Analysis A2 of Assignment 2 DSD(COMP6281)

| Tasks | T1 | T2 | T3 |
|---|---|---|---|
| Time taken (sec) | 111.86 | 38.14 | 39.62 |

The time taken by task T1(multithreading) is 1111.86, time taken by task T2 (multiprocessing) is 38.14 and time taken by task T3(MPI) is 39.62, it can be observed from this that multithreading is taking the most time of all three tasks and the other two tasks are of almost the same time.

## Comparison between multithreading and multiprocessing

By formal definition, multithreading refers to the ability of a processor to execute multiple threads concurrently, where each thread runs a process. Whereas multiprocessing refers to the ability of a system to run multiple processors concurrently, where each processor can run one or more threads.

Multithreading is useful for IO-bound processes, such as reading files from a network or database since each thread can run the IO-bound process concurrently. Multiprocessing is useful for CPU-bound processes, such as computationally heavy tasks since it will benefit from having multiple processors; similar to how multicore computers work faster than computers with a single core.

Note that using multithreading for CPU-bound processes might slow down performance due to competing resources that ensure only one thread can execute at a time, and overhead is incurred in dealing with multiple threads.

On the other hand, multiprocessing can be used for IO-bound processes. However, overhead for managing multiple processes is higher than managing multiple threads as illustrated above. You may notice that multiprocessing might lead to higher CPU utilization due to multiple CPU cores being used by the program, which is expected.

Python Multithreaded: total execution time will be like using single thread. Let’s say first thread will count to the lower half and when it finishes the second thread will continue counting second half.

Python Multiprocessing: both threads will run in parallel, while the first thread works on lower half, second thread at the same time will work on the second half. Total processing time will be (on CPU with two threads/cores) half of multithreaded. But array is in memory and only one thread has access to. Other threads cannot access array data! The simplest way to solve it is to make a copy of data what takes time and space. Example, if user enters data using keyboard, data must be copied first into second process before being processed.

## Comparison between MPI and multiprocessing

The running time of mpi4py increases with inter-process communication. Specifically, with a fixed chunk size, as the number of processes decreases, the number of message exchanges between the master and the workers increases. But multiprocessing modules determine the chunk size depending on the data and the number of processes, which may reduce the communication overhead. In a single computer MPI and multiprocessing will work with almost the same capability but in a distributed environment MPI would work better than multiprocessing.