**Session 2025-2026**

| **Vision:** Dream of where you want. | **Mission:** Means to achieve Vision |
| --- | --- |

**Program Educational Objectives of the program (PEO):** (broad statements that describe the professional and career accomplishments)

| PEO1 | **Preparation** | **P: Preparation** | **Pep-CL abbreviation**  **pronounce as Pep-si-lL easy to recall** |
| --- | --- | --- | --- |
| PEO2 | **Core Competence** | **E: Environment (Learning Environment)** |
| PEO3 | **Breadth** | **P: Professionalism** |
| PEO4 | **Professionalism** | **C: Core Competence** |
| PEO5 | **Learning Environment** | **L: Breadth (Learning in diverse areas)** |

**Program Outcomes (PO):** (statements that describe what a student should be able to do and know by the end of a program)

**Keywords of POs:**

Engineering knowledge, Problem analysis, Design/development of solutions, Conduct Investigations of Complex Problems, Engineering Tool Usage, The Engineer and The World, Ethics, Individual and Collaborative Team work, Communication, Project Management and Finance, Life-Long Learning

**PSO Keywords:** Cutting edge technologies, Research

“I am an engineer, and I know how to apply engineering knowledge to investigate, analyse and design solutions to complex problems using tools for entire world following all ethics in a collaborative way with proper management skills throughout my life.” *to contribute to the development of cutting-edge technologies and Research*.

**Integrity:** I will adhere to the Laboratory Code of Conduct and ethics in its entirety.

**Name and Signature of Student and Date**

(Signature and Date in Handwritten)

| **Session** | **2025-26 (ODD)** | | **Course Name** | **HPC Lab** | |
| --- | --- | --- | --- | --- | --- |
| **Semester** | **7** | | **Course Code** | **22ADS706** | |
| **Roll No** | **35** | | **Name of Student** | **Aman Makode** | |
|  |  | |  |  |  |
| Practical Number | | **Practical 4** | | | |
| Course Outcome | | Upon successful completion of the course the students will be able to design, implement, and analyze a parallel algorithm using the OpenMP framework. You will also be able to measure and compare the performance of serial and parallel programs, demonstrating an understanding of concepts like speedup and computational efficiency. | | | |
| Aim | | **Matrix Multiplication using OpenMP** | | | |
| Problem Definition | | Write and execute two C programs to perform the multiplication of two square matrices, A and B, of size NtimesN to produce a result matrix C.   1. **Serial Version:** A standard, single-threaded implementation. 2. **Parallel Version:** An implementation using OpenMP directives to parallelize the computation across multiple threads.   Measure the execution time for both versions with a large matrix size (e.g., N=500 or N=1000) and calculate the performance gain. | | | |
| Theory | | **Matrix multiplication** is a fundamental operation in linear algebra. For two matrices A and B of size NtimesN, the resulting matrix C is calculated as:    This operation has a time complexity of O(N3), making it computationally intensive and an excellent candidate for parallelization.  **OpenMP** (Open Multi-Processing) is an application programming interface (API) that supports multi-platform shared-memory parallel programming in C, C++, and Fortran. It uses a set of compiler directives, library routines, and environment variables. The key directive used here is #pragma omp parallel for, which automatically distributes the iterations of a for loop among multiple threads. The collapse(2) clause is used to parallelize nested loops. For accurate timing in a parallel context, OpenMP provides the omp\_get\_wtime() function, which returns wall-clock time.  **Performance Speedup** is a metric used to quantify the performance gain from parallelization. It is calculated as the ratio of the execution time of the serial program to the execution time of the parallel program.  Speedup=Tparallel​ /Tserial​​ | | | |
| Procedure and Execution : | | **Step 1: Create the Serial Program (matmul\_serial.c)** Write a C program for standard matrix multiplication. Use the clock() function to time the core matrix multiplication logic.  **Step 2: Compile and Run the Serial Version** Compile and run the program from the terminal.  **Step 3: Create the Parallel Program (matmul\_openmp.c)** Modify the serial code by including the <omp.h> header and adding the #pragma omp parallel for collapse(2) directive. Use omp\_get\_wtime() for timing.  **Step 4: Compile and Run the Parallel Version :** Compile using the -fopenmp flag. Set the number of threads using an environment variable before running. | | | |
| Code: | | | |
| Output: | | | |
| Output Analysis : | | After running both programs with N=500, you might observe outputs similar to this:   * **Serial Output:** Serial MatMul elapsed time: 10.851245 seconds * **Parallel Output:** OpenMP MatMul elapsed time: 2.945876 seconds   From these results, the **speedup** can be calculated:  Speedup=2.94587610.851245​≈3.68  The analysis shows that the parallel version is approximately **3.68 times faster** than the serial version when using 4 threads. The speedup is not a perfect 4x due to overheads such as thread creation, synchronization, and memory access patterns. However, it clearly demonstrates a significant performance improvement. | | | |
| Link of student Github profile where lab assignment has been uploaded | | <https://github.com/aman-makode-11/HPCPractical/blob/main/35_P-4_HPC.docx> | | | |
| Conclusion | | This experiment successfully demonstrated the application of OpenMP to parallelize the computationally intensive task of matrix multiplication. The parallel implementation achieved a significant reduction in execution time compared to the serial version, confirming the effectiveness of parallel computing. The calculated speedup was substantial, highlighting how leveraging multi-core processors can greatly enhance the performance of scientific and engineering applications. | | | |
| Plag Report (Similarity index < 12%) | |  | | | |
| Date | | **2/09/2025** | | | |