

**PROJECT REPORT**

**Parallel Programming Comparison of sorting**

**Algorithms using Pthreads vs.**

**OpenMP vs. serial**

**SUBJECT:OPERATING SYSTEM**

**GROUP MEMBERS:**

**AYAZ HASAN [20K-1044]**

**FERDEEN BAKHT[20K-0219]**

**ALI KHAFUL WARA[19K-1066]**

**INTRODUCTION**

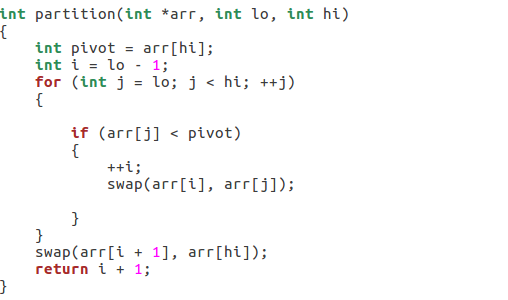
This project is dedicated to solving complex sorting algorithms(Quick sort,Merge Sort,Heap Sort) whom time complexity may take upto O(n2) or O(nlog(n)) with the help of parallel programming using p-thread and openMP.And The comparison between Parallel and serial Implementation .

**PROJECT DESCRIPTION**

**i) QUICK SORT**

Implementation of simple quicksort algorithm is as follows:

1. Choose a pivot element which is the last element of an array.
2. All elements which are smaller than the pivot element move to the left side and greater ones will move to the right side of the pivot.In this way array is divided into two subarray.

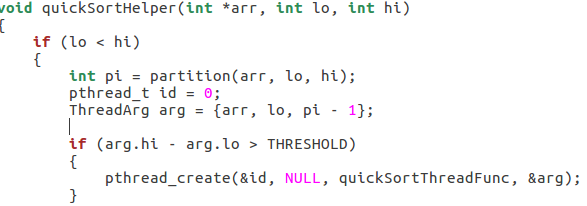


**Using Pthread**

In pthread we select a global pivot and compare each thread’s value with it and find all the elements which are smaller than this global pivot value.

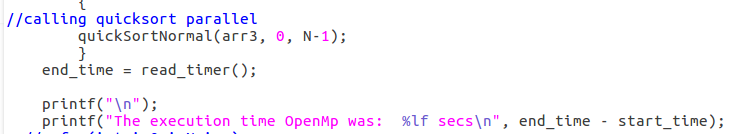


Now we will create threads in order to sort arrays in parallel.In serial we perform sorting in a single process while in thread we create a different process which is run in parallel .



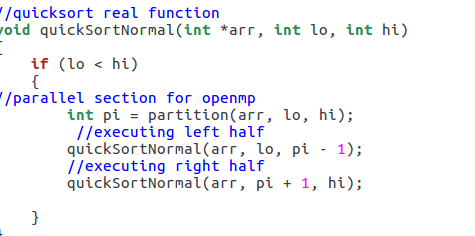
**Using OpenMP**

In OpenMp using shared memory we perform sorting in parallel



The above implementation calling the function of quick sort

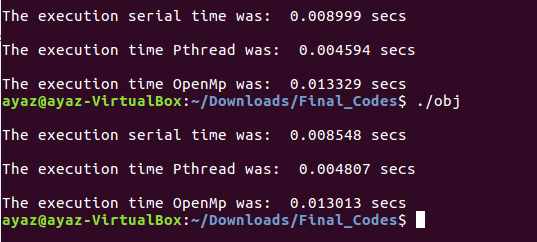
Next we get the index of pivot and store it in variable “pi” and on the basis of this variable we create a parallel section for openMP .1 section perform sorting on left half and other on right half.



**Result**

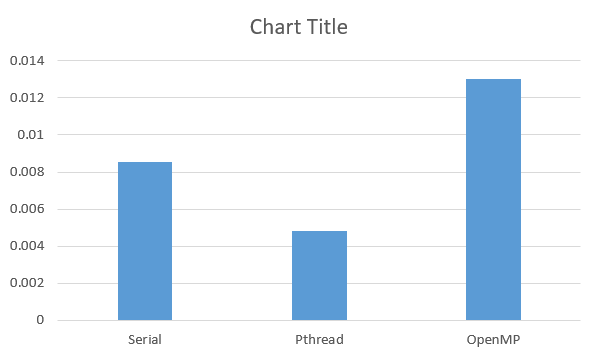
The output of the sorting algorithm quicksort by using serial ,pthread and openMP is as follows:

Note:we use random function to generate



**Comparison**

As We generate a numbers using random function so every time when execute the program we get different time but one thing is that we obtain from the result is that OpenMp is not suitable for this algorithm as compare to the Pthread and serial implementation.The problem with openMp is that it is difficult to express parallelism with recursion in sorting algorithm like QuickSort .And Quick sort is depend on recursion heavily .

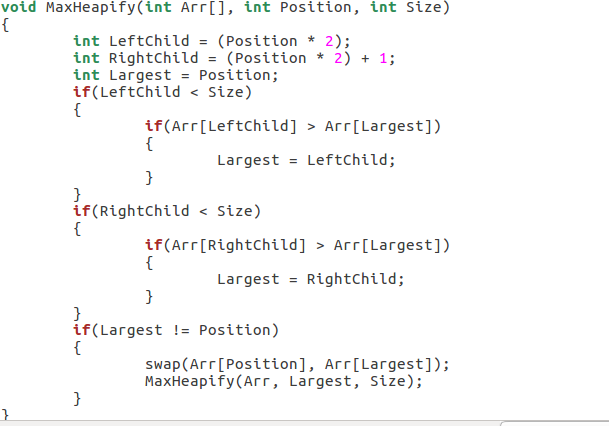


**ii) HEAP SORT**

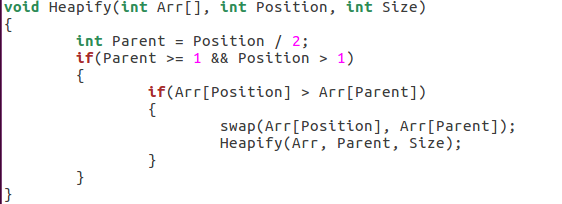
Implementation of the heap contain the following variables of algorithms:

* Build a max heap from the input data.
* At this point, the maximum element is stored at the root of the heap. Replace it with the last item of the heap followed by reducing the size of the heap by 1. Finally, heapify the root of the tree.
* The heapify procedure can only be applied to a node if its children nodes are heapified. So, the heapification must be performed in the bottom-up order.

Function used for Max/Min Heap output

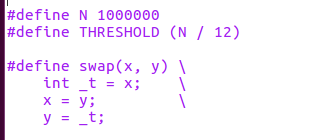


Function for Construction(nodes) Heap

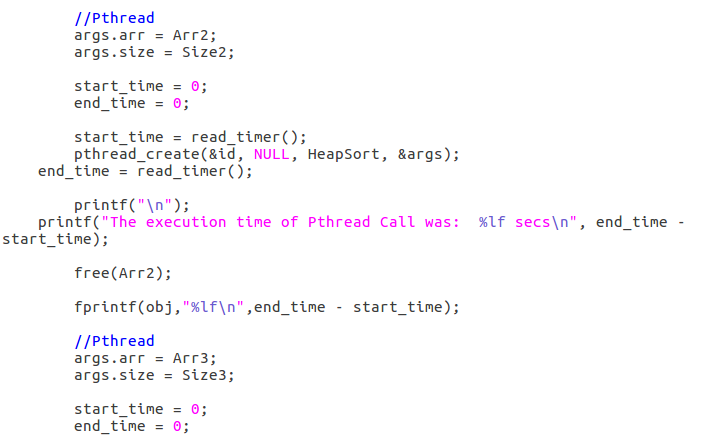


**Using Pthread**

In Pthread we select a global node to have easy access while threading and compare each thread’s value with it and discover all the nodes which are min/max than particular node value which is in condition while for that we also initialize the global function of **swap()** to have easy call while procedure threading.

****

Now creating the threads for sorting the nodes and call heapsort function while creating a thread for nodes.



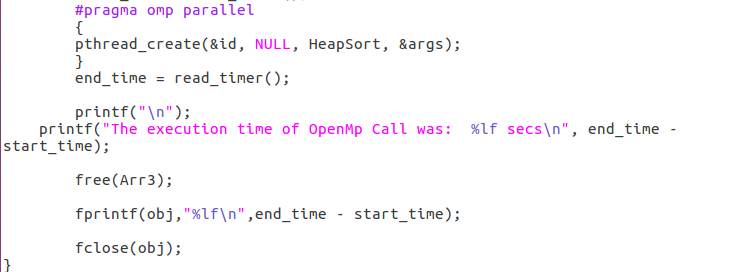
**args:** is the variable of structure which consist of array and its size

**readtimer ():** function for resulting the value time taken (start\_time & end\_time)

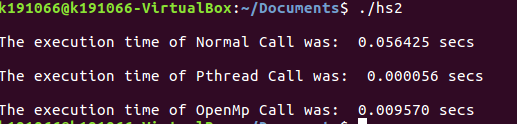
**Using OpenMP**

In openMP heapsort can be consider based on the hypercube model, as the heap data to be sorted are divided into n (n is the computer thread amount) blocks called sub-heap which will be sorted respectively. Thus, the scale of the problem is decomposed into 1/n, secondly, the sorted sub-heaps will be merged using merge sort algorithm, finally, the parallel optimized heap sort algorithm is compared to the traditional heap sort algorithm.

Implementation of Heapsort while using OpenMp



**Result** The Output is based on time execution of Heapsort using different procedure as Simple, Pthread, OpenMP. Readtimer function is moreover using in the backend of operating system to calculate the time during runtime.



**Comparison**

As We create a value utilizing arbitrary work so each time when execute the program, we get diverse time but one thing is that we get from the result is that openMP isn't appropriate for this calculation as compare to the Pthread and Normal implementation.

The issue with openMP is that it is troublesome to specific parallelism with recursion in sorting calculation like. And heapsort is depending on recursion intensely while inserting nodes.

**iii)Merge Sort**

Implementation of Merge sort is as follows

* We have a function called mergesort() which divides the function recursively into

different sections.

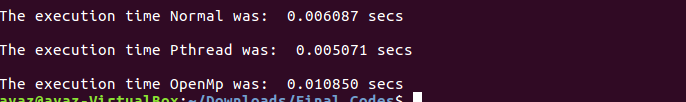
* Each section runs a function called merge() which creates a temporary array and sorts

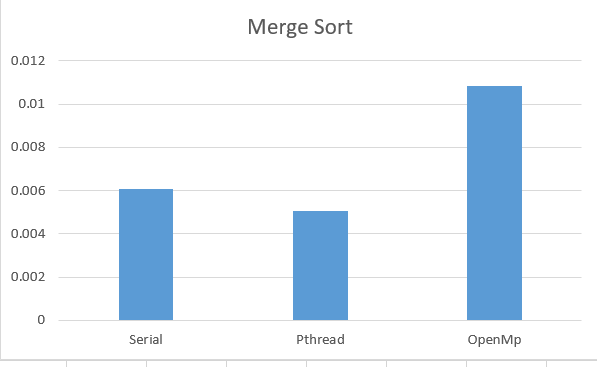
that particular section of recusion

* Both merge() and mergesort() functions run in such a way the small parts are sorted then the large parts of the array are being sorted

**Using OpenMp and Pthread**

Same Implementation as Quicksort





Tools and Technologies

* Programming Language : C & C++
* Platform:Linux

Tools and Technologies

* Programming Language : C & C++
* Platform:Linux