Assignment No: - HPC-Group1-2

TITLE	Parallel Sorting Algorithms
PROBLEM STATEMENT /DEFINITION	Write a program to implement Parallel Bubble Sort and Merge sort using OpenMP. Use existing algorithms and measure the performance of sequential and parallel algorithms.
OBJECTIVE	• To understand concept of Bubble Sort and Merge Sort based on sequential algorithm. • To understand concept of parallel algorithm. • To compare performance by varying number of processors used and also with sequential algorithm.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems 1. Open source Linux or its derivative 2. Master slave parallel computation model
REFERENCES	• Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar, "Introduction to Parallel Computing", 2nd edition, Addison-Wesley, 2003, ISBN: 0-201-64865-2.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Related Mathematics 8. Algorithm. 9. Test Cases 10. Conclusion and applications

Assignment No: - HPC-Group1-2

Problem statement: Write a program to implement Parallel Bubble Sort and Merge sort using OpenMP. Use existing algorithms and measure the performance of sequential and parallel algorithms.

- **Aim**
  Write a program to design and implement parallel Bubble Sort and Merge Sort algorithm.

- **Prerequisites**
  - Concept of existing sequential algorithms.
  - Concept of High Performance Computing.

- **Learning Objectives**
  - To understand concept of Bubble Sort and Merge Sort based on sequential algorithm.
  - To understand concept of parallel algorithm.
  - To compare performance by varying number of processors used and also with sequential algorithm.

- **Learning Outcomes**
  After successfully completing this assignment, you should be able to
  - Display result for parallel Bubble Sort and Merge Sort.
  - Analyze performance by varying number of processors used and also with sequential algorithm.
  - Calculate speedup, efficiency, throughput

- **Concepts related Theory :-**

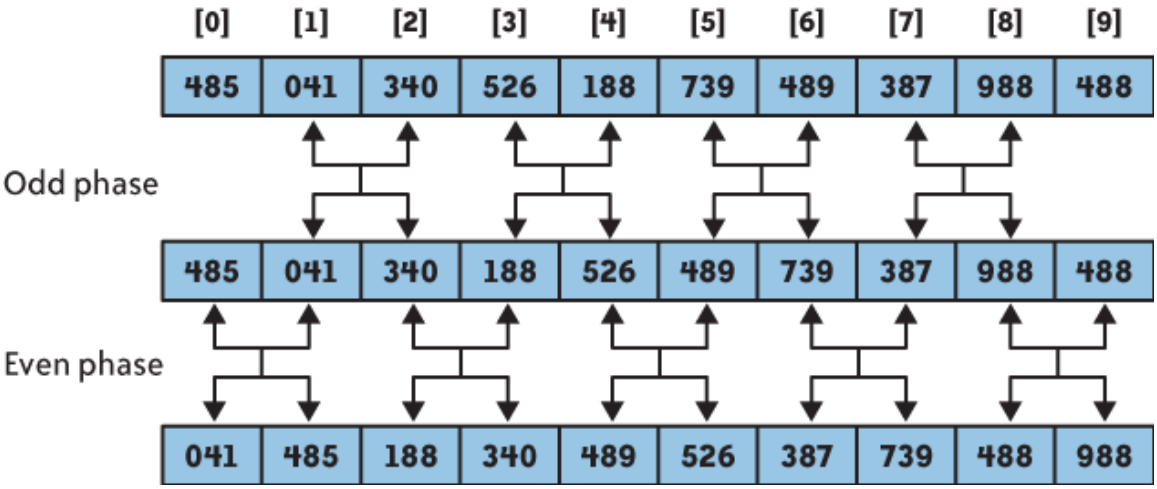
- **Bubble Sort Algorithm:-**

  Sequential Bubble Sort Algorithm:

  One of the straight-forward sorting methods
  –Cycles through the list
  –Compares consecutive elements and swaps them if necessary
  –Stops when no more out of order pair.
  _Slow & inefficient
  _Average performance is $O(n^2)$.

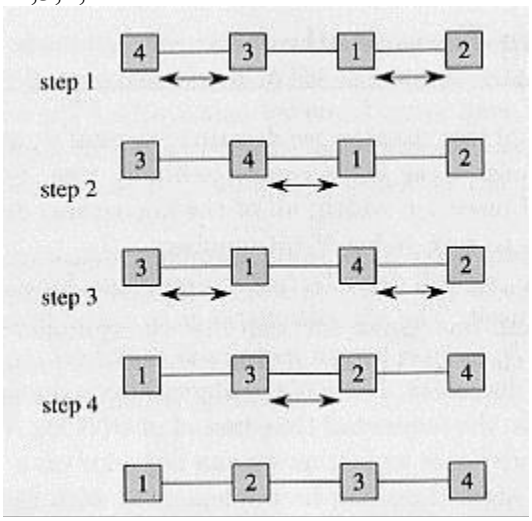
  Parallel Bubble Sort

  Compare all pairs in the list in parallel.
  When to stop?
  Shared flag, sorted, initialized to true at beginning of each iteration (2 phases), if any processor perform swap, sorted = false



Parallel Bubble Sort Complexity
Sequential bubble sort, $O(n^2)$.
Parallel bubble sort? (if we have unlimited # of processors)
Do n-1 comparisons for each iteration => O(n)

Parallel Bubble Sort Example:
How many steps does it take to sort the following sequence from least to greatest using the Parallel Bubble Sort? How does the sequence look like after 2 cycles?
•4,3,1,2



- **Merge Sort Algorithm:-**

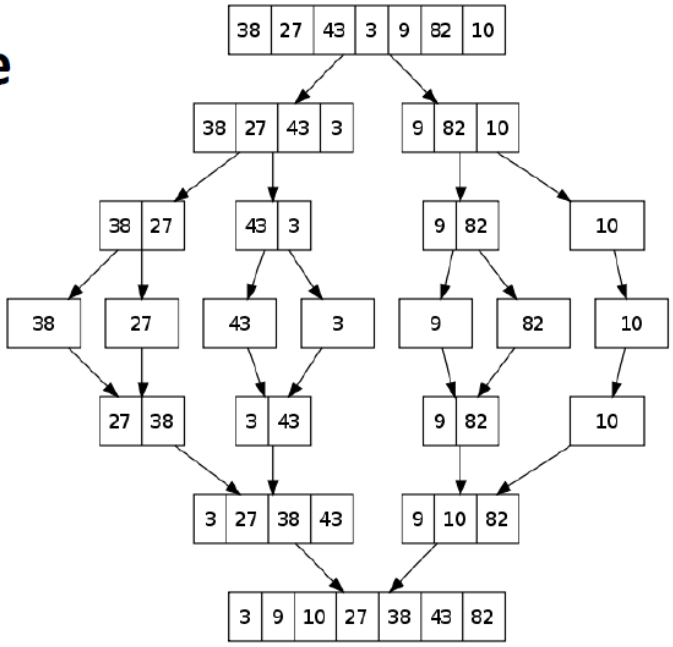
Sequential Merge Sort:

Divide and Conquer:
•Dividing problem into sub-problems
•Division usually done through recursion
•Solutions from sub-problems then combined to give solution to the original problem.

Collects sorted list onto one processor

•Merges elements as they come together
•Simple tree structure
•Parallelism is limited when near the root

Example



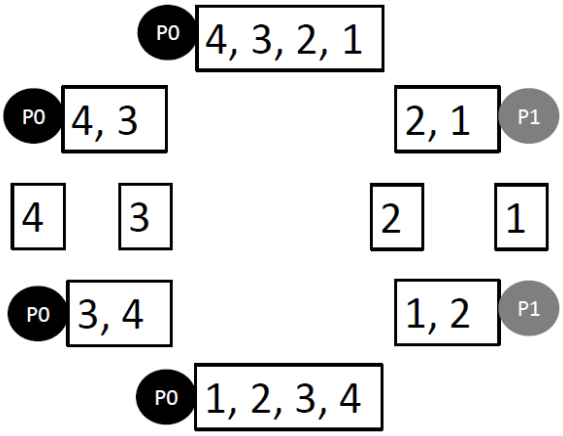
Merge Sort Complexity:

$$T(n) = \begin{cases} b & n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & n > 1 \end{cases}$$

Solve the recurrence relation
T(n) = O(nlogn)

Parallel Merge Sort:
•Parallelize processing of sub-problems
•Max parallelization achieved with one processor per node (at each layer/height).

Parallel Merge Sort Example:
Perform Merge Sort on the following list of elements. Given 2 processors, P0 & P1, which processor is responsible for which comparison?
•4,3,2,1



Parallel Merge Sort Complexity:
•Merge sort, O(nlogn)
•Easy way to remember complexity, n (elements) x logn (tree depth)
•If we have n processors, O(logn)

- **Test Cases:**
  Students should write test cases depending on their input and test the program on large input data

- **YouTube video links:**
  https://youtu.be/F8Cluc31bJc
  https://youtu.be/SpBPp3JjFb4
  https://youtu.be/QaiEB4BjjNg

- **Conclusion :**
  After successfully completing this assignment, student should be able to understand and implement parallel bubble sort and merge sort in OpenMP.