PARALLEL PROGRAMMING

Comparison of Sorting Algorithm using Pthreads vs OpenMP

**BY:**

Munir Hassan (16K-3761)

Basit Ali Khatri (16K-3762)

Muhammad Bilal Khan (16K-3778)

INSTRUCTOR: Ms. Tania Irum

SECTION: D

TABLE OF CONTENT

[1- Introduction: 3](#_Toc512452323)

[2- POSIX Threads (PThreads): 3](#_Toc512452324)

[3- OpenMP 3](#_Toc512452325)

[4- Quick Sort 4](#_Toc512452326)

[5- Merge Sort 5](#_Toc512452327)

[6- Bubble Sort 5](#_Toc512452328)

# Introduction:

Ever since the 1970’s when the microprocessor that we know today was developed and launched increase in its performance been achieved by increasing the clock speed. All the way until early 2000 the increase in clock speed was statically, every two years the number of transistors that would fit on a chip would double.

Today it’s very common with a multi-core processor in most computers, laptops, work stations, servers, phones and tablets. Adding just the cores does not means high performance, in some cases it can actually mean worse performance because the individual cores may not have as high clock speed as a single core processor. It’s is absolutely vital that the software running on the hardware utilize the cores that the processor offer in order to get enhanced performance.

Developing software that can take advantage of multiple cores can be done in many different programming languages. If the languages do not have a standard library that supports threads there is often a third-party library that can be added to allow usage of threads. For all experiments in this paper C was used as main language, reasoning for using C was that C has been and is a very popular programming language and is used in many different areas of development, GCC compiler on Linux also has support for the two libraries that was tested. The threading techniques that was investigated in this paper was POSIX Threads and OpenMP.

# POSIX Threads (PThreads):

POSIX Threads is a common threading model when working in C or C++, it is a standardized model to work with threads created by IEEE. POSIX Threads can be used to parallelize task of a program in order to increase the execution speed.

POSIX Threads offers a header file and a library that must be included at compilation time or else the program wont compile because functions and data structures won’t be found by the compiler. POSIX Threads offers a lot of functions to create, manipulate, synchronize and kill threads in a program. Starting point for threads in a Pthreads program is a function called pthreads\_create, this function takes a pointer to a function that is the starting point for the new thread. Pthread\_exit then is used to kill the thread and in the main thread pthread\_join is used to wait for the executing threads.

# OpenMP

OpenMP is a model for parallel programming in shared-memory systems and was developed in the 1990’s by SGI. OpenMP is available for three different programming languages, Fortran, C and C++. It consists of a number of compiler directives that can be used in the source code, these directives need functions to run and those functions is specified in a library that must be included at compile time. It works by letting the compiler translate the directives in to functions that is found in the OpenMP library, it does this at compile time. That means that the compiler must have support for OpenMP, if it does not have support for OpenMP it won’t know what to do with the directives. In most cases this won’t crash the application, that is a sub goal of OpenMP. It should be portable between different systems, even systems that does not support it. Of course, these systems won’t have the increased performance from parallel computing but they should still be able to run the program. Note: Recursion in Open MP takes more time than Pthread because of task creation again and again.

# Quick Sort

Quick Sort is sort algorithm that was developed by Tony Hoare in 1959. It works by dividing the original array into sub-arrays. The first sub-array contains all the elements that is less than a chosen pivot element and the second sub-array contains all the elements that is larger than the pivot element. Now both of the arrays can be sorted separately, but before this is done they are partitioned again. To do this a new pivot is chosen in each of the sub arrays and then they are portioned into even smaller sub-arrays. This whole process continues until we have one element sub arrays that does not need to be sorted. In this partitioning process the array is sorted. In its worst-case scenario, it has a computational complexity of O(n2) and an average of O (n log n).

# Merge Sort

Merge sort is a stable divide and conquer algorithm and is considered to be one of the fastest. It recursively divides the data set into subarrays until each subarray consists of a single element. It then merges each subarray back, sorting each new subarray as it builds its way back to a single sorted array. Regardless of the shape of the data set to be sorted, merge sort performs the same number of steps, and will therefore have the same time complexity for all cases, O (n log n). Even though it is an efficient algorithm in terms of sorting, it has a drawback in that it uses O(n) extra memory when sorting. The implementation can spawn recursive tasks during both the sorting and merge phases to increase the amount of work that can be done in parallel. This ensures that even in the last steps of the algorithm, at the top of the merge sort hierarchy, where the longest, sorted subsets of the input data are merged, plenty of concurrent work is available to keep all worker threads busy.

# Bubble Sort

Bubble sort is the oldest, the simplest and the slowest sorting algorithm in use having a complexity level of O(n2). Bubble sort works by comparing each item in the list with the item next to it and swapping them if required. The algorithm repeats this process until to make passes all the way through the list without swapping any items. Such a situation means that all the items are in the correct order. By this way the larger values move to the end of the list while smaller values remain towards the beginning of the list. It is also used in order to sort the array such like the larger values comes before the smaller values. In other words, all items are in the correct order. It takes more time than other sorts like selection and insertion because it requires many more component exchanges and is just good for a pretty well-ordered array. More importantly bubble sort is usually the easiest one to write correctly.

# Conclusion

We have used several inputs in above parallel sorting program to show properties of the Open MP and POSIX Threads. Suggestions for enhancements to the specification were made whenever it seemed appropriate. It is also shows the problem of recursion in Open MP. Both Pthreads and Open MP has its own characteristics.