***NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES, KARACHI CAMPUS***



OPERATING SYSTEMS

SPRING 2023

Project Report

Section 4K

**Comparison between Process and Threads**

| **Group Members** |
| --- |
| **Sareem Farooqui 21K-3381** |
| **Makisa Imtiaz 21K-4506** |
| **Kuljeet Khatri 21K-3362** |

**OVERVIEW**

In this project, multithreaded sorting algorithms are compared and analyzed in relation to their basic sequential counterparts using C language. Through this, we find potential speedup for multithreading and what factors affect performance.

**Project Details**

Five different sorting algorithms namely quick sort, merge sort, bubble sort, insertion and selection sorts are sorting data in ascending order. A comparison is being done that compares elements in a data set with each other using a comparison operator, such as less than, more than or equal to. Merge and quick sorts are done recursively while selection, bubble and insertion sorts have an iterative approach. All algorithms have a sequential approach which is executing all instructions from start to finish without any extra assistance while a multithreaded part executes such that a process is divided into smaller sub tasks and multiple threads are distributed over multiple cores. The data sets consisted of integers (4 bytes) that ranging from 25 to 200 number of elements and elements were placed in random order in the array.

The computer specifications used for the experiment were:

* SYSTEM 1

Windows 10 &

64-bit operating system

Core i5

OS build 19044.2846

* SYSTEM 2

MacOS Ventura 13.3.1

1.4 GHz Quad-Core Intel Core i5

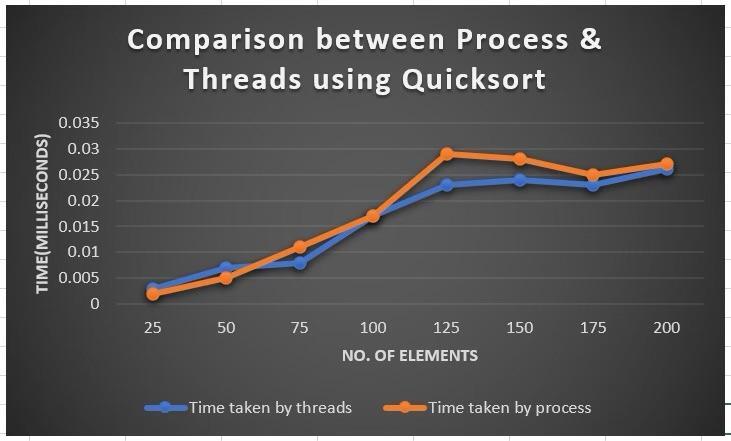
* SYSTEM 3
* Windows 10 Pro &
* 64-bit operating system
* Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz
* OS build 19044.2846

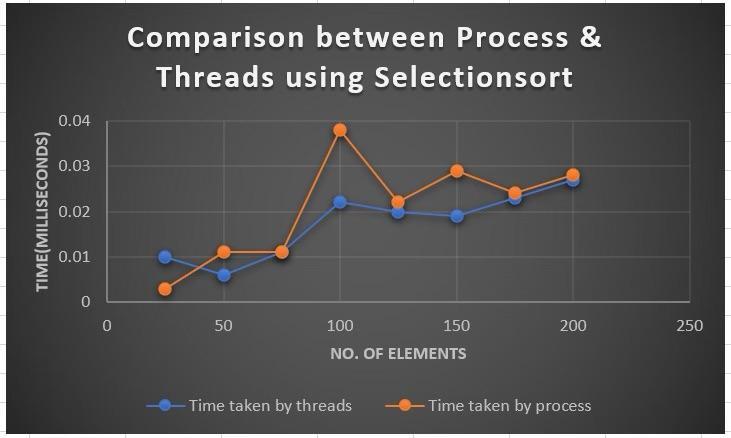
**General Workflow**

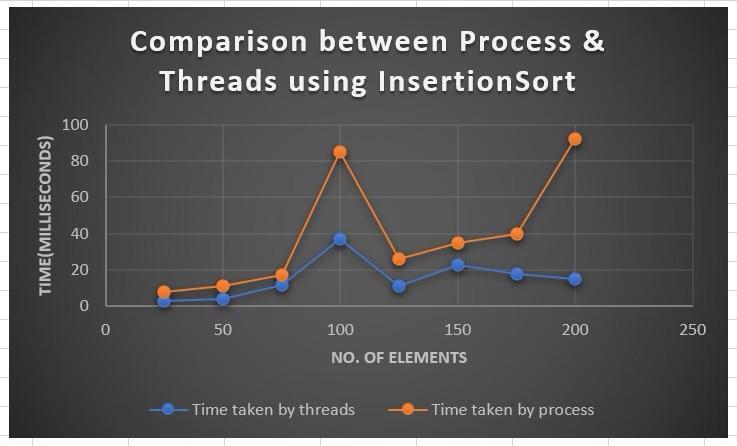
Group 2

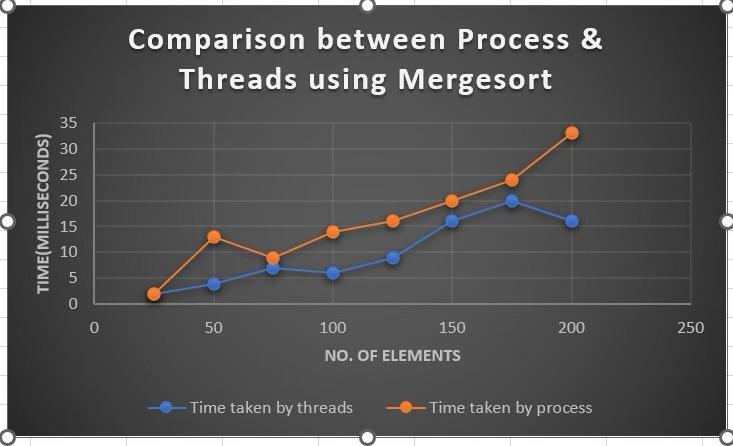
**Results and Comparison**

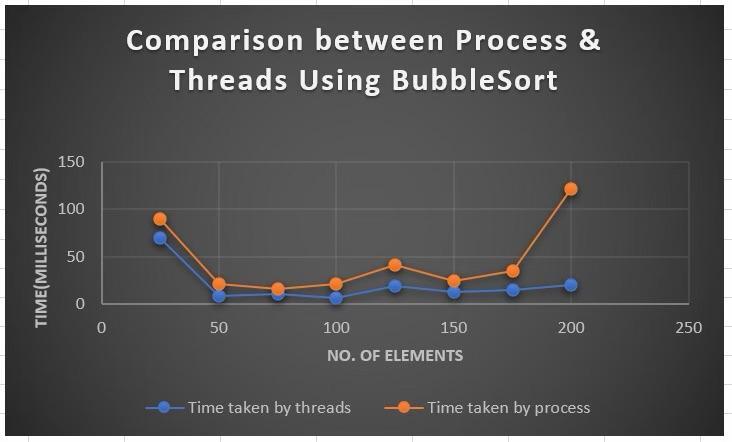
Multithreading apparently increases the efficiency however as can be seen, running the algorithm with multiple threads does not become efficient until the size of the data sets reaches 60 in merge sort and 40 in selection sort as shown in the above graphs. The max performance potential of the threads is bound to the number of physical cores available, which in the work is four cores, four threads and using more threads than physically available, virtual threads, will not increase the performance.











**Conclusion**

In this project it is shown that utilizing multithreading as a source for more computer power, indeed increases the performance of an algorithm. However, the increase starts off when the data being processed is sufficiently big enough. With exception of quick sort because of the time complexity of the algorithm, the total number of comparisons and swaps required to sort the array remains the same regardless of whether it is run with multithreading or not, most algorithms did reach a speedup of 2x when executing with four threads. A reason for not fully reaching the potential speedup might be the hardware used in the experiment. The computer on which the tests were executed had a quad-core CPU. Using all cores at once may not give full effect as other processes might be running concurrently, stealing some processing power. Had more cores been available, a 4x speedup for four threads might have been reached.