****

**Project Report**

**Topic**

“**Parallel Processing of Algorithms**”

**BY**

**Zain Abbas**

STUDENT ID:**21K-3387**

**Muhammad Haseeb**

STUDENT ID:**21K-3256**

**Muhammad Hasan**

STUDENT ID:**21K-4885**

BS CS 4E

Miss Anaum Hamid

Project Introduction

This project is about determining the most efficient programming technique out of the following three: Serial, OpenMp, and Multi-threading (pthreads). This is done by experimenting each of these techniques on the following three sorting algorithms: Insertion Sort, Selection Sort, and Cocktail Sort. The serial, obviously, falls under the sequential programming, while the other two are used to achieve Parallel Programming.

# Project Objective

In this project, we aim to discover the most efficient technique which is done by executing programs of which each contain an array, with unsorted elements, and using a timer to record the sorting time of each technique.

Technology

.

C programming language

Ubuntu 22

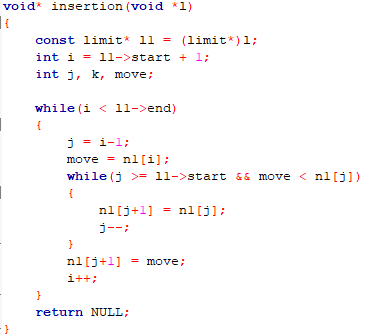
References

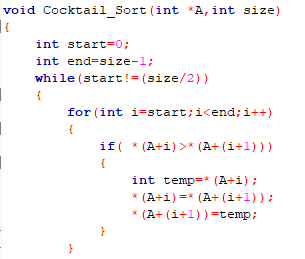
geeksforgeeks.com

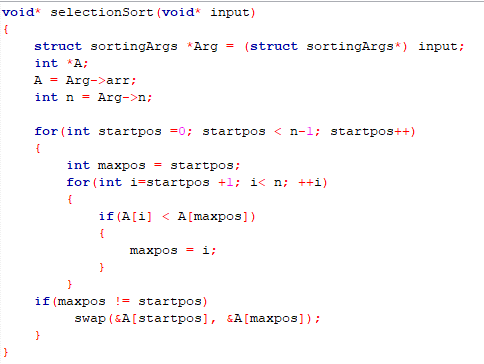
researchgate.net

stackoverflow.com

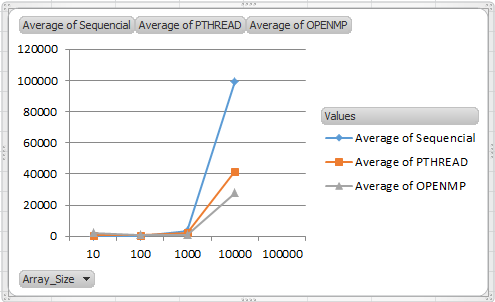
Code Snippets

Insertion Sort

Cocktail Shaker Sort

Selection Sort

Comparative Graphs

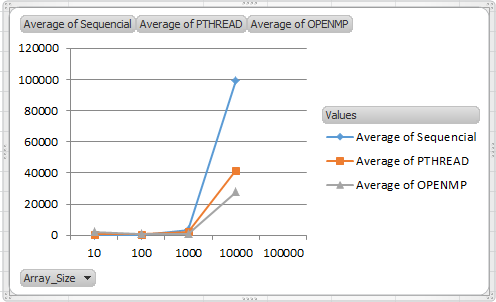
**Insertion Sort (Clock Time)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Array\_Size** | **Sequencial** | **PTHREAD** | **OPENMP** |
| 10 | 3 | 709 | 2382 |
| 100 | 58 | 598 | 817 |
| 1000 | 3622 | 2288 | 1017 |
| 10000 | 99434 | 41588 | 27861 |
| 100000 | 9248780 | 2157261 | 2216668 |

**(Burst Time)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Array\_Size** | **Serial** | **PTHREAD** | **OPENMP** |
| 10 | 7 | 647 | 2425 |
| 100 | 63 | 459 | 960 |
| 1000 | 3583 | 598 | 1017 |
| 10000 | 99172 | 78097 | 27861 |
| 100000 | 9247398 | 4290962 | 2216668 |

**Selection Sort**

**(Clock Time)**

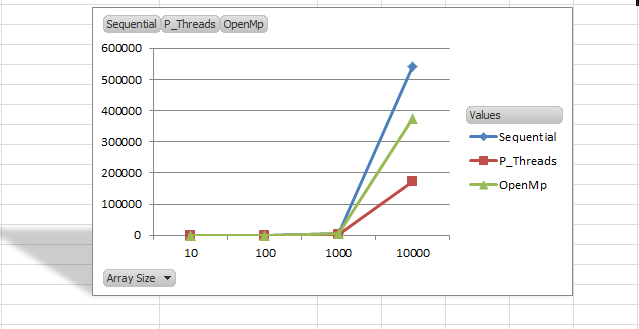
|  |  |  |  |
| --- | --- | --- | --- |
| **Array\_Size** | **Sequencial** | **PTHREAD** | **OPENMP** |
| 10 | 1 | 148 | 12 |
| 100 | 35 | 340 | 59 |
| 1000 | 2606 | 14240 | 2436 |
| 10000 | 210689 | 631466 | 194323 |
|  |  |  |  |

**(Burst Time)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Array\_Size** | **Serial** | **Pthread** | **OpenMP** |
| 10 | 5 | 105 | 16 |
| 100 | 38 | 172 | 63 |
| 1000 | 2610 | 4932 | 2443 |
| 10000 | 156670 | 557027 | 140868 |
| 100000 | 18648623 | 53594473 | 12976536 |

**Cocktail Sort**

**(Clock Time)**



|  |  |  |  |
| --- | --- | --- | --- |
| Time Taken | Time Taken | Time Taken | Array Size |
| 1 | 450 | 125 | 10 |
| 32 | 731 | 269 | 100 |
| 4546 | 1929 | 39144 | 1000 |
| 541007 | 71912 | 419953 | 10000 |
| **Sequential** | **P\_Threads** | **OpenMp** |  |

**(Burst Time)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Array\_Size** | **Serial** | **Pthread** | **OpenMp** |
| 10 | 3 | 752 | 209 |
| 100 | 34 | 699 | 619 |
| 1000 | 4553 | 3218 | 3964 |
| 10000 | 573569 | 318419 | 719846 |

Conclusion

After executing the programs, the sorting times of each technique implemented on a sorting algorithm is compared. The technique which produces the lowest time is considered as the best technique, and the one which produces the highest time is considered as the worst. This is done for all three sorting algorithms

The above results show that the primary factor of efficiency is the size of the data. The smaller the data the better sequential performs. On the other hand, the larger the data the better PTHREAD and OPENMP performs. The second factor is the CPU burst time; the results show that depending on the algorithms, parallel programming tend to be faster but on the other hand, creation of multiple threads or processes may need more CPU time as compared to the sequential programming.

Limitation

Maximum no of sorting algorithms could not be used due to time constraints, which could have provided the better conclusion about which technique is less time consuming on which data set and on which algorithm.