ENCE360 assignment
Threads and processes
Mohadesa Sharifi Msh233

Introduction

This report presents a comparative analysis of performance between multi-threading and multi-processing. The program completes an integration calculation given a number range, a number of slices, and the index of the function it should integrate over this range using the trapezoidal method. The objective is to analyze and compare the execution times of the programs as we vary the number of threads and processes and the number of queries.

Experimental Setup

Programs

We have three implementations for integrating mathematical functions:

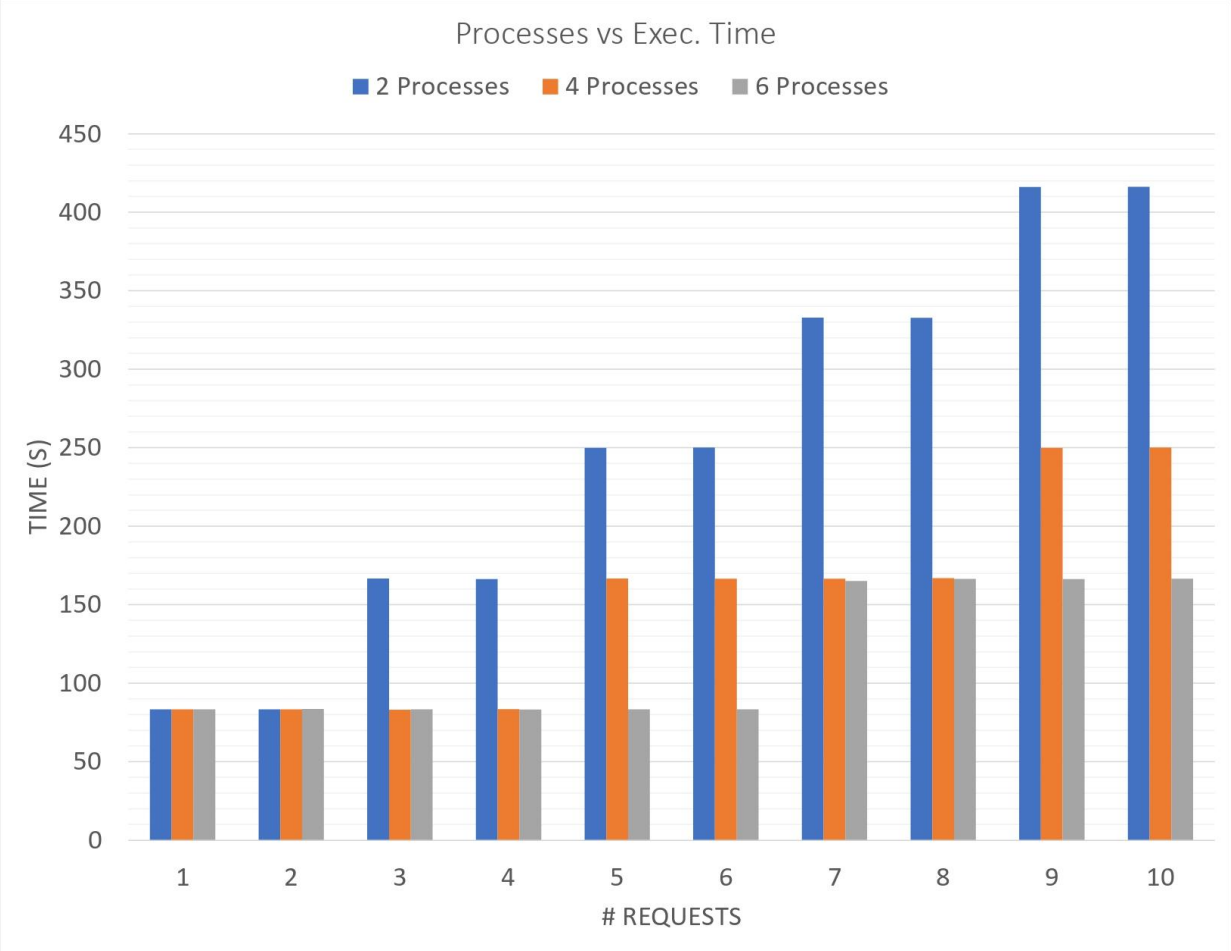
- **process.c**: This program runs the function integrateTrap() inside a child process. The parent process should then re-prompt for input. Users can have several queries running at once.
- **thread.c**: this program contains a number of threads to split the work between. It divides the work of the integration equally between a fixed number of threads by splitting up the range.
- **threadProcess.c**: This program combines multi-threading and multi-processing for concurrent integration.

Experiment Parameters

In all three programs, we varied the number of requests (lines of input) from 1 to 9 and experimented with different configurations of threads or child processes: 2, 4, and 6.

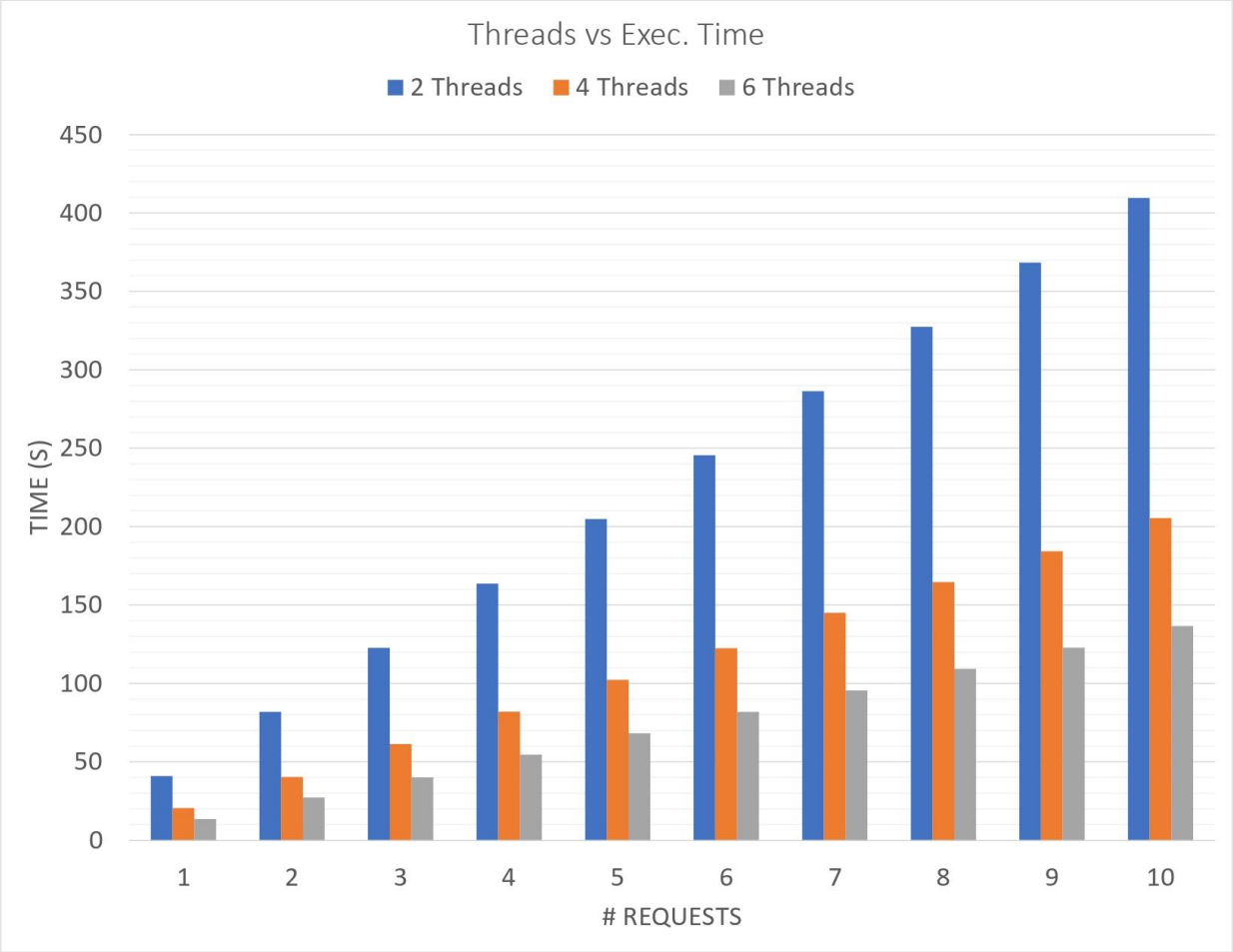
Performance comparison: Process

As the number of queries increases, the performance of the program utilizing multiple processes improves, resulting in reduced execution times. The execution time exhibits significant variations between the programs using 2 processes and those using 4 or 6 processes, particularly when the number of queries exceeds 2. Similarly, noticeable differences in execution time are observed between programs utilizing 4 processes and those utilizing 6 processes as the number of queries exceeds 4.



Performance comparison: threads

The performance of a program utilizing multiple threads remains consistent as the number of queries increases, with the average execution time remaining unchanged. However, when considering a specific query, the execution time differs significantly depending on the number of threads employed - 2 threads, 4 threads, or 6 threads. Notably, as the number of threads increases, the workload is distributed across a greater number of threads, resulting in a significant decrease in execution time.



Performance comparison: Processes and Threads

The graph below shows that as the number of processes and threads increases, the execution time experiences a significant reduction.

When we combine both multi-threaded and multi-process programs, we can take advantage of parallel processing and task division over threads. This approach enables us to handle multiple queries simultaneously, taking advantage of the efficiency of processes and the speed of calculation achieved through multiple threads. As a result, there is a significant reduction in execution time.