****

**OPERATING SYSTEMS PROJECT REPORT**

**Parallel Programming - Comparison of sorting Algorithms using Pthreads vs. OpenMP**

# Group Members

Huzaifa Abid 17K-3807

Muhammad Mustafa Manga 17K-3795

Danish Abdullah 17K-3720

# Introduction

Parallel programming is a type of computation in which many calculations or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time to obtain the result much quicker. There are many ways to achieve parallel programming but we will be discussing two of them right now i.e. Pthreads and OpenMP.

Sorting algorithms are of different types; some use iterations which are dependent on each other (bubble sort, insertion sort etc.) hence threading would not be of much use because threads will have to wait for other threads to provide their results to proceed, which will eventually result in a sequential programming result. However some algorithms (merge sort and quick sort) have iterations which are independent of each other, therefore; threading will be useful to increase their performance by reducing the processing time.

Pthreads and OpenMP both have their pros and cons, sometimes Pthreads are better with an algorithm and other times OpenMP is better. This is what we will be finding out that which threading method is more suitable with the sorting algorithms.

# How We Started

Initially, we generated random numbers (website reference given at the end), data sets of different sizes i.e. 100, 30,000, 100,000, 1,000,000. We implemented merge sort, quick sort and Radix sort using the sequential method, Pthreads and OpenMP. Time was calculated and compared to see which algorithm was faster through which method.

# Problems Faced

First problem we faced was the random number generation. Random number generation through programming was taking much more times as data sets were in thousands and millions. Second problem was the limit of threads in a process. Ubuntu and other distributions (Mint, Parrot, Kali and Cent) limited the thread count in a process to around 32,755. Therefore, data

# Solution to Problems

Random number problem was solved by using a website which produced random numbers according to our requirements. Data sets were generated at stored in different files. Thread limit problem was solved by applying data parallelism. Data was divided into 4 equal parts for merge and quick sort whereas for Radix sort it was divided into 2 equal parts. Each division of data was sent to different threads which implemented the respective sorting algorithm. At the end all the divisions were merged and stored in a file.

# Methodology

The sorting algorithms the program tested are as follows:

* Merge sort
* Quick sort
* Radix sort

First, all three algorithms are implemented using sequential approach, Pthreads and OpenMP. Data parallelism approach is used in all three algorithms. Threading decreases their processing time immensely but the question is which threading method is better? The graphs will show the output and compare the results of all algorithms.

The program runs multiple data sets on the algorithms using sequential approach and both threading methods and compares their time complexities. Data sets are of different types, which are as follows:

* Small data (100 values)
* Large data (100,000 and 1,000,000 values)
* Already sorted data (30,000 values)
* Reverse sorted data (30,000 values)

Time complexity is calculated by computing the time taken by the algorithm to completely sort the data and is displayed in milliseconds.

Results are compared to check which threading method works better with which algorithm.

# Graphs (Linux)

# Graphs (Windows)

# References

<https://www.youtube.com/playlist?list=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEG>

<https://www.openmp.org/>

<http://man7.org/linux/man-pages/man7/pthreads.7.html>

<https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html>

<https://www.openmp.org/wp-content/uploads/OpenMPRef-5.0-111802-web.pdf>

<https://numbergenerator.org/randomnumbers>

<https://linux.die.net/man/2/chdir>