Shang-Ian Tan
a1818695

# PDC Assignment 2 Report

## Objective

The objective of this analysis is to determine the minimum order of the matrix for which parallelising the trapezoid program using MPI on a four-core system yields significant efficiency gains.

## Methodology

1. **Instrumentation:** Timings of matrix multiplications were collected using MPI timing functions, strategically placed before and after the call to `**Mat_vect_mult()**`. The following are the functions used:
   i. **`MPI_Barrier()`:** Placed before, to stop the program until all processes reach this point. **(Line 81)**
   ii. **`MPI_Wtime()`:** Placed before & after, to begin and end timing respectively. **(Lines 82 & 86)**
   iii. **`MPI_Reduce()`:** Placed after, to reduce the computed local elapsed times from each process, keeping the longest time of all. **(Line 95)**

2. **Data Collection:** Timings were collected for **1**, **2**, and **4** processors across varying matrix sizes. The fastest of 3 calls to the program for each combination of processor count & matrix size was taken. This was done as program runtime could have been affected by any unrelated processes running in the background. The assumption is that the fastest time is derived from the program call which was the least affected by such unrelated processes. The **initial order of matrix** was set at **16**, and adjustments were made to find a suitable maximum size based on program response, which **ended** at **size 2048**, as efficiency began to plateau.

3. **Results Presentation:** The results are presented in tables containing **best runtime (ms)** & **efficiencies (E)** for each combination of **processor count (P)** & **order of matrix (N)**, and a line graph of the efficiencies over matrix size for each **N & P** combination.

**Table 1: Best Runtimes (ms)**

| **Processor Count (P)** | | Order of Matrix (N) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **16** | **32** | **64** | **128** | **256** | **512** | **1024** | **2048** | **4096** | **8192** |
| **P = 1** | **Best Runtime (ms)** | 0.007 | 0.010 | 0.018 | 0.049 | 0.188 | 0.789 | 3.005 | 12.713 | 49.961 | 201.807 |
| **P = 2** | | 0.012 | 0.012 | 0.017 | 0.053 | 0.106 | 0.405 | 1.633 | 6.505 | 25.650 | 101.361 |
| **P = 4** | | 0.021 | 0.018 | 0.036 | 0.109 | 0.127 | 0.311 | 1.398 | 4.457 | 13.310 | 52.250 |

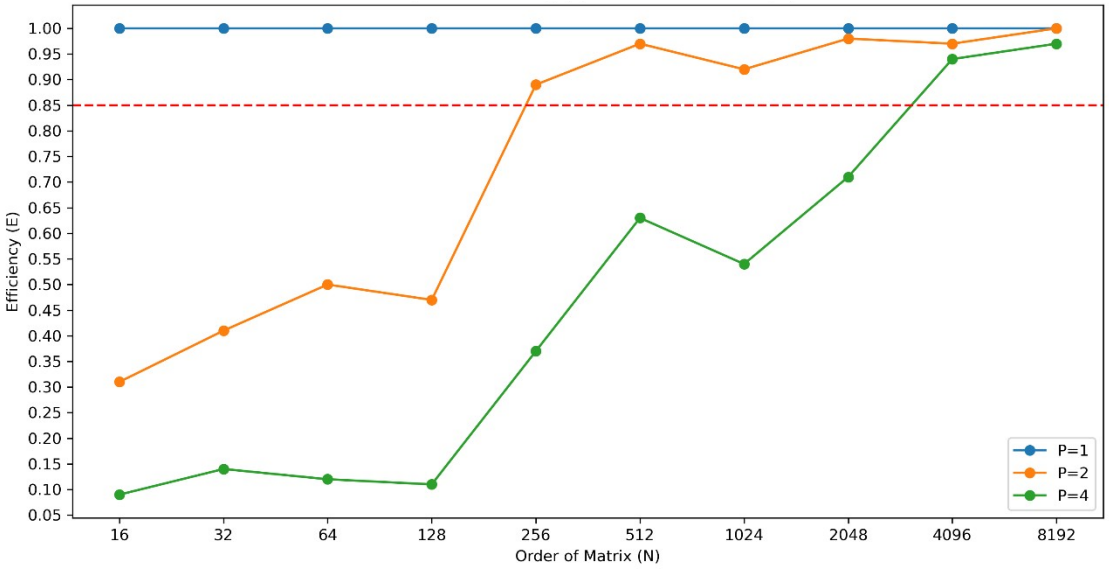
*Note: Runtimes rounded to 3 decimal points.*

**Table 2: Efficiencies (E)**

| Processor Count (P) | | Order of Matrix (N) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **16** | **32** | **64** | **128** | **256** | **512** | **1024** | **2048** | **4096** | **8192** |
| **P = 1** | **Efficiency (E)** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **P = 2** | | 0.31 | 0.41 | 0.50 | 0.47 | 0.89 | 0.97 | 0.92 | 0.98 | 0.97 | 1.00 |
| **P = 4** | | 0.09 | 0.14 | 0.12 | 0.11 | 0.37 | 0.63 | 0.54 | 0.71 | 0.94 | 0.97 |

*Note: Runtimes rounded to 2 decimal points, following assignment's definition for "worthwhile" efficiency.*

**Graph 1: Efficiency (E) vs. Order of Matrix (N)**




## Findings

1. At **P = 2**, **N > 256** and **P = 4**, **N > 4096,** it becomes worthwhile to run the program in parallel as **efficiency (E)** is greater than 85%.
2. As analysis shows an upward trend in efficiency as **Order of Matrix** grows no matter the **Processor Count**, I conclude that for every **P**, there will be a combination worthwhile to run in parallel as **N** grows sufficiently large.
3. Since USS is a **4-core system**, according to the analysis, programs should only be run in parallel on it when the **Order of Matrix (N)** is **greater than or equal to 4096**.