Assignment 4 Report

Shaunak Badani
20171004

1. Concurrent quick sort

Processes :

- A global array has been declared for all processes.
- The quicksort function is invoked.
- This function quickly performs an insertion sort if the number of elements is less than 5, and returns. Else :
- The array is partitioned around a particular pivot.
- The pivot in my code has been chosen as the first element of the array.
- The pivot is put in its right position, all the elements smaller than that are placed left of it, unordered, and all the elements bigger than that are placed right to it, unordered.
- Now, two child processes are created.
- One of the children takes care of the left half of the array, call it the left child.
- The right child sorts the right half of the array.
- This recursion goes on until the two ends of the function parameters do not converge.

Threads :

- A single thread is created, whose *start_routine* parameter, i.e. the function that the thread calls upon creation is *threaded_quicksort*.
- The partition function is called first. The pivot element is placed in its right position.
- After the partition function is over, two different threads are created. One to sort the left half, one to sort the right half.
- Those two threads are then created, which goes on recursively until l<r.

Performance analysis :

- For small N [N <= 5], normal and concurrent quicksort are almost the same speed, while threaded quicksort takes a little longer to complete.
- For N >= 7, normal quicksort is the fastest, followed by threaded quicksort, and last but not the least, concurrent quicksort.