S.P.P.U. External Practical Viva Questions and Answers

Name: HK

Viva Questions and Answers (Parallel Merge Sort using OpenMP)

1. What is the objective of your project?

Answer:

The objective is to implement and parallelize the Merge Sort algorithm using OpenMP tasks to sort an array efficiently by utilizing multiple CPU cores.

2. What is Merge Sort?

Answer:

Merge Sort is a divide-and-conquer algorithm that recursively divides the array into two halves, sorts them, and then merges the sorted halves.

3. How have you parallelized the Merge Sort?

Answer:

Using #pragma omp parallel with #pragma omp single and #pragma omp task directives to allow the two recursive calls to mergeSort() to be executed in parallel.

4. What is the role of '#pragma omp task'?

Answer:

It creates a new task that can be executed in parallel with other tasks, helping in parallel execution of the left and right subarray sorting.

5. Which libraries have you used and why?

Answer:

- iostream - for input and output operations.
- omp.h - for OpenMP functions like task creation and measuring parallel execution time.

6. How do you measure the execution time?

Answer:

Using omp_get_wtime() function to record start and stop time and calculating the difference.

7. Why is dynamic memory used for arrays?

Answer:

Dynamic memory (new int[n]) allows flexibility for the user to define array size at runtime.

8. What are the advantages of parallel merge sort?

Answer:

- Reduces sorting time significantly on multicore systems.
- Efficient for large datasets due to recursive division and merging in parallel.

9. How does the 'merge' function work?

Answer:

It combines two sorted subarrays into one sorted array by comparing elements and copying them into the main array sequentially.

10. What are potential challenges in parallelizing recursive algorithms?

Answer:

- Overhead of task creation if tasks are too small.

- Managing memory and thread synchronization to avoid conflicts.

Best of Luck, HK!