CSE 625 Parallel Programming

Project 3

By Caleb Klenda

# Machine Specifications

The project was all performed on my home computer with the following specifications:

* **CPU**

Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz

AVX2 (256-bit MM registers)

10 cores / 20 threads

20 MB Intel Smart Cache (L3-cache)

* **RAM**

32 GB DDR4 RAM

* **GPU**

TUF RTX3080 (Ampere GPU)

8704 CUDA cores  
 5 MB of L2-Cache

10GB GDDR6X

# Problem 1

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A picture containing text, gauge, device

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

# Problem 2

In the CodeBlocks project, All\_Pair\_distance, it implements four functions to   
 compute the pair-wise distance matrix of MNIST train images (loaded from  
 train-images.bin). These four methods are:

1- sequential\_all\_pairs (sequential computing)   
 2- block\_all\_pairs (C++ multi threads - block work distribution)

3- block\_ cyclic\_all\_pairs (C++ multi threads - block cyclic work distribution)

4- dynamic\_all\_pairs (C++ multi threads - dynamic work distribution)

2 .1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Matrix Size | 400 | 800 | 10,000 | 20,000 | 30,000 | 60,000 |
| Method 1 | 0.0503091 | 0.200686 | 32.9804 | 136.509 | 322.867 | 1560.27 |
| Method 2  12 threads | 0.0086379 | 0.0312908 | 6.41424 | 34.1735 | 97.6546 | 425.77 |
| Method 3  12 threads  Chunk size 2 | 0.0052788 | 0.0184834 | 2.78006 | 11.524 | 28.0694 | 112.191 |
| Method 4  12 threads  Chunk size 2 | 0.0050414 | 0.0183471 | 2.77072 | 11.5129 | 27.8863 | 111.889 |

* 1. (8 points) In the report, explain the key ideas of the function, dynamic\_all\_pairs, of its work distribution implementation and how and why the std::mutex object is used.

dynamic\_all\_pairs assign chunks to threads at runtime allowing it to adapt the the problem it is solving. global\_lower which allows access to the first row of the currently processed chunk, which, whenever a thread runs out of work, it will reference to determine what it should do next. Because multiple threads are accessing and modifying global\_lower, a mutex is needed so that only one thread can use that resource at a time

# Problem 3

Work amount (i.e., the number of outmost iterations) done by each thread for 2 threads on block\_all\_pairs for m=60,000 mxm matrix

We know that for block\_all\_pairs T(i) = i + 1. When using three threads, we split the work into the following:

So, in the general case the threads work as follows:

,

* We plug in 60,000 for m to achieve W(1)60000= 2,000,010,000
* We plug in 60,000 for m to achieve W(2)60000= 600,010,000
* We plug in 60,000 for m to achieve W(3)60000= 1,000,010,000

From this we can observe that as that W(1) accounts for 11% of the work, W(2) accounts for 33% of the work, and the W(3) accounts for the remaining 55% of the work.

# Problem 4

Work amount (i.e., the number of outmost iterations) done by each thread for 2 threads on block\_cyclic\_all\_pairs,

Using the formula given we compute the following for each thread:

* =
* =

We can see that these exactly match the results from Problem 3 so:

* We plug in 60,000 for m to achieve W(1)60000= 2,000,010,000
* We plug in 60,000 for m to achieve W(2)60000= 600,010,000
* We plug in 60,000 for m to achieve W(3)60000= 1,000,010,000