Computer Systems Engineering - I

Course Assignment 04

Naren Akash, R J
2018111020

An Introduction

Quicksort is generally recognized as the fastest sorting algorithm based on the comparison of keys, in the average case. Quicksort has some natural concurrency. The low list and high list can sort themselves concurrently.

Concurrent Quicksort

We randomly choose a pivot from one of the processes and broadcast it to every process. Each process divides its unsorted list into two lists: those smaller than (or equal) the pivot, those greater than the pivot. Each process in the upper half of the process list sends its "low list" to a partner process in the lower half of the process list and receives a "high list" in return.

Now, the upper-half processes have only values greater than the pivot, and the lower-half processes have only values smaller than the pivot. Thereafter, the processes divide themselves into two groups and the algorithm recurses. After $log(P)$ recursions, every process has an unsorted list of values completely disjoint from the values held by the other processes. The largest value on the process i will be smaller than the smallest value held by process $i+1$. Each process can sort its list using sequential quicksort.

Observations

	Sequential Quicksort	Concurrent QuickSort
$\lvert Input \rvert = 2$	$0:02.91\ seconds$	$0:04.28\ seconds$ *(Insertion Sort)*
$\lvert Input \rvert = 15$	$0:07.80\ seconds$	$0:08.08\ seconds$

Performance Analysis

It is surprising to note that the sequential quicksort is better than the concurrent quicksort.

When, say, left child, access the left array, the array is loaded into the cache of a processor. Now when the right array is accessed (because of concurrent accesses), there is a cache miss since the cache is filled with left segment and then the right segment is copied to the cache memory. This to-and-fro process continues and it degrades the performance to such a level that it performs poorer than the sequential code. There are ways to reduce the cache misses by controlling the workflow of the code. But they cannot be avoided completely.