CS69001: Computing Lab – I
Autumn 2009

Assignment 5

Sorting a shared array in parallel

Due: October 16, 2009 (Friday)

In this assignment, parallel versions of the merge sort algorithm are to be implemented. A process runs throughout the lifetime of the program. Let us call this process the *parent process*. This parent process creates child processes as and when needed in order to realize parallel versions of the merge sort algorithm.

The parent process first creates two shared memory segments each capable of holding an integer array of size 10^6–10^7. One of these arrays is to be used as the input array—call it A. The other array B is used to store the sorted result. Before calling each of the following parts, the parent (re)populates A with a randomly generated sequence of integers.

Part 1 (25)

The parent first creates a child process. The child process sorts the right half of A, whereas the parent process sorts the left half of A. The two sorted lists are stored in A itself. Any sorting algorithm may be used for sorting each half of the array. After the child process sorts its part, it terminates. The parent (after doing its own sorting work) waits until the child terminates. After that, the parent merges the two sorted sublists into a single sorted list B.

Part 2 (25)

In this part, the merging step is also done in parallel. After the sorting child terminates, the parent creates another child process. The parent starts merging the two sorted lists in the usual way, but stops as soon as half of the elements are generated. The new child process, on the other hand, starts merging from the end of B (from the largest element to smaller elements). The child terminates after it writes the larger half in B.

For solving the next two parts, the user supplies a parameter d, a small positive integer (like $1 \leq d \leq 4$), indicating the maximum allowed depth of recursion.

Part 3 (25)

Run the merge sort recursion up to depth d. During each invocation of the recursive function, do not make a second recursive call, but create a child process to handle the right half in parallel. When the depth of recursion exceeds d, the corresponding part of the array is sorted by a single process using any sorting algorithm. That is, no further child processes are created.

In an invocation where a child is created, the creator process (when done with its left half) waits until the child finishes. The two sorted sublists are then merged, and the process (unless it is the parent process) terminates.

Part 4 (25)

Repeat Part 3 with the only exception that each merging task is shared with a (new) child process. Follow a strategy similar to Part 2.

Submit a single C/C++ file solving all the above parts. The file must contain your name and roll number.