

**Parallel Computing Assignment 2**

**Report**

|  |  |  |
| --- | --- | --- |
| Name | SEC | BN |
| أحمد أسامة فوزي زهران | 1 | 6 |
| أحمد عبدالعاطي محمد | 1 | 10 |

Under Supervision of:

Eng. Mohamed Abdallah

# Part 1

# NVprof Results (in ns)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1000 x 1000 Matrix** | **1000 x 10 Matrix** | **10 x 1000 Matrix** |
| **Kernel 1** | **73250** | **2432** | **2271** |
| **Kernel 2** | **322343** | **4256** | **124226** |
| **Kernel 3** | **440362** | **214628** | **4064** |

Cases:

1. Square Matrix

* As Expected, the thread for each matrix cell (Kernel 1) is the faster due to higher parallelization.
* The Row based (Kernel 2) and Column based (Kernel 3) gave nearly the same results but still the Row based (Kernel 2) is still better because it accesses the memory sequential and not jumping like Column based (Kernel 3)

1. Tall Columns

* As Expected, the thread for each matrix cell (Kernel 1) is the faster due to higher parallelization.
* The Row based (Kernel 2) is much faster than Column Based (Kernel 3) because there is smaller element handled per thread and much more threads to handle the data.

1. Wide Rows

* As Expected, the thread for each matrix cell (Kernel 1) is the faster due to higher parallelization.
* The Column based (Kernel 3) is much faster than Row Based (Kernel 2) because there is smaller element handled per thread and much more threads to handle the data.

Part 2

Our Observation Results

* 3D kernel is called **X\*Y** times.
* 2D kernel is called **Y** times (why Y times? Because we reduce the rows which is better for memory (sequential access))
* 1D kernel is called **once** at the very end.

Part 3

How to run the exe?

EX:

* + a.exe input\_file\_name.txt output\_file\_name.txt
* Note: the params are optional (if input and output files not specified, defaults to 'input\_sample.txt' and 'solution.txt' in req1 and 'volume\_input\_sample.txt' and 'solution.txt' in req2)