Walchand College of Engineering, Sangli
Department of Computer Science and Engineering
Name: Meet Vipul Gandhi

PRN: 2020BTECS00112

Class: Final Year (Computer Science and Engineering)

Year: 2023-24   **Semester:** 1

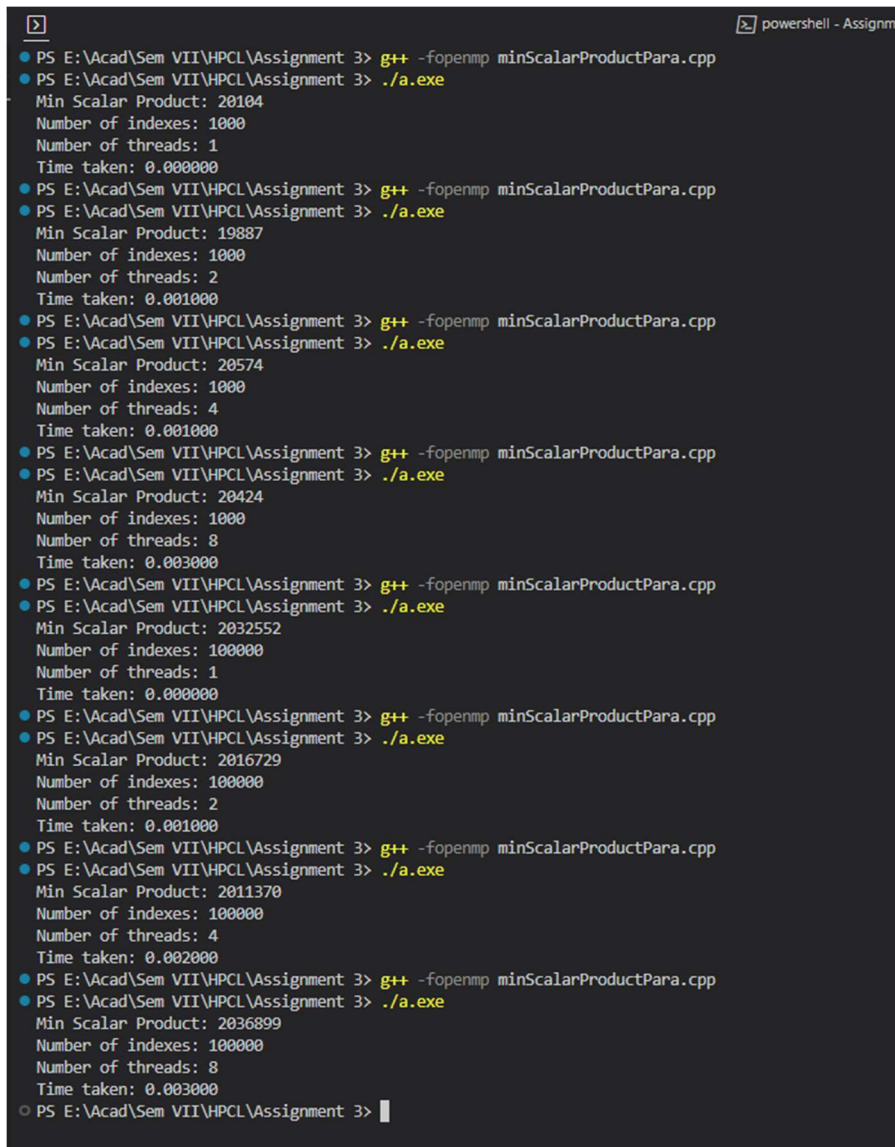**Course:** High Performance Computing Lab

<h2 style="text-align:center">Practical No. 3</h2>

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:

Final Year: High Performance Computing Lab 2023-24 Sem I

Information and analysis:

The outer loop iterates through the array, similar to the sequential bubble sort. Inside the outer loop, start is computed based on whether it's an even or odd pass. We then use *#pragma omp parallel for* to parallelize the inner loop. Each thread gets its own j, but they share the nums array. The inner loop logic remains the same as in the sequential bubble sort. Then we use *#pragma omp parallel reduction(+:sum)* to parallelize the for loop to calculate scalar product.
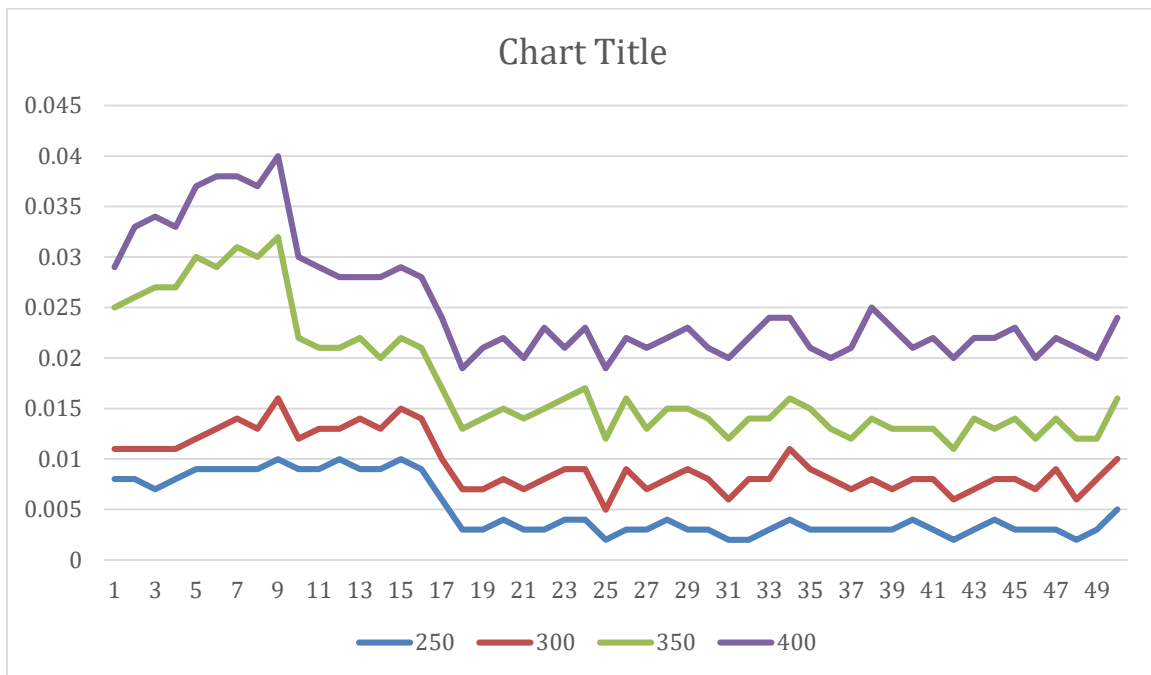
Final Year: High Performance Computing Lab 2023-24 Sem I
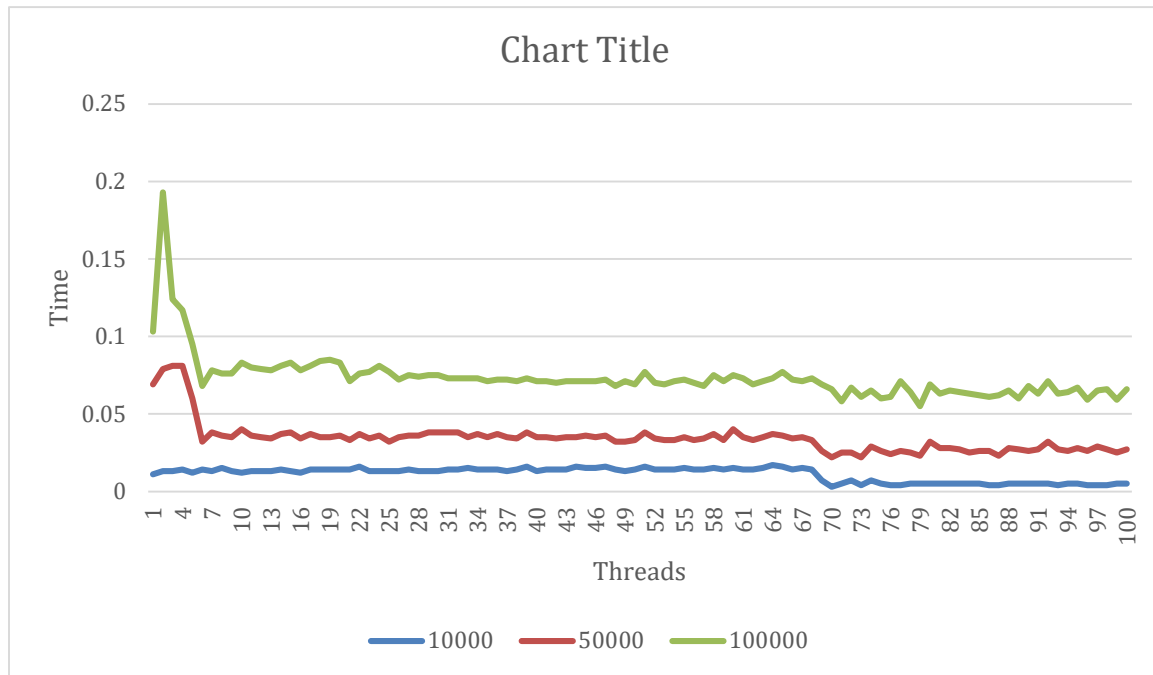
Threads \ Size	10000	50000	100000
1	0.011	0.058	0.034
2	0.013	0.066	0.114
3	0.013	0.068	0.043
4	0.014	0.067	0.036
5	0.012	0.048	0.035
6	0.014	0.018	0.036
7	0.013	0.025	0.04
8	0.015	0.021	0.04
9	0.013	0.022	0.041
10	0.012	0.028	0.043
11	0.013	0.023	0.044
12	0.013	0.022	0.044
13	0.013	0.021	0.044
14	0.014	0.023	0.044
15	0.013	0.025	0.045
16	0.012	0.022	0.044
17	0.014	0.023	0.044
18	0.014	0.021	0.049
19	0.014	0.021	0.05
20	0.014	0.022	0.047
21	0.014	0.019	0.038
22	0.016	0.021	0.039
23	0.013	0.021	0.043
24	0.013	0.023	0.045
25	0.013	0.019	0.045

Final Year: High Performance Computing Lab 2023-24 Sem I

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:

Final Year: High Performance Computing Lab 2023-24 Sem I

Threads \ Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
250	0.008	0.008	0.007	0.008	0.009	0.009	0.009	0.009	0.01	0.009	0.009	0.01	0.009	0.009	0.01
300	0.003	0.003	0.004	0.003	0.003	0.004	0.005	0.004	0.006	0.003	0.004	0.003	0.005	0.004	0.005
350	0.014	0.015	0.016	0.016	0.018	0.016	0.017	0.017	0.016	0.01	0.008	0.008	0.008	0.007	0.007
400	0.004	0.007	0.007	0.006	0.007	0.009	0.007	0.007	0.008	0.008	0.008	0.007	0.006	0.008	0.007

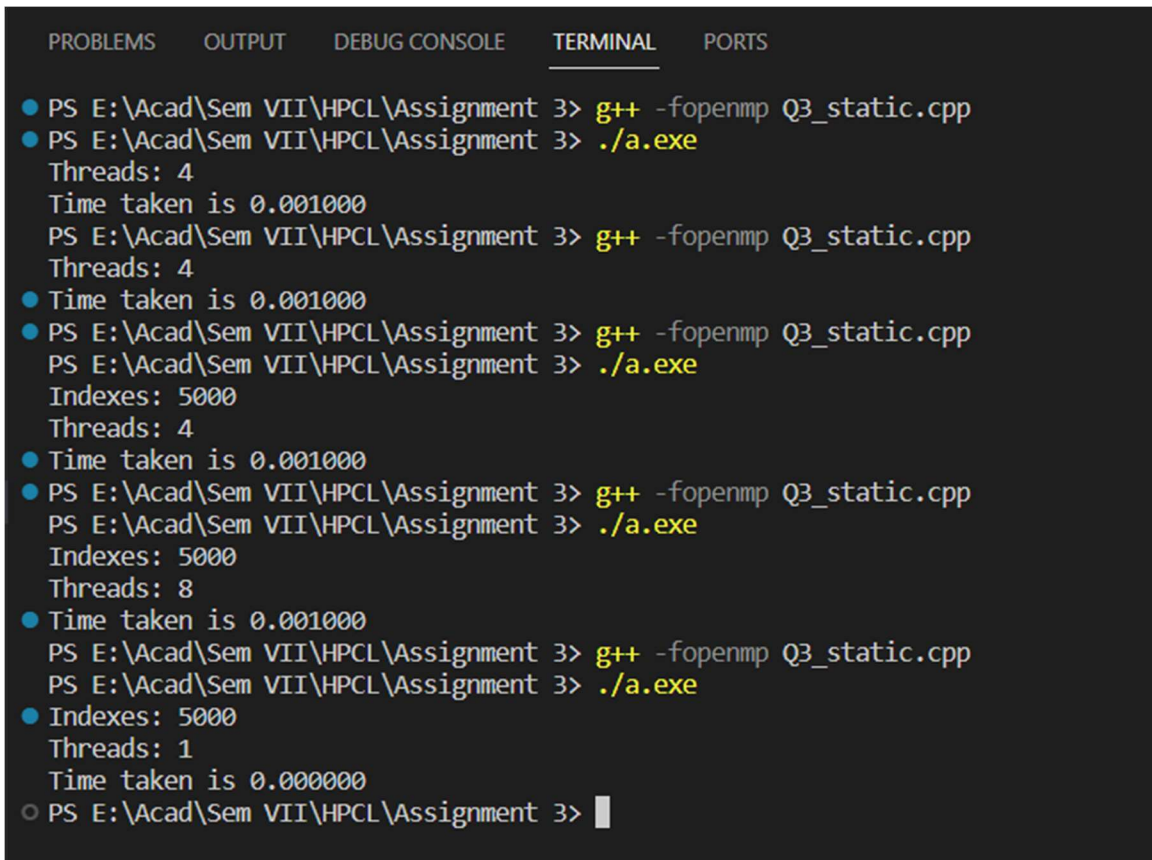| Threads \ Size	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
250	0.009	0.006	0.003	0.003	0.004	0.003	0.003	0.004	0.004	0.002	0.003	0.003	0.004	0.003	0.003
300	0.005	0.004	0.004	0.004	0.004	0.004	0.005	0.005	0.005	0.003	0.006	0.004	0.004	0.006	0.005
350	0.007	0.007	0.006	0.007	0.007	0.007	0.007	0.007	0.008	0.007	0.007	0.006	0.007	0.006	0.006
400	0.007	0.007	0.006	0.007	0.007	0.006	0.008	0.005	0.006	0.007	0.006	0.008	0.007	0.008	0.007

Information and analysis:

In this code, we have a function matrix_addition that performs the addition of two matrices. The main function iterates over different matrix sizes and different numbers of threads. It allocates memory for the matrices, initializes them with some values, and then measures the execution time of the matrix addition operation.

Collapse clause: The collapse clause is used to combine multiple loops into a single loop for parallelization. It can improve parallelization efficiency when dealing with nested loops. In this example, the two loops are combined into a single loop for parallelization.

Final Year: High Performance Computing Lab 2023-24 Sem I

Problem Statement 3: For 1D Vector (size=200) and scalar addition, Write an 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:

a. Schedule clause with static chunk size

Final Year: High Performance Computing Lab 2023-24 Sem I

b. Schedule clause with dynamic chunk size

Size\Threads	200	1000	10000	10000000
1	0	0	0.08	0.168
4	0.01	0.01	0.06	0.264
6	0.01	0	0.065	0.25

Final Year: High Performance Computing Lab 2023-24 Sem I

c. Nowait

	Time
With NoWait Clause	0.003
Without NoWait Clause	0.002

Final Year: High Performance Computing Lab 2023-24 Sem I

Information and analysis:

a. **Schedule Clause**: The schedule clause in OpenMP controls how loop iterations are divided among threads in a parallel loop. It has different scheduling options like *static, dynamic, guided, and auto*.

    i. **Static scheduling:** The loop iterations are divided into chunks of size chunk_size and assigned to threads before execution. Each thread gets a fixed set of iterations at compile time.

    ii. **Dynamic Scheduling:** In dynamic scheduling, the iterations are assigned to threads dynamically at runtime. Each thread receives a chunk of work, executes it, and then requests another chunk until all iterations are processed.

    iii. **Guided Scheduling:** Guided scheduling is a hybrid approach that starts with large chunks and progressively reduces the chunk size as the loop progresses. It's particularly useful when you don't know the workload distribution in advance.

    iv. **Auto Scheduling:** With auto scheduling, the compiler decides whether to use static, dynamic, or guided scheduling based on heuristics. This allows the compiler to choose the most suitable scheduling strategy for the specific loop and system.

b. **Nowait Clause:** The nowait clause in OpenMP is used to indicate that a thread can proceed with its tasks without waiting for other threads. It's often used to overlap computations with other parallel regions. In above example we executed the program without using the nowait clause and we can see that helloworld function gets called after all the threads complete executing the for loop. In second run we add the nowait clause and we can see that helloworld function gets called as soon as the for-loop execution is completed.

GitHub Link: https://github.com/meetgandhi692/HPC-Lab/tree/79b911b51b14aed89f30d1a35fe33b2761b4459f/Assignment%203