MONASH University

[Template for Lab Session submission]

Applied Session Week {Week 7}

Student ID	**Student Name**	**Student Email**
32837933	Chua Sheng Xin	schu0077@student.monash .edu

Tasks

{Answers
Task 1
a) The overall wall-clock time required to search for the prime numbers strictly less than an integer 1000000 is 0.393104s when the number of processes is 1, compared to the results of the serial version which is 0.433929s. The speed up is approximately 1.104 which indicates it is only slightly better. The overall wall-clock time required to search for the prime numbers strictly less than an integer 1000000 is 0.252409s when the number of processes is 2, compared to the results of the serial version which is 0.433929s. The speed up is approximately 1.719 which indicates a significant performance gain. The maximum number of MPI processes I can run is 2 because my machine has only 2 sockets with 1 core in each socket. Due to unresolved error with Docker, I resolved to use a Linux Virtual Machine. Increasing the number of MPI processes does not always yield larger speed-ups due to factors such as overhead costs and Amdahl's Law. This is because increasing the number of MPI processes adds communication overhead and if the overhead becomes significant compared to the computation time, the speed up may stall or even decrease. Amdahl's Law on the other hand states that the maximum speed up is limited by the fraction of the program that must be executed serially. As such even if the number of process increases, the program is still required to wait for the serial part to finish, this creates a bottleneck for the speed up.

Number of MPI Processes	Wall-Clock Time	Speed Up
1	0.393104s	1.104
2	0.252409s	1.719

b) The overall wall-clock time required to search for the prime numbers strictly less than an integer 1000000 is 0.393104s when the number of processes is 1, compared to the results of the OpenMP which is 0.249489s when the number of thread is 1. The speed up is approximately 0.634 which indicates that it is much slower. The overall wall-clock time required to search for the prime numbers strictly less than an integer 1000000 is 0.252409s when the number of processes is 1, compared to the results of the OpenMP which is 0.247541s when the number of thread is 2. The speed up is approximately 0.981 which indicates that its speed is similar. The maximum number of MPI processes and threads I can run is 2 because my machine has only 2 sockets with 1 core in each socket.

Number of MPI Processes	Number of Threads	Wall-Clock Time for Open MPI	Wall-Clock Time for OpenMP	Speed Up
1	1	0.393104s	0.249489s	0.634
2	2	0.252409s	0.247541s	0.981

c) The two reasons on why the speed up may not be the same as the theoretical speed-ups are communication overhead and load imbalance. This is because in MPI, the communication between processes can create overhead when processes need to exchange large amounts of data. This overhead is not present in a serial version, and it's minimal in OpenMP due to shared memory. The theoretical speed-up assumes scaling without considering communication costs which is rarely true in practice. Moreover, if the workload is not evenly distributed among the processes or threads, some may finish earlier and remain idle while others continue processing. This imbalance reduces efficiency and leads to lower speed-ups than predicted. The two reasons on why Open MPI is lower than the OpenMP implementation in Week 3 are memory access patterns and parallelization granularity. This is because OpenMP operates within the same shared memory space which allows them to access memory directly without communication overhead. In contrast, MPI processes run in separate memory spaces that require communication to share data. This difference can make MPI slower for memory intensive tasks. Additionally, OpenMP can benefit from finer granularity in parallelization when dealing with tasks that have small and tightly coupled operations. MPI's process based model on the other hand is more suited to tasks that can be divided into large independent parts. The overhead of managing many small messages in MPI can make it slower than thread based models for some type of work.

Task 2

a) The time taken and speed-up with x number of threads on a single compute node for the OpenMP version:

Time taken for OpenMP version with 2 threads: 0.18s, speed up: 0.28/0.18s ≈ 1.56

Time taken for OpenMP version with 4 threads: 0.17s, speed up: 0.28/0.17s ≈ 1.65

Time taken for OpenMP version with 8 threads: 0.16s, speed up: 0.28/0.16s ≈ 1.75

Time taken for OpenMP version with 16 threads: 0.14s, speed up: 0.28/0.14s = 2.00

OpenMP Version

Number of Threads	Time Taken (s)	Speed-up
2	0.18	1.56
4	0.17	1.65
8	0.16	1.75
16	0.14	2.00

b) The time taken and speed up with x number of MPI processes on a single compute node for the Open MPI version:

Time taken for OpenMPI with 2 processes: 0.18s, speed-up: 0.28/0.20s ≈ 1.40

Time taken for OpenMPI with 4 processes: 0.18s, speed-up: 0.28/0.18s ≈ 1.56

Time taken for OpenMPI with 8 processes: 0.15s, speed up: 0.28/0.15s ≈ 1.87

Time taken for OpenMPI with 16 processes: 0.16s, speed up: 0.28/0.16s = 1.75

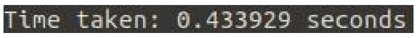
OpenMPI Version

Number of Processes	Time Taken (s)	Speed-up
2	0.20	1.40
4	0.18	1.56
8	0.15	1.87
16	0.16	1.75

CAAS does run faster on the same number of threads and processes. This is because CAAS is typically optimized for parallel computing with better hardware and lower overheads compared to a standard VM setup. This has resulted in faster execution times for both OpenMP and MPI programs. Moreover, CAAS does provide better speed up as the number of threads and processes increase. This is because CAAS has more cores (16 tested vs 2 for my machine), better memory bandwidth and lower communication overheads. The parallel efficiency of both OpenMP and MPI scales better on CAAS, leading to higher speed-ups as the number of threads and processes increase.

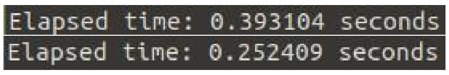
}

{Screenshots
Serial Version



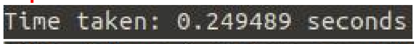
```
Time taken: 0.433929 seconds
```

Open MPI Version

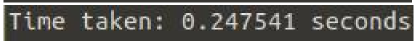


```
Elapsed time: 0.393104 seconds
Elapsed time: 0.252409 seconds
```

OpenMP Version

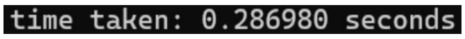


```
Time taken: 0.249489 seconds
```



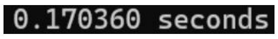
```
Time taken: 0.247541 seconds
```

CAAS Serial Version

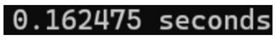


```
time taken: 0.286980 seconds
```

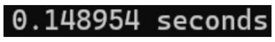
CAAS Open MPI Version



```
0.170360 seconds
```

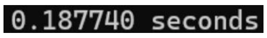


```
0.162475 seconds
```

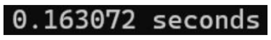


```
0.148954 seconds
```

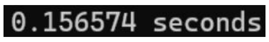
CAAS OpenMP Version



```
0.187740 seconds
```



```
0.163072 seconds
```



```
0.156574 seconds
```

}

{References
ChatGPT. I have used ChatGPT in generating answers and codes.
}