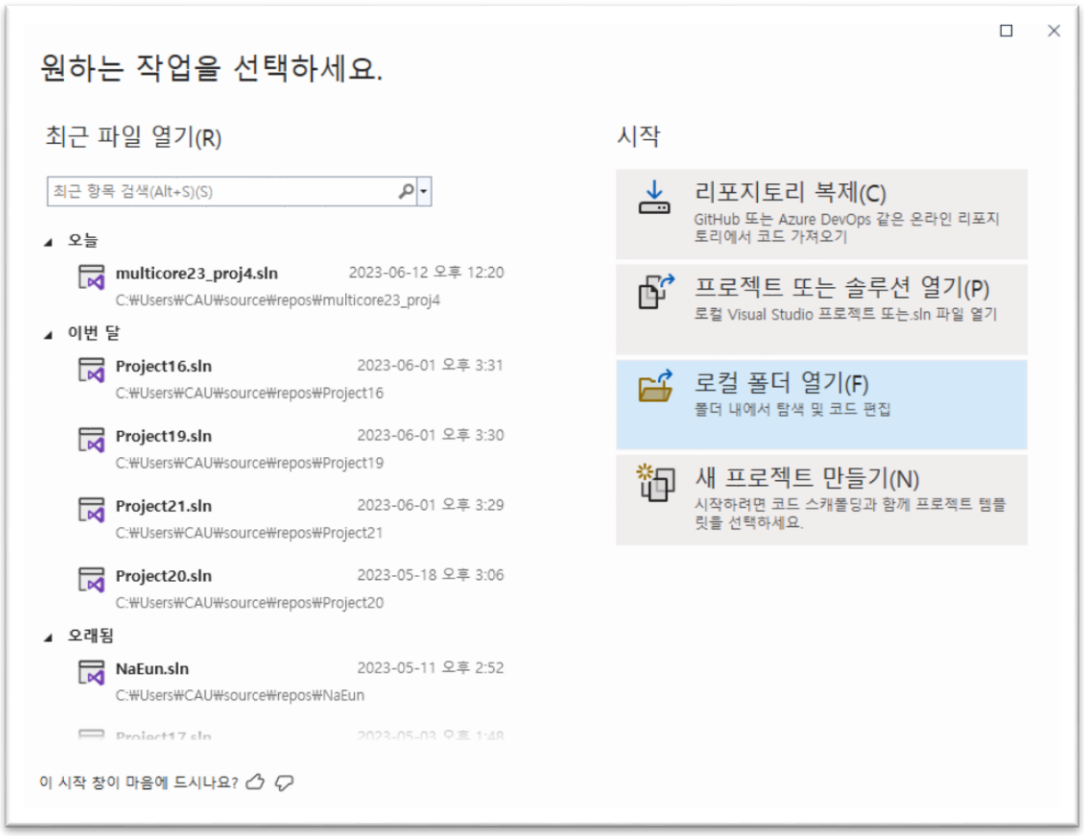
- **GPU**



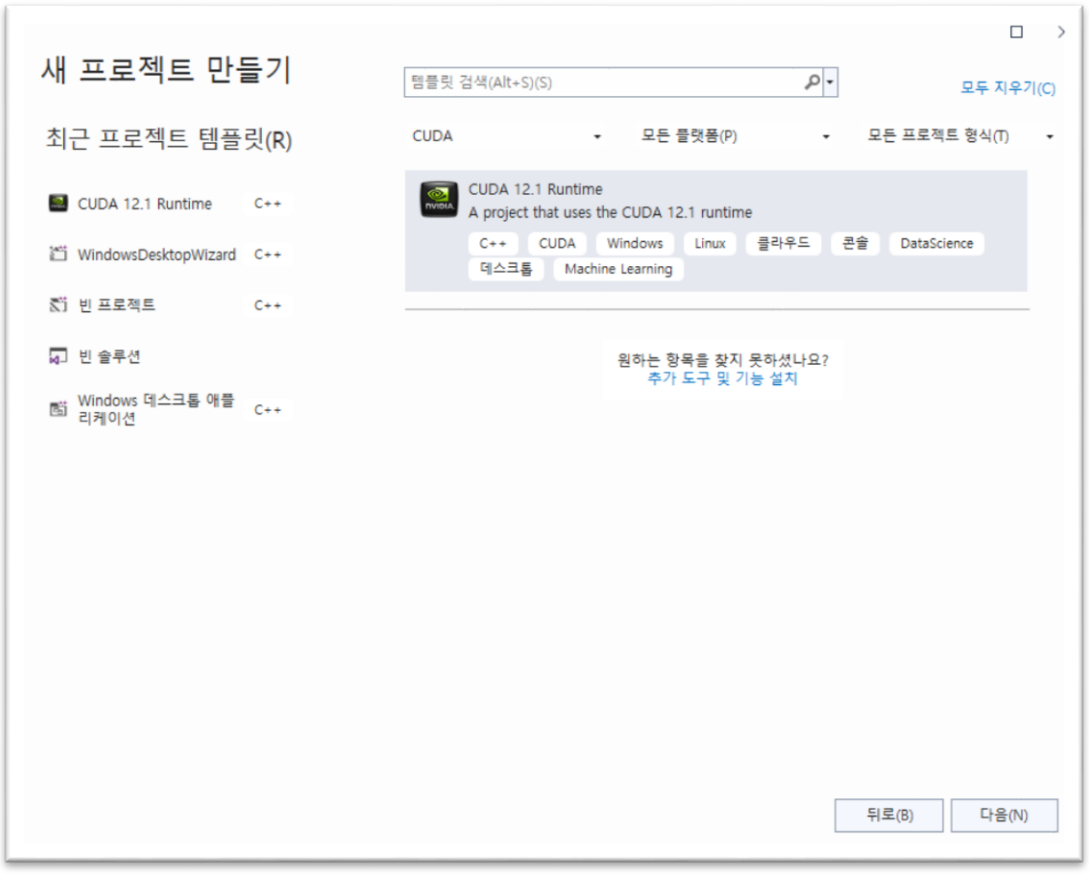
GPU type: NVIDIA GeForce RTX 2080 Ti

[How to compile and execute]

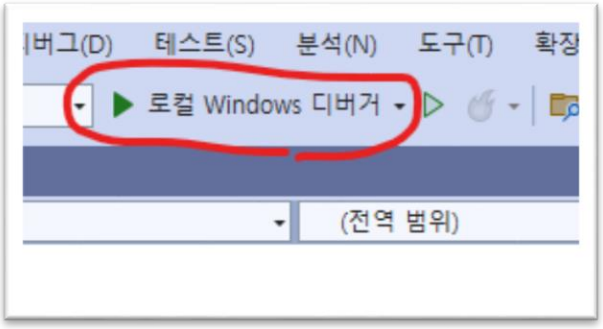
1. **Firstly, install Visual Studio 2022.**
2. **Secondly, install CUDA.**
3. **After installations, open the Visual Studio 2022.**



4. **Create new project. The project should be CUDA Run time project.**



5. **Put the source code (thrust_ex.cu) into the project.**
6. **Finally, run the source code.**



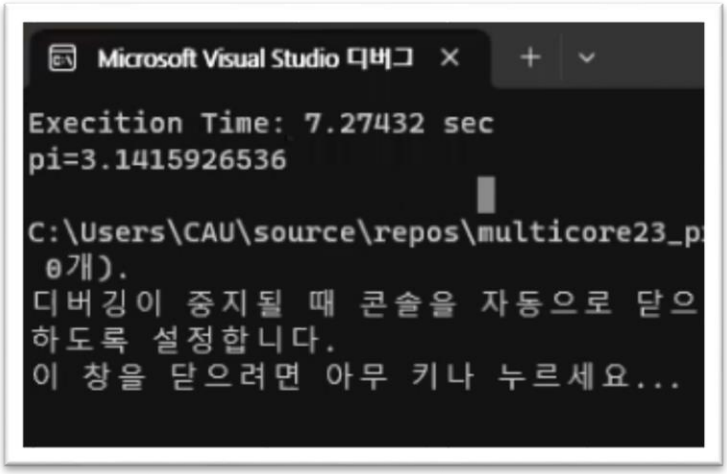
[Source Code]

- **thrust_ex.cu**

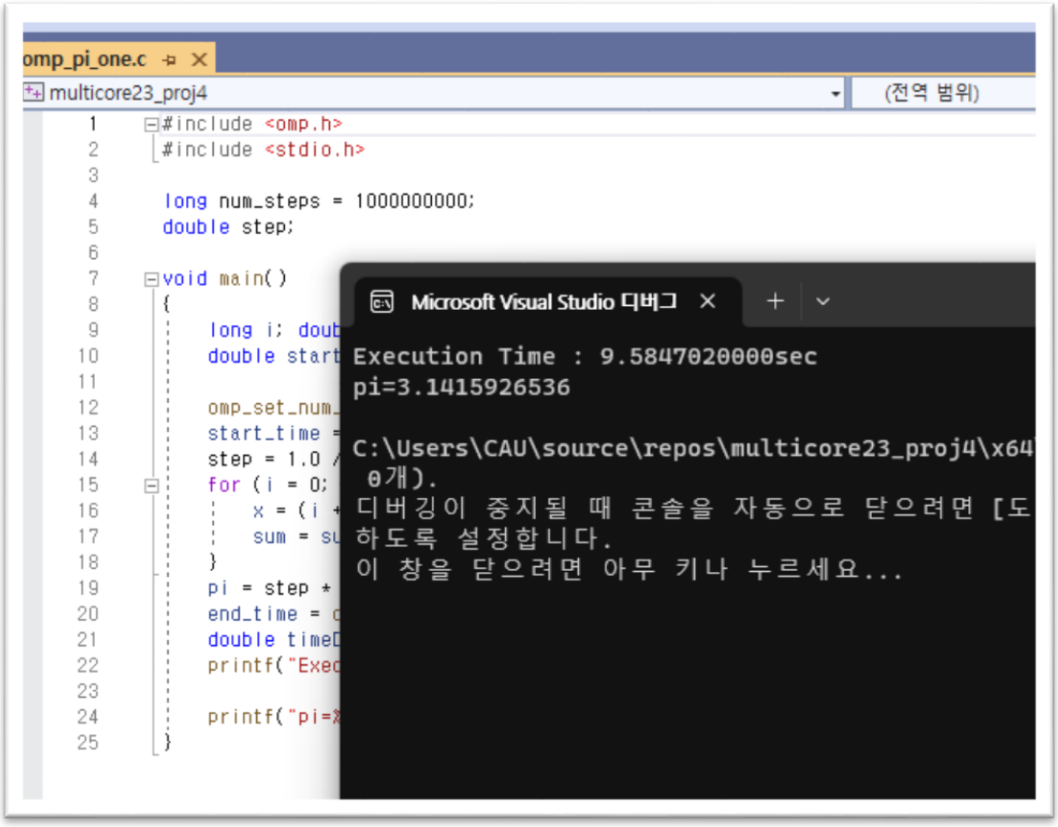
```
#include <stdio.h>
#include <iostream>
#include <chrono>
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/sequence.h>
#include <thrust/transform.h>
#include <thrust/transform_reduce.h>

using namespace std;
using namespace std::chrono;

struct functor {
    __host__ __device__
        double operator() (const double& x) const {
        return (4.0 / (1.0 + ((x + 0.5) / 1000000000) * ((x + 0.5) / 1000000000)));
    }
};

int main() {
    long num_steps = 1000000000;
    double pi = 0.0;
    double step = 1.0 / (double)num_steps;
    functor my_functor;
    thrust::plus<double> binary_op;
    double init = 0.0;

    auto start_time = high_resolution_clock::now();

    thrust::device_vector<double> X(num_steps);
    thrust::sequence(X.begin(), X.end());

    pi = thrust::transform_reduce(X.begin(), X.end(), my_functor, init, binary_op) * step;

    auto end_time = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(end_time - start_time);

    cout << "Execition Time: " << duration.count() / 1000000.0 << " sec" << endl;
    printf("pi=%.10lf₩n", pi);
    return 0;
}
```

[Results]

- **thrust_ex.cu**



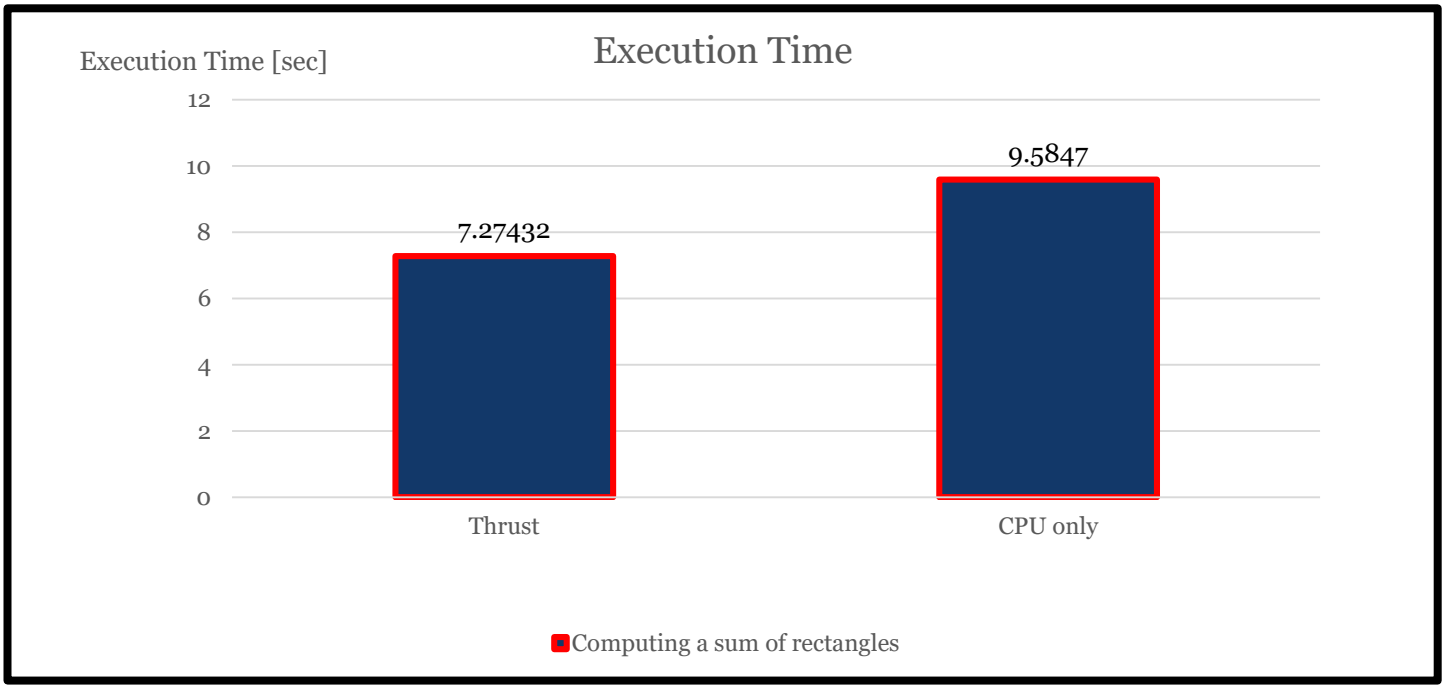
- **omp_pi_one.c**



- **Execution Time**

Thrust	
Exec time	7.27432 sec

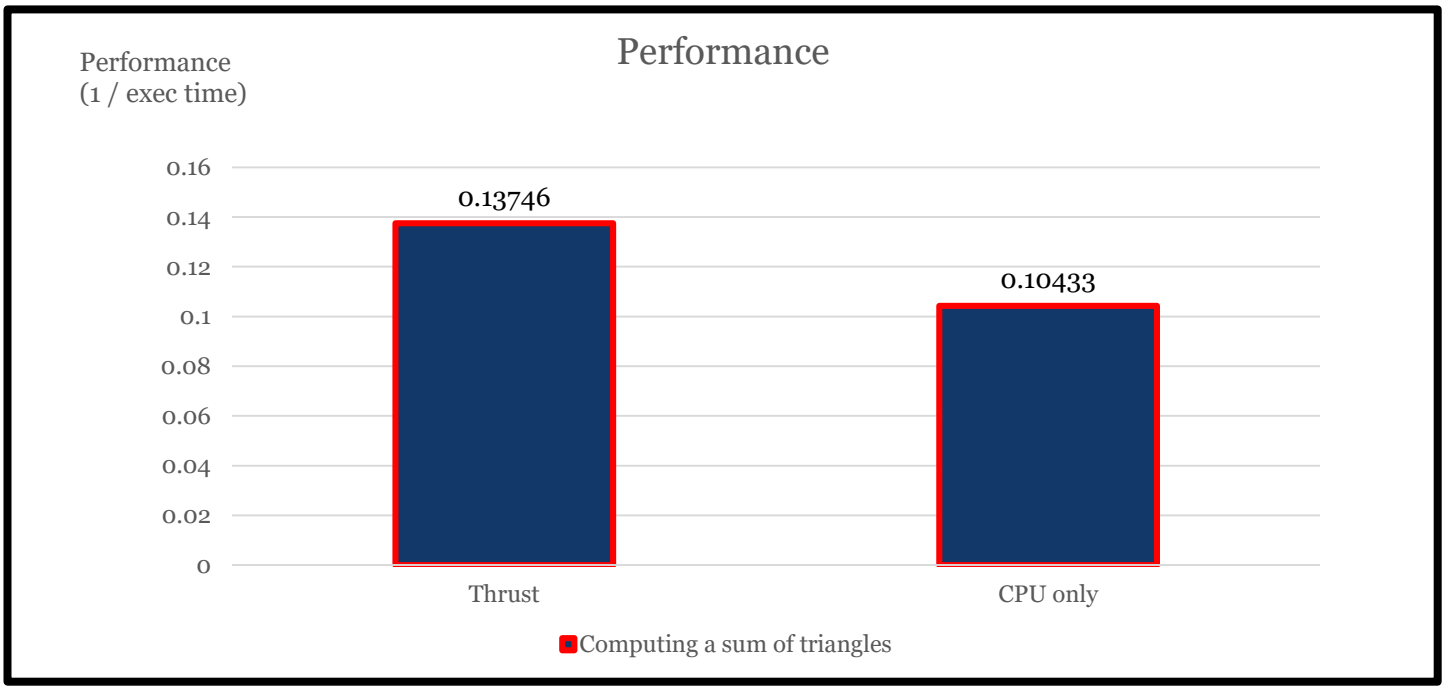
CPU only	
Exec time	9.58470 sec



- **Performance**

Thrust	
Performance (1/exec time)	0.13746

CPU only	
Performance (1/exec time)	0.10433



[Interpretation/Explanation on the Results]

The above results shows that using thrust library has better performance than CPU only. When using CPU only, it takes 9.5847 seconds. On the other hand, when using Thrust library, it takes 7.27432 seconds. It's because Thrust is a C++ template library for CUDA based on the Standard Template Library (STL) and it enhances the parallel program much faster. In other words, Thrust library is much powerful in the parallel applications than CPU.