# MYE023: Homework #3

Due on Monday, June 12, 2017

*Vassilios V. Dimakopoulos*

**George Z. Zachos**

June 12, 2017

# Contents

**1 Exercise #1** **3**
1.1 About . . . 3
1.2 Experiment details . . . 3
1.2.1 System Specifications . . . 3
1.3 Timing Results . . . 4
1.4 Conclusion . . . 4

**2 Exercise #2** **5**
2.1 About . . . 5
2.2 Experiment details . . . 5
2.2.1 System Specifications . . . 5
2.3 Timing Results . . . 5
2.4 Conclusion . . . 6

**3 Exercise #3** **6**
3.1 About . . . 6
3.2 Experiment details . . . 6
3.2.1 System Specifications . . . 6
3.3 Timing Results . . . 6
3.4 Conclusion . . . 7

# 1 Exercise #1

## 1.1 About

This exercise is about the multiplication of integer $N$x$N$ arrays using `MPI` and strip-partitioning. During strip-partitioning, only the outermost for-loop of the serial calculation is parallelized. The purpose of this exercise is to time the matrix multiplication and observe how altering the number of `MPI` nodes[1] will affect execution time and what amount of this time is spent in communications.

## 1.2 Experiment details

During this experiment, `MPI` node number takes value in {2, 4, 8}, 4 `MPI` processes run in each node and array size is 1024x1024.

### 1.2.1 System Specifications

The experiments were conducted on a cluster of Dell OptiPlex 7020s connected via Ethernet Local Area Network (LAN):

- CPU: Intel® Core™ i5-4590 CPU @ 3.30GHz (64 bit)
- RAM: 2 DIMMs x4GiB @ 1600MHz DDR3
- Cache line size: 64B (in all levels)
- Cache associativity:
  - L1, L2: 8-way set associative
  - L3: 12-way set associative

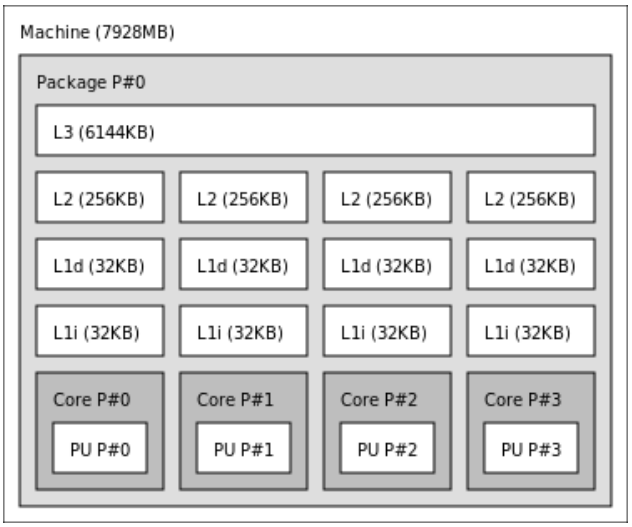


Figure 1: Topology information of a Dell OptiPlex 7020

[1] and consequently the number of `MPI` processes

## 1.3 Timing Results

In the following tables and plots the recorded execution times are displayed.

| Timing results of matrix multiplication (Time unit: seconds) Array size: 1024x1024 | | | | | | |
|---|---|---|---|---|---|---|
| # of nodes | # of processes | 1st run | 2nd run | 3rd run | 4th run | Average time |
| 2 | 8 | 1.796173 | 1.677769 | 1.640642 | 1.744993 | 1.71489425 |
| 4 | 16 | 1.281953 | 1.165645 | 1.255963 | 1.188320 | 1.22297025 |
| 8 | 32 | 1.077676 | 1.080606 | 1.104992 | 1.100065 | 1.09083475 |

Table 1: Timing results of 2D matrix multiplication

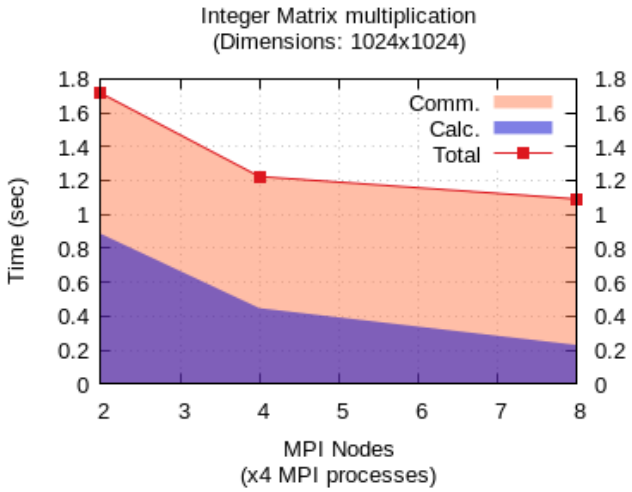


Figure 2: Timing results of 2D matrix multiplication

## 1.4 Conclusion

Based on the results presented above and given that the average execution time of the serial program is 3,79148025 seconds, we conclude that:

- Program performance is increased as the number of nodes is increased. A speedup of about 2.2 and 3.5 times is achieved in the worst and best case correspondingly.
- Even though communication overheads are introduced the time spent in calculations is halved proportionally to the number of nodes, while the time spent in communications slightly increases. For this reason it is safe to conclude that the speedup is achieved due to the significant speedup in calculation time.

# 2 Exercise #2

## 2.1 About

This exercise is about the multiplication of integer $N$x$N$ arrays using `MPI`, `OpenMP` and strip-partitioning. This hybrid programming model means to take advantage of the best features of the distributed and shared-memory programming models. `MPI` is used to distribute work between nodes while `OpenMP` is used to achieve parallelization over each node. The purpose of this exercise is the same as in Exercise #1.

## 2.2 Experiment details

During this experiment, `MPI` node number takes value in {2, 4, 8}, 1 `MPI` process & 4 `OpenMP` threads run on each node and array size is 1024x1024.

### 2.2.1 System Specifications

The experiments were conducted again on a cluster of Dell OptiPlex 7020s.

## 2.3 Timing Results

In the following tables and plots the recorded execution times are displayed.

| Timing results of matrix multiplication (Time unit: seconds) Array size: 1024x1024 | | | | | | |
|---|---|---|---|---|---|---|
| # of nodes | # of processes | 1st run | 2nd run | 3rd run | 4th run | Average time |
| 2 | 2 | 1.409800 | 1.487245 | 1.398054 | 1.433354 | 1.43211325 |
| 4 | 4 | 1.336848 | 1.319516 | 1.368628 | 1.422378 | 1.3618425 |
| 8 | 8 | 1.352629 | 1.279406 | 1.268784 | 1.254344 | 1.28879075 |

Table 2: Timing results of 2D matrix multiplication

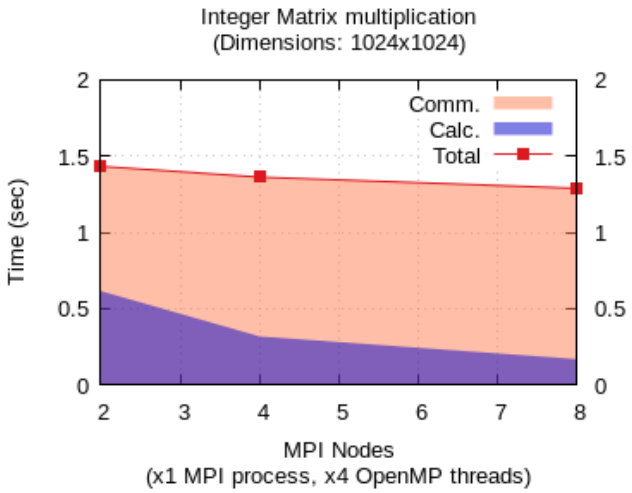


Figure 3: Timing results of 2D matrix multiplication

### 2.4 Conclusion

Based on the results presented above and given that the average execution time of the serial program is 3,79148025 seconds, we conclude that:

- Increasing the number of `MPI` nodes lead in a linear decrease of the time spent in calculations, while the time spend in communications increases and almost leads to a negligible speedup. As a result, the overall execution time remains almost constant with speedup varying between 2.5 and 3 times.

## 3 Exercise #3

### 3.1 About

This exercise is about the calculation of the mathematical constant $\pi$ using `MPI` and dynamic scheduling. During dynamic scheduling the parallelizable loops are divided into chunks of iterations (tasks) and are dispatched to the processes available for execution. The dispatch takes place in respect to the current processor workload where each process executes and as a result load balancing is achieved. The purpose of this exercise is to time the calculation of $\pi$ and observe how altering the number of `MPI` nodes will affect execution time and what amount of this time is spent in communications.

### 3.2 Experiment details

The calculation consists of $N * 10^3$ loop iterations, where N is the number of tasks provided by the user during runtime and $10^3$ is the chunksize. Finally, node number takes value in {2, 4, 8}.

#### 3.2.1 System Specifications

The experiments were conducted once again on a cluster of Dell OptiPlex 7020s.

### 3.3 Timing Results

In the following tables and plots the recorded execution times are displayed.

| Timing results of $\pi$ calculation (Time unit: seconds) | | | | | | |
|---|---|---|---|---|---|---|
| # of nodes | # of processes | 1st run | 2nd run | 3rd run | 4th run | Average time |
| 2 | 8 | 0.134720 | 0.135512 | 0.138103 | 0.133741 | 0.135519 |
| 4 | 16 | 0.107681 | 0.111158 | 0.110939 | 0.109459 | 0.10980925 |
| 8 | 32 | 0.088115 | 0.088238 | 0.087841 | 0.088177 | 0.08809275 |

Table 3: Timing results of $\pi$ calculation

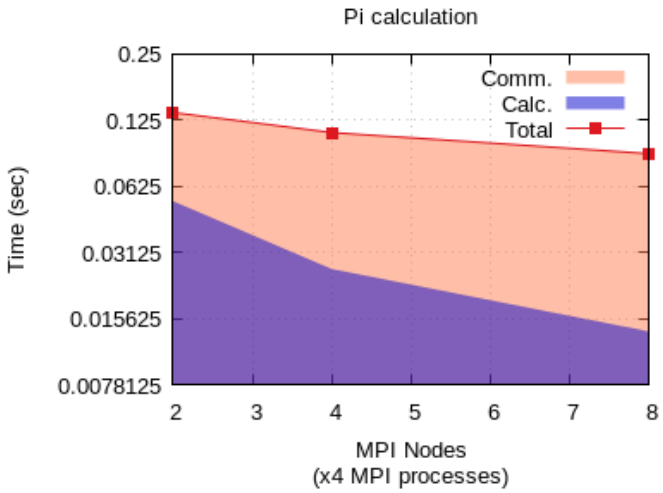


Figure 4: Timing results of $\pi$ calculation

### 3.4 Conclusion

Based on the results presented above and given that the average execution time of the serial program is 0.385222 seconds, we conclude that:

- Using `MPI` leads to execution times at least four (4) times bigger compared to the one of the serial program and keeps increasing proportionally to the number of nodes.
- Although the time spent in calculations is decreased and the speed up achieved exceeds 200 times, the huge communication overheads result in an increase of the total execution time.