**Class:** Final Year B.Tech(Computer Science and Engineering)

**Year:** 2025-26 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 3**

**Exam Seat No:**

**Title of practical:**

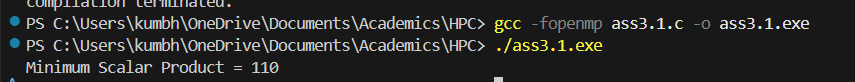
Study and Implementation of schedule, nowait, reduction, ordered and collapse clauses

**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:**

**OpenMP** #pragma omp parallel for distributes loop iterations across threads for faster execution.

**Vector Scalar Addition**: Perfectly parallelizable, speedup increases with more threads until memory bandwidth becomes the bottleneck.

**Minimum Scalar Product:** Needs sorting first (can use parallel sort for large datasets), then uses a parallel reduction for summing.

**Performance Tip:** Test with OMP\_NUM\_THREADS=1,2,4,8 and different SIZE to see scaling.

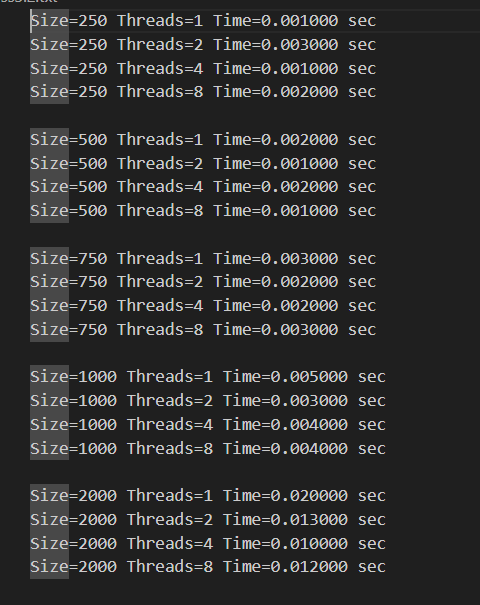
**Problem Statement 2:**

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:**

****

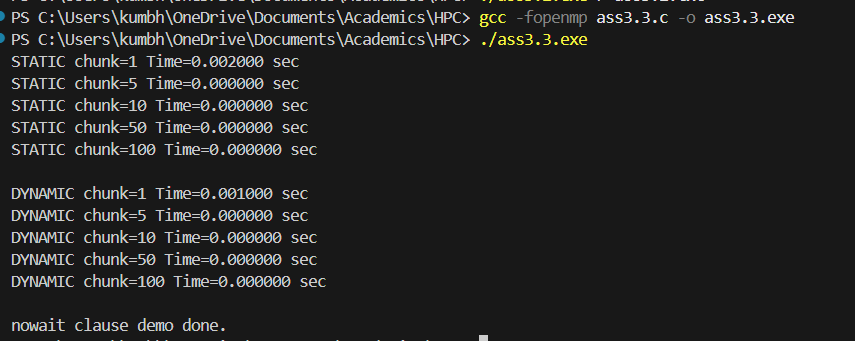
**Information and analysis:**

* For small problems, parallelism may not help and can even slow things down.
* For larger problems, using more threads usually speeds up computation, but not always proportionally, due to parallel overhead and hardware limits.
* This output helps you see the effect of parallelism and the importance of choosing the right number of threads for your problem size

**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:** Better for uniform workloads (all iterations take same time).
* **DYNAMIC:** Better when iteration times vary.
* **Chunk size:** Small chunks improve load balance but increase overhead; largechunks reduce overhead but risk imbalance.
* **nowait:** Avoids unnecessary barriers, improves performance when loops are independent.

**Github Link: https://github.com/Anjali1874/HPC-Lab**