

Problem2

Report

Student NO: 20183784
Student Name: 노현진

[Execution environment]

- OS

DESKTOP-9RV3QK4
Precision 3630 Tower

이 PC의 이름 바꾸기

i

장치 사양

복사

^

장치 이름

DESKTOP-9RV3QK4

프로세서

Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz 3.60 GHz

설치된 RAM

32.0GB(31.8GB 사용 가능)

장치 ID

F89FE21E-99F5-4AEC-9F65-030082496F71

제품 ID

00330-52958-55864-AAOEM

시스템 종류

64비트 운영 체제, x64 기반 프로세서

펜 및 터치

이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.

관련 링크

도메인 또는 작업 영역

시스템 보호

고급 시스템 설정

Windows

Windows 사양

복사

^

에디션

Windows 11 Pro

버전

22H2

설치 날짜

2023-03-08

OS 빌드

22621.1702

경험

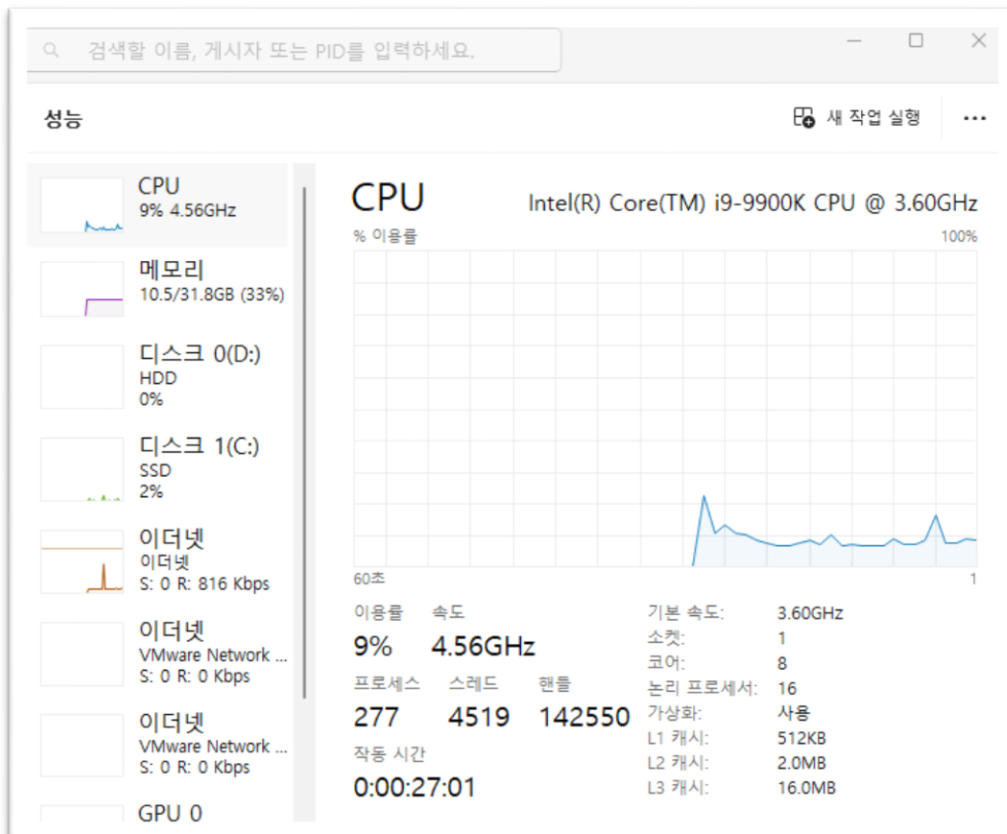
Windows Feature Experience Pack 1000.22641.1000.0

Microsoft 서비스 계약

Microsoft 소프트웨어 사용 조건

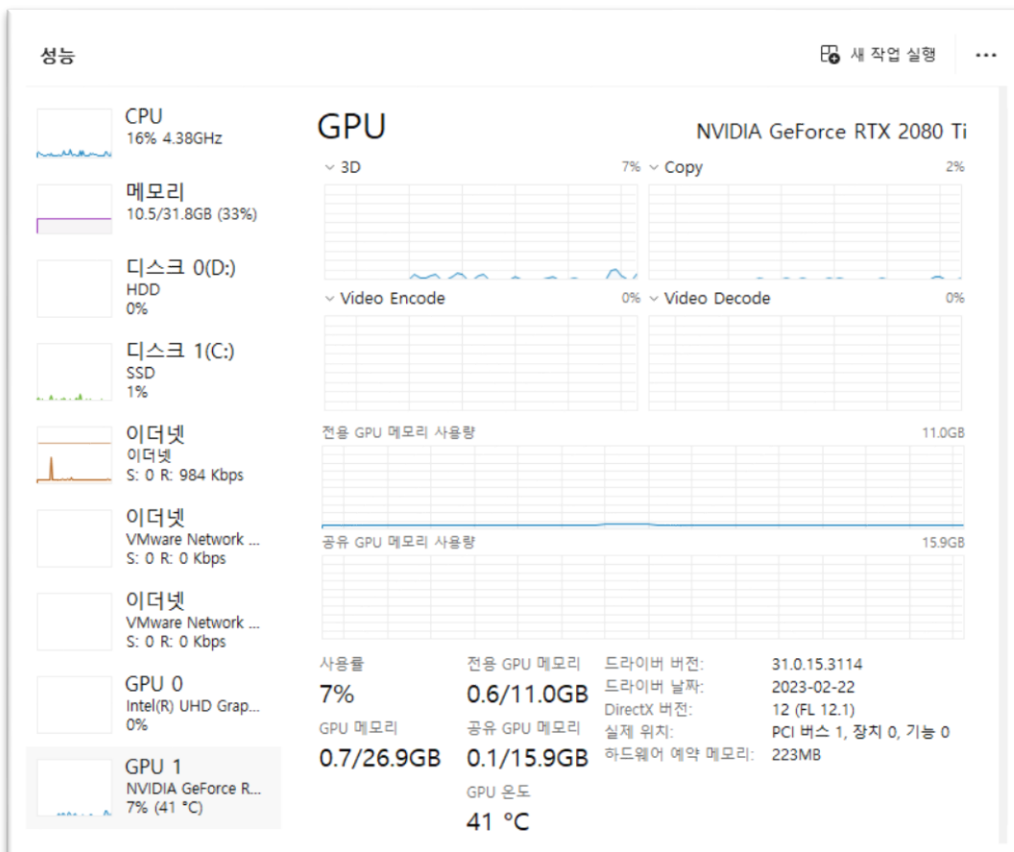
OS type: Windows 11

CPU



CPU type: Intel® Core™ i9-9900K CPU

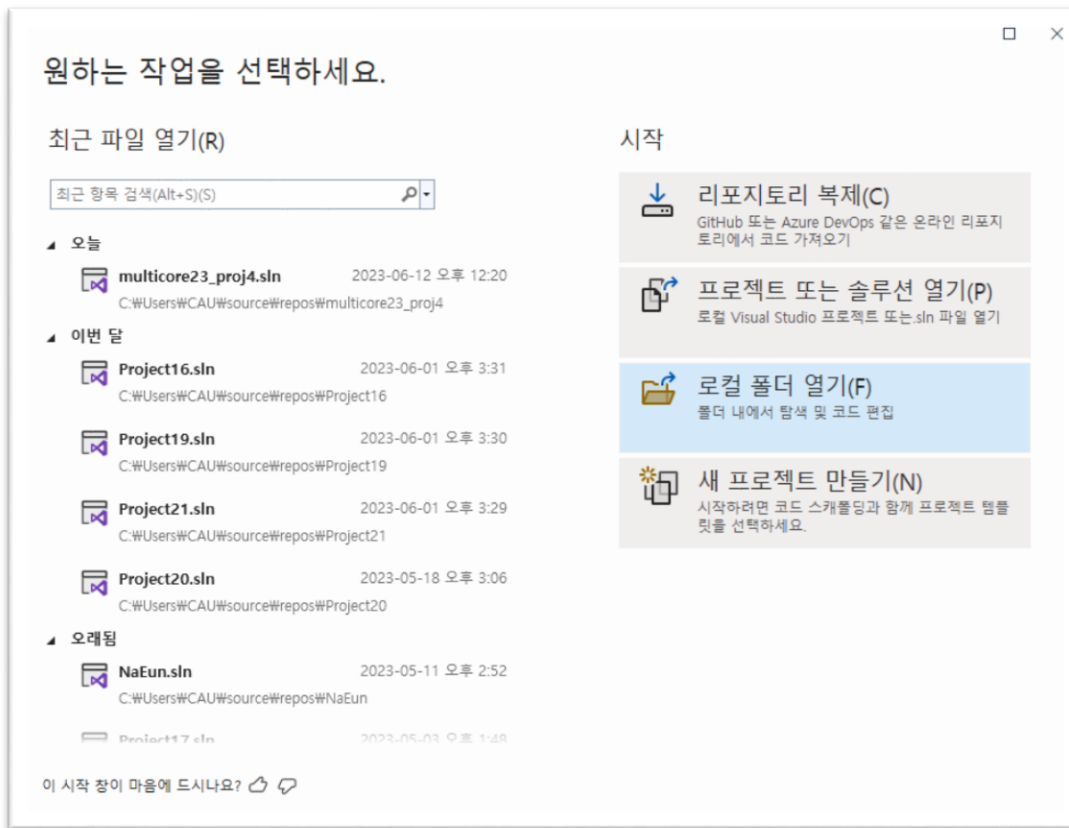
GPU



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.

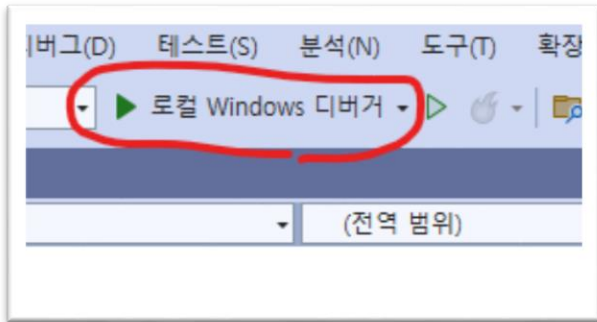


4. Create new project. The project should be CUDA Run time project.



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) / 10000000000) * ((x + 0.5) / 10000000000)));
    }
};

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());

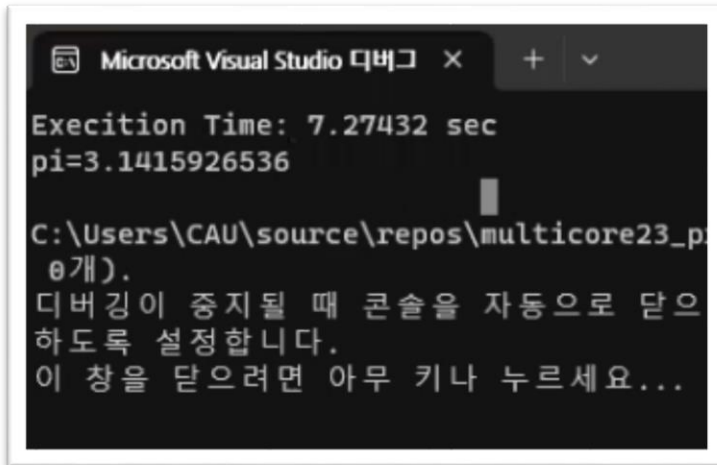
    pi = thrust::transform_reduce(X.begin(), X.end(), my_functor, init, binary_op) * step;

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

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

[Results]

- thrust_ex.cu

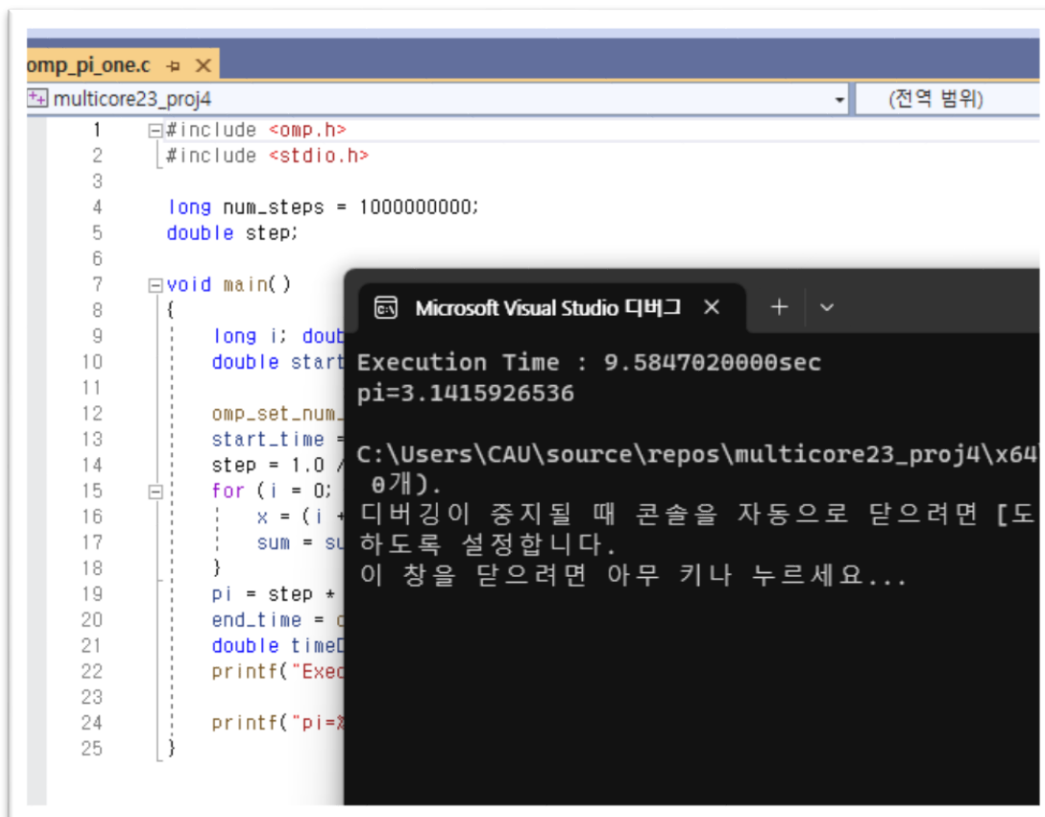


```
Microsoft Visual Studio 디버그 x + v

Execution Time: 7.27432 sec
pi=3.1415926536

C:\Users\CAU\source\repos\multicore23_p
0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

- omp_pi_one.c



```
omp_pi_one.c x
multicore23_proj4 (전역 범위)

1 #include <omp.h>
2 #include <stdio.h>
3
4 long num_steps = 1000000000;
5 double step;
6
7 void main()
8 {
9     long i; double
10    double start
11
12    omp_set_num_
13    start_time =
14    step = 1.0 /
15    for (i = 0;
16        x = (i +
17        sum = su
18    }
19    pi = step *
20    end_time = c
21    double timeD
22    printf("Exec
23
24    printf("pi=%
25 }
```

```
Microsoft Visual Studio 디버그 x + v

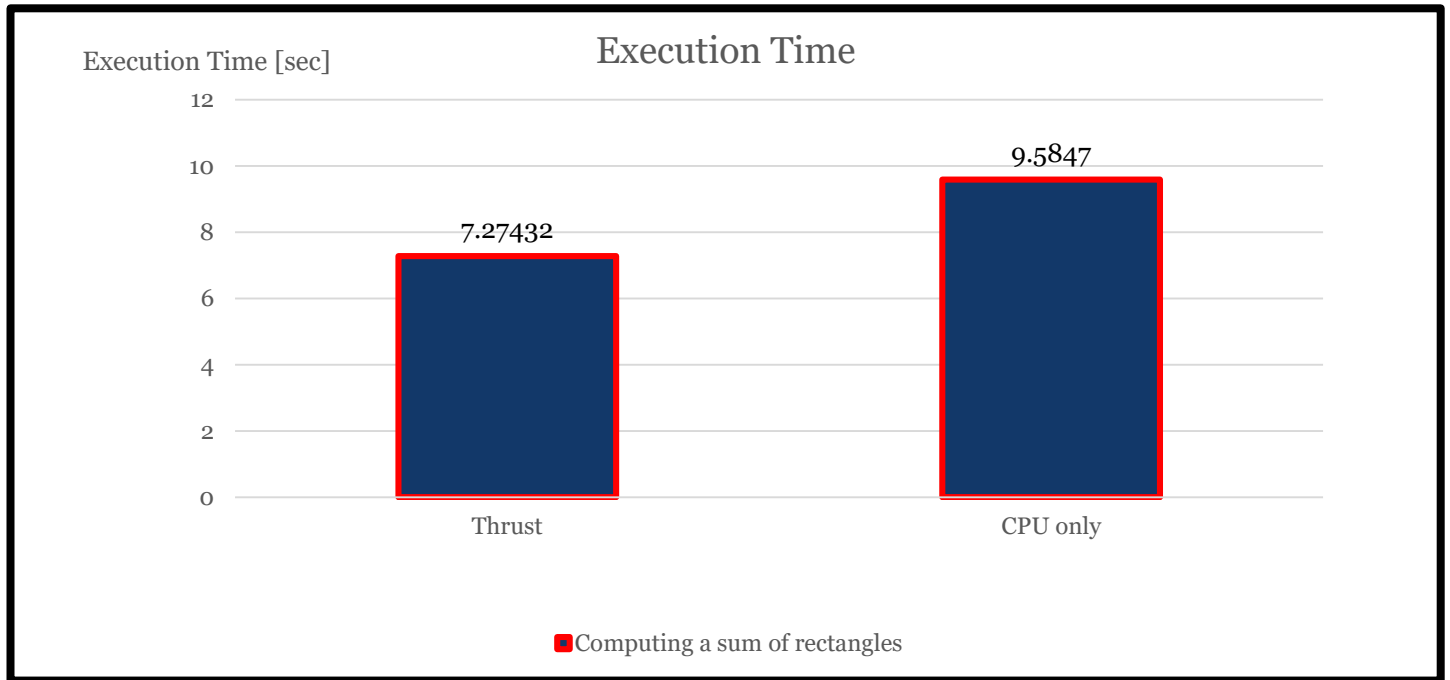
Execution Time : 9.5847020000sec
pi=3.1415926536

C:\Users\CAU\source\repos\multicore23_proj4\x64
0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

- Execution Time

Thrust	
Exec time	7.27432 sec

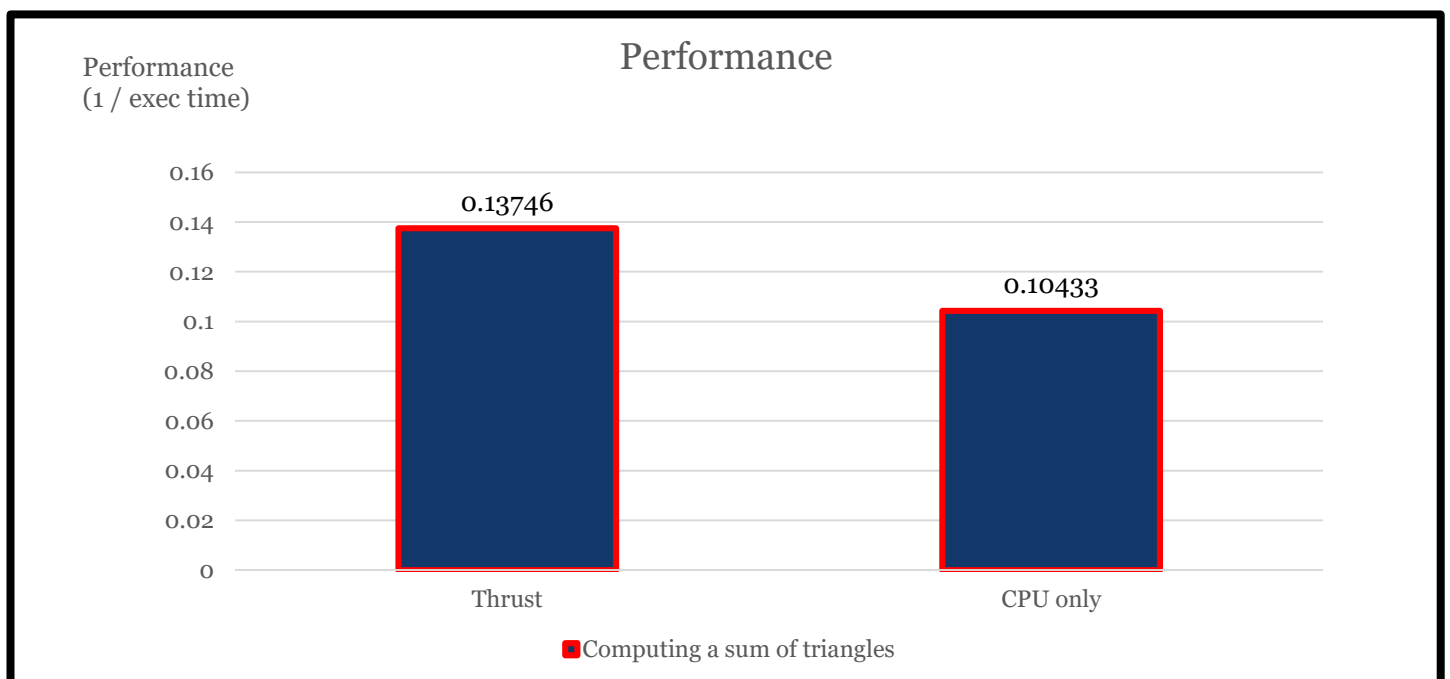
CPU only	
Exec time	9.58470 sec



- 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.