Computer Systems Engineering-1

Assignment 4

Tanish Lad - 2018114005

Concurrent Quicksort Question Report

OVERVIEW

The code is an attempt to sort the given input array using the Quicksort algorithm, and find the performance differences between the normal version, the concurrent version and the threaded version and analyse them.

Solution Approach

1. Normal Quicksort

   Take any random element as pivot and partition the array into two halves such that the elements smaller than the pivot go into the left half, and the elements bigger than the pivot go into the right half, and sort both the parts recursively by again calling the Quicksort function.

   If the number of elements in the range are less than 5, then that range is directly sorted using Insertion sort.

2. Concurrent Quicksort

   Same Approach, but here we fork and in the child process, we sort the left part by calling the Quicksort function again,

and in the parent process, we again fork and in it's child process, we sort the right half.

If the number of elements in the range are less than 5, then that range is directly sorted using Insertion sort.

3. Threaded Quicksort

   Same Approach as that of Concurrent Quicksort, but instead of creating new processes, we create new threads and sort left and right halves in those threads and then join the threads.

   If the number of elements in the range are less than 5, then that range is directly sorted using Insertion sort.

Performance Analysis

Time is in seconds

N	Normal Quicksort	Threaded Quicksort	Concurrent Quicksort
100	0.000025	0.004760	0.007326
1000	0.000082	0.010578	0.043635
10000	0.000756	0.087288	0.431645
50000	0.004683	0.514099	2.975814

The input array is the decreasing order of elements from 1 to N.

As can be seen, the fastest of these is the Normal Quicksort, followed by Threaded Quicksort and Concurrent Quicksort is the slowest.

I did not test the codes for small inputs because the insertion sort effect and be large in those cases.

If input went as large as 10^5, the threaded version resulted into segmentation fault.

The threaded version is slower due to creation of a new thread each time, and the concurrent version is even slower because it creates a new process each time and access of shared memory takes considerable time, but inter-thread data sharing is much faster than inter-process data sharing, where we need to use Inter Process Communication (IPC) to get data in and out of the process.