**Class:** Final Year (Computer Science and Engineering)

**Year:** 2024-25 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 3**

**Exam Seat No:21510043**

**Name:shweta kakade**

**Title of practical:**

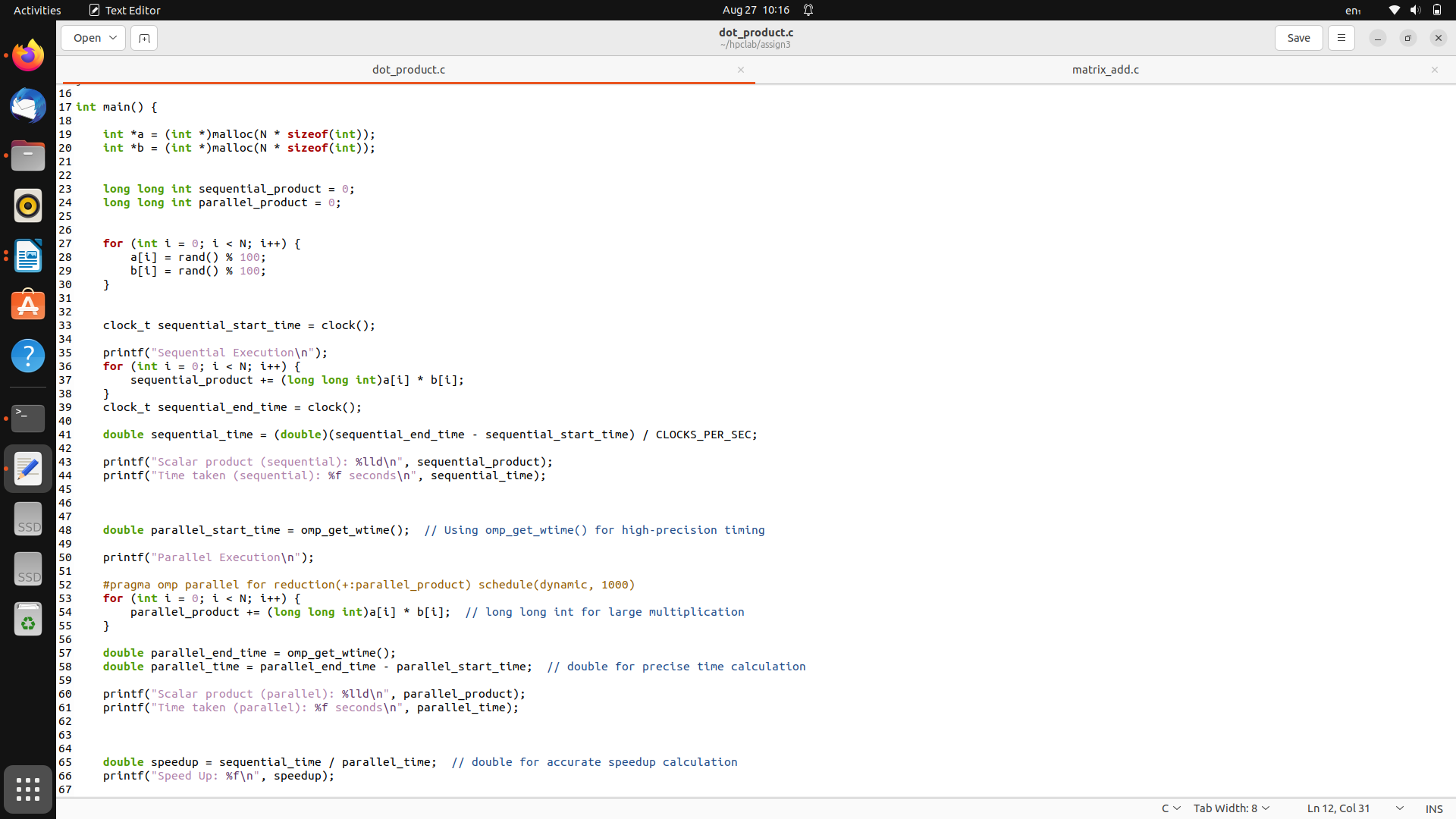
Study and Implementation of schedule, nowait, reduction, ordered and collapse clauses

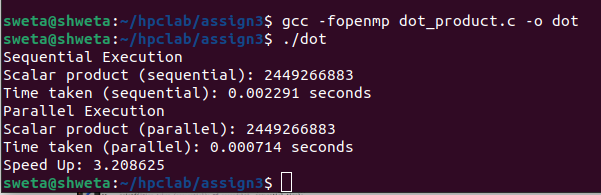
**Problem Statement 1:**

Analyse and implement a Parallel code for below program using OpenMP.

// C Program to find the minimum scalar product of two vectors (dot product)

**Screenshots:**

****

****

**Information and analysis: it gives result that paralle execution take less than sequential execution.**

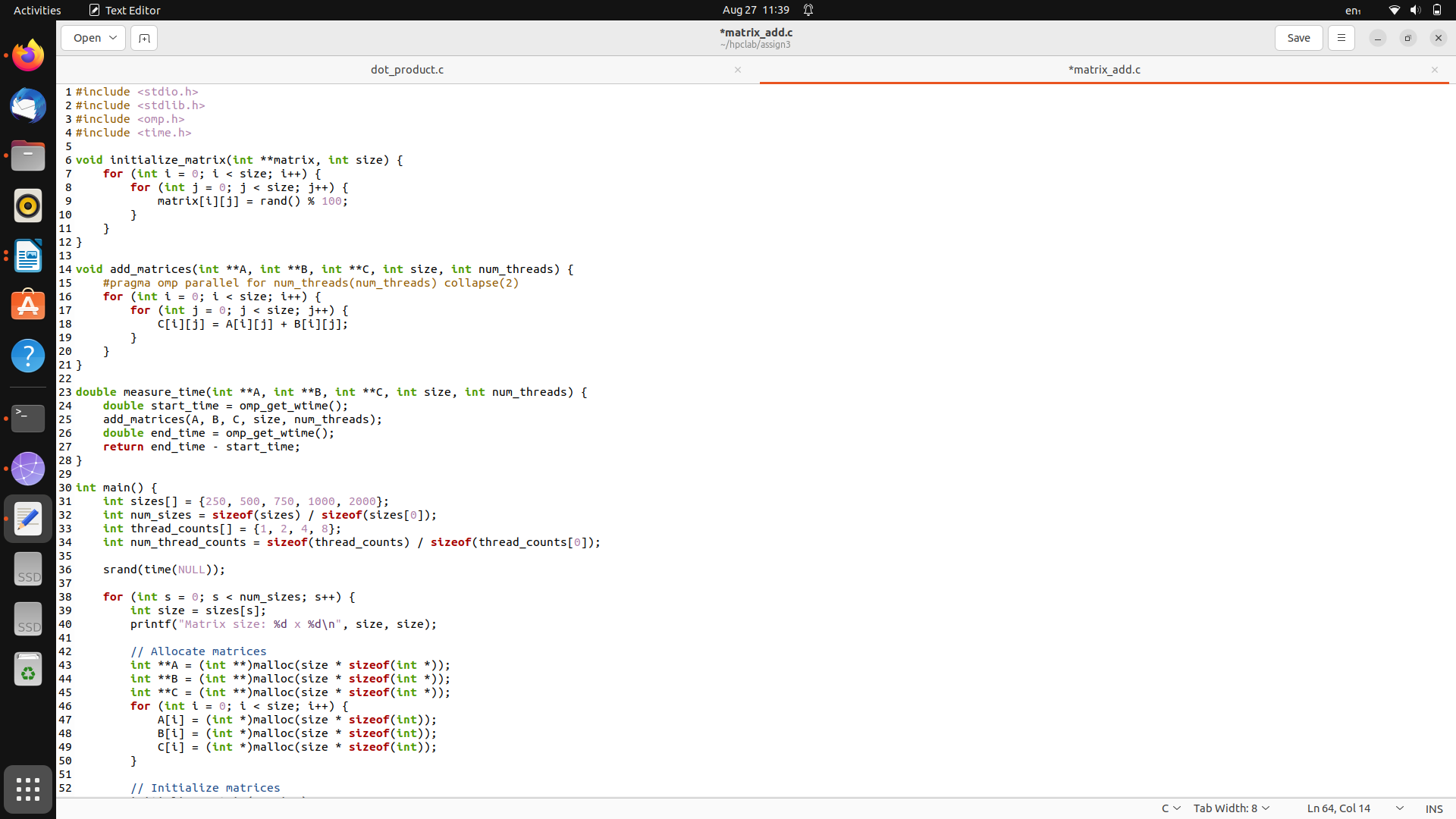
**Problem Statement 2:**

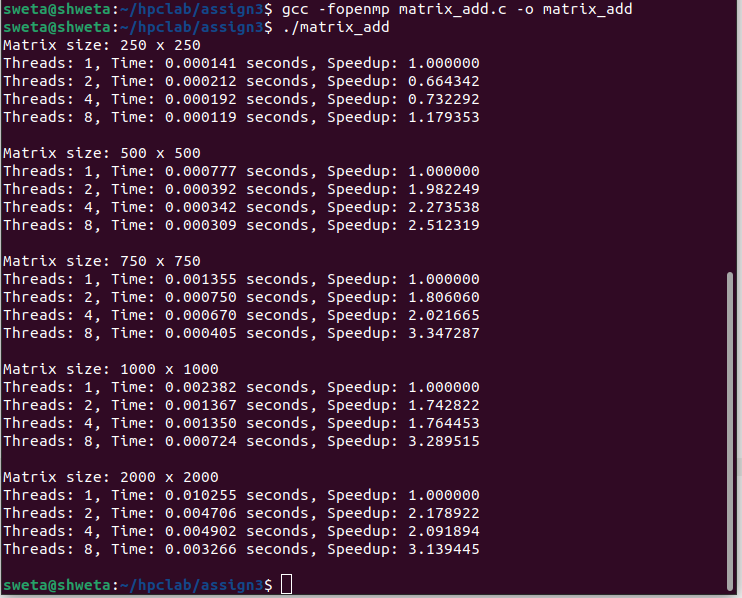
Write OpenMP code for two 2D Matrix addition, vary the size of your matrices from 250, 500, 750, 1000, and 2000 and measure the runtime with one thread (Use functions in C in calculate the execution time or use GPROF)

i. For each matrix size, change the number of threads from 2,4,8., and plot the speedup versus the number of threads.

ii. Explain whether or not the scaling behaviour is as expected.

**Screenshots:**

****

****

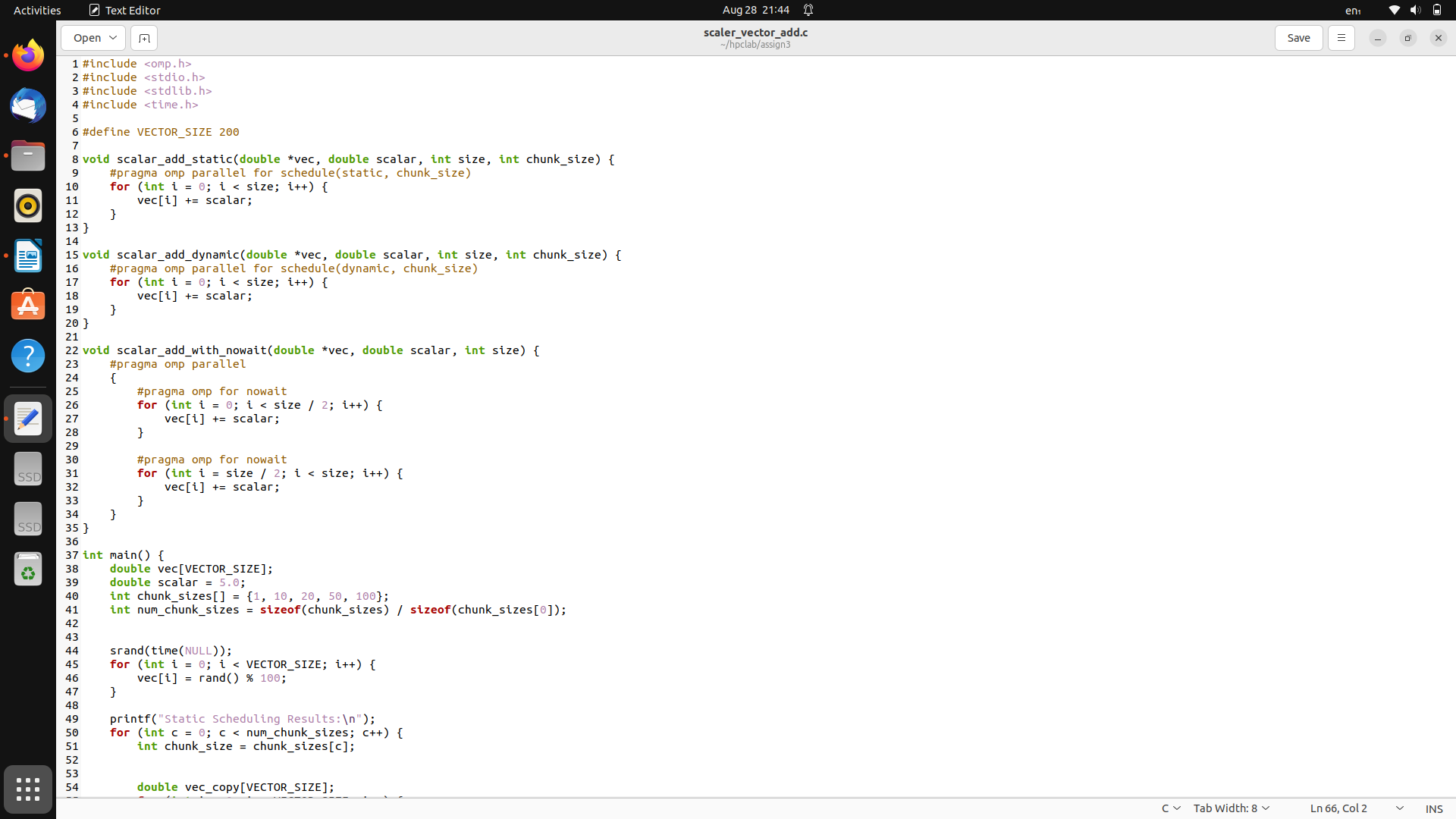
**Information and analysis:**

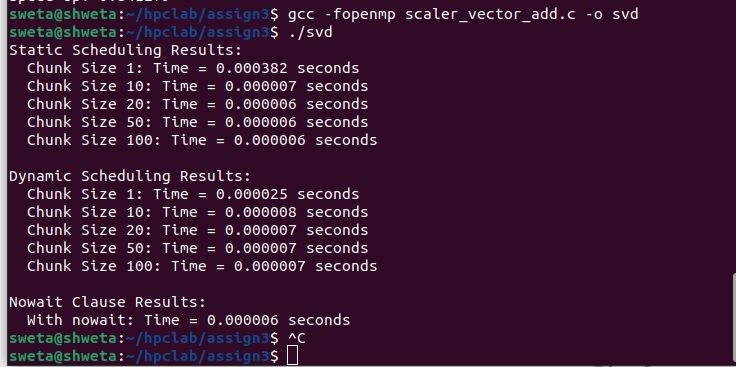
Speedup: Speedup is the ratio of single-threaded execution time to multi-threaded execution time. Ideally, it should increase with the number of threads, but overhead and inefficiencies can sometimes result in less-than-ideal scaling.Parallel Efficiency: For small matrix sizes, the overhead of managing multiple threads can negate the benefits of parallelism. As the problem size increases, the benefits of parallelism become more apparent, leading to better speedup and efficiency.

**Problem Statement 3:**

For 1D Vector (size=200) and scalar addition, Write a OpenMP code with the following: i. Use STATIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup. ii. Use DYNAMIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup. iii. Demonstrate the use of nowait clause.

**Screenshots:**



****

**Information and analysis:**

* Static Scheduling Results**:**

The output shows the time taken for different chunk sizes (1, 10, 20, 50, and 100) in static scheduling. Static scheduling means the workload is divided into chunks that are evenly distributed across threads.

The time taken decreases significantly as the chunk size increases, demonstrating how larger chunks can reduce overhead and improve performance.

* Dynamic Scheduling Results:

Similar to static scheduling, but with dynamic scheduling, the workload is distributed to threads as they become available, rather than in a predetermined manner.

The times are even lower than static scheduling, suggesting that dynamic scheduling might be more efficient for this specific task.

* Nowait Clause Results:

The time recorded when using the nowait clause is extremely low, indicating that avoiding the implicit barrier at the end of parallel loops can lead to significant performance gains.

**Github Link: https://github.com/shweta29k/HPC\_LAB**