Spring 2024

CS 412 (Algorithms: Design and Analysis)

Weekly Challenge 03: Parallel Computing

Announced: Friday, January 26, 2024.
Deadline: Friday, February 9 , 2024 (11:59 pm PKT).
Total marks: 2.

Instructions: Submit **individually** your solution as a PDF with the file name as your *studentID.pdf*; typeset in LaTeX. You must submit your solution on Canvas.

---

1. (1 point) The worst-case complexity for any sequential comparison-based sorting algorithm is $\Omega(n\lg n)$. However, parallel computing can further reduce the running time of the algorithms. In its most basic sense, parallel computing refers to the simultaneous execution of tasks using multiple processors and cores. The algorithms can be parallelized when some of their steps can be computed in parallel. Lack of dependencies implies potential for parallel execution. For example, divide-and-conquer approach-based algorithms are often well-suited for parallelization. The division of the problem into sub-problems that can be solved independently facilitates parallel execution.

   Selection Sort is a well-known sorting technique that scans an array to find the maximum item, puts it at the last location in the array, and then scans the array $(1 \ldots n-1)$ for the second maximum item, places it before the last location, then third maximum $(1 \ldots n-2)$ and so forth, until reaches the smallest item to be put at the first location of the array. It has $O(n^2)$ complexity.

   Is it possible to parallelize the classic selection sort without making any changes and gain performance improvement? If yes, explain the aspects of the algorithm that can be parallelized. Identify an approach with the help of which parallelization capabilities of selection sort can be improved. Explain your proposed approach in 4 to 5 lines.

   Note: Performance improvement in parallel computing comes from many different factors like the nature of the algorithm, available computing architecture, number of processors or cores, size of data, etc. Note that for this particular assignment, we are only interested in exploring the nature of the algorithm. Consider that the number of available processors or cores is n, where $n > 1$.

   Useful resources

   - Example (Bubble Sort): 220-L20 (usfca.edu) (Reference: CS 220: Introduction to Parallel Computing, University of San Francisco)
   - Chapter 26 of Introduction to Algorithms, fourth edition. Authors: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein

   Solution: While the conventional selection sort algorithm cannot be parallelized without making any changes, parallelization can be done to some xtent with making some changes. The Min-Max Bi-directional parallel selection sort [1] will be the proposed approach:

   1 Dividing the array into subarrays and assigning each subarray to a core/thread.

   2 Finding the local minimum and maximum values from each subarray.

   3 Taking the minimum from first subarray and comparing it with the minimum of the second subarray and so on.

   4 By swapping, put them at their exact locations.

5 Taking the maximum of the first subarray and comparing it with the maximum of second subarray and so on.

6 By swapping, put them at their exact locations.

7 Repeat these steps for the whole array

Although this approach might improve actual runtime performance, the time complexity remains $O(n^2)$ since the fundamental operations of comparing elements and swapping them still require $O(n^2)$ in the worst case.

References

[1] Khaled Thabit and Afnan Bawazir. Novel approach of selection sort algorithm with parallel computing and dynamic programing concepts. *Journal of King Abdulaziz University Computing and Information Technology Sciences*, 2:27–44, 2013.