**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 2** | | | |
| Course Outcome | | Upon successful completion of the course the students will be able to   1. To understand how long a program runs. 2. To identify bottlenecks. 3. To optimize code and compare different implementations. 4. To benchmark HPC applications. | | | |
| Aim | | **Measuring Program Performance** | | | |
| Problem Definition | | Your primary goal is to compare the performance of a serial versus a parallel matrix multiplication program. First, you'll write a standard C program for the multiplication and time its execution. Next, you will modify this code using OpenMP to create a parallel version and measure its new runtime. After compiling both versions, you will run them with the same matrix size to gather performance data. Finally, you'll calculate the speedup to determine how effective the parallelization was. | | | |
| Theory | | The core theory is **parallel computing**, which aims to accelerate program execution by dividing a large problem into smaller, independent sub-tasks. These sub-tasks are then processed simultaneously on multiple processor cores. For a problem like matrix multiplication, where the calculation of each result element is independent, this approach is highly effective.  Frameworks like **OpenMP** provide a simple way to manage this division of labor. However, the maximum achievable speedup is theoretically limited by **Amdahl's Law**, which states that the performance gain is constrained by the portion of the program that must remain serial (i.e., cannot be parallelized). | | | |
| Procedure and Execution : | | Here are the procedure and execution steps are as follows,   1. **Create Serial Program (matmul\_serial.c)**    1. Write a C program for standard N×N matrix multiplication.    2. Use the clock() function to time the calculation loop. 2. **Compile and Run the Serial Version**    1. **Compile:** gcc -o matmul\_serial matmul\_serial.c    2. **Run:** ./matmul\_serial 500 and note the execution time. 3. **Create Parallel Program (matmul\_openmp.c)**    1. Modify the serial code by adding the #pragma omp parallel for collapse(2) directive before the main loops.    2. Replace clock() with OpenMP's omp\_get\_wtime() for accurate parallel timing. 4. **Compile and Run the Parallel Version**    1. **Compile:** gcc -fopenmp -o matmul\_openmp matmul\_openmp.c    2. **Set Threads:** export OMP\_NUM\_THREADS=4 (or your desired core count).    3. **Run:** ./matmul\_openmp 500 and note the new execution time. 5. **Analyze the Results**    1. Compare the two execution times.    2. Calculate the performance gain using the formula: **Speedup = Serial Time / Parallel Time**. | | | |
| Code: | | | |
| Output: | | | |
| Output Analysis : | | The primary output from this experiment consists of two execution times, one for the serial program and one for the parallel one. You will observe that the time reported by the OpenMP version is significantly lower than the baseline serial time. By dividing the serial time by the parallel time, you derive the speedup factor. This value quantifies the performance gain, showing precisely how many times faster the parallel code ran using the specified number of processor cores. | | | |
| Link of student Github profile where lab assignment has been uploaded | | <https://github.com/aman-makode-11/HPCPractical/blob/main/35_P-2_HPC.docx> | | | |
| Conclusion | | This experiment confirms that parallelizing matrix multiplication with OpenMP provides a substantial reduction in runtime. The performance improvement is achieved by efficiently distributing the computational workload across multiple CPU cores to be processed simultaneously. This outcome serves as a practical demonstration of parallel computing principles in solving resource-intensive problems. Ultimately, the exercise validates that for divisible tasks, parallel programming is an essential strategy for enhancing computational efficiency. | | | |
| Plag Report (Similarity index < 12%) | |  | | | |
| Date | | **2/09/2025** | | | |