|  |  |
| --- | --- |
|  |  |
| Problem2 Report |  |
|  |  |
|  | Student NO: 20183784Student Name: 노현진 |
|  |  |

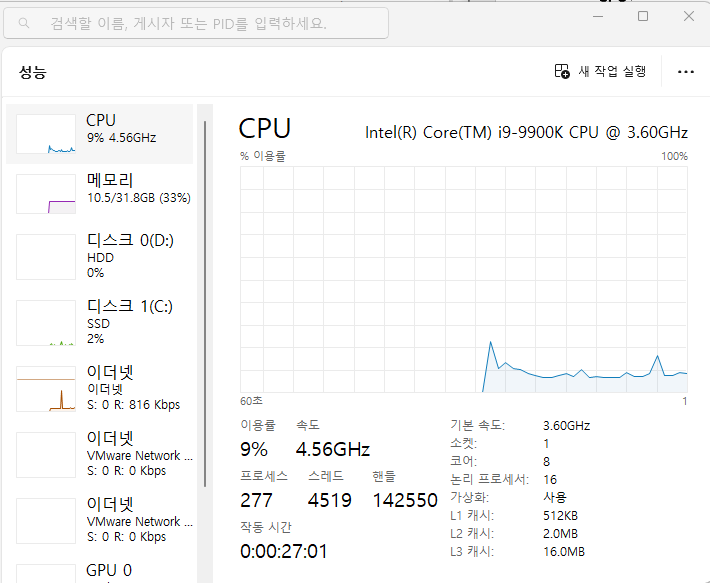
**[Execution environment]**

* **OS**



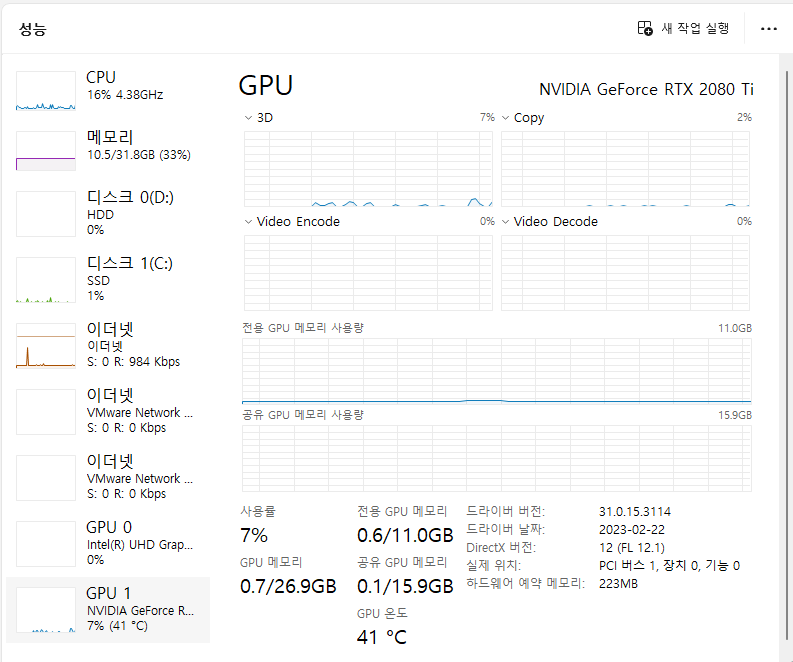
**OS type: Windows 11**

* **CPU**



**CPU type: Intel® Core™ i9-9900K CPU**

* **GPU**



**GPU type: NVIDIA GeForce RTX 2080 Ti**

**[How to compile and execute]**

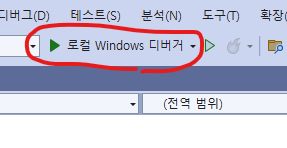
1. **Firstly, install Visual Studio 2022.**
2. **Secondly, install CUDA.**
3. **After installations, open the Visual Studio 2022.**



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

****

1. **Put the source code (thrust\_ex.cu) into the project.**
2. **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;

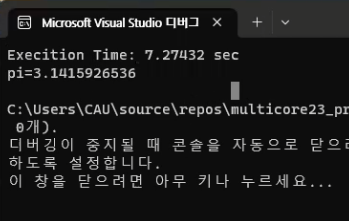
printf("pi=%.10lf\n", pi);

return 0;

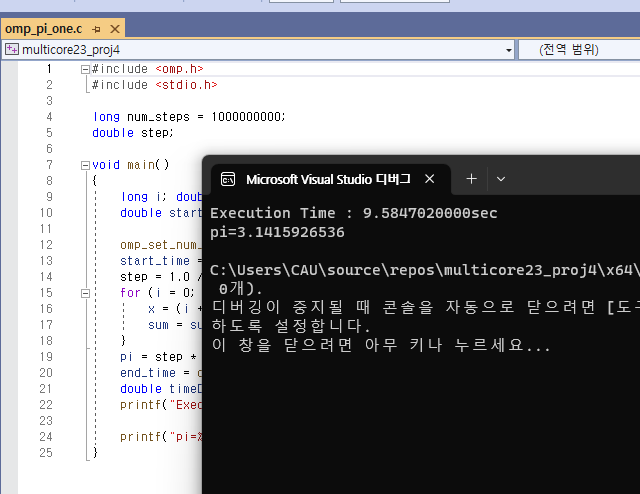
}

**[Results]**

* **thrust\_ex.cu**



* **omp\_pi\_one.c**



* **Execution Time**

|  |  |
| --- | --- |
| CPU only |  |
| Exec time | 9.58470 sec |

|  |  |
| --- | --- |
| Thrust |  |
| Exec time | 7.27432 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.