**Session 2025-2026**

|  |  |
| --- | --- |
| **Vision:** To help businesses uncover crucial  insights | **Mission:** To be a good data scientist |

**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):** 1. Understand and Apply Parallel Programming Concepts

2. Analyse and Improve Program Performance.

3. Demonstrate Practical Skills in HPC Tools and Environments.

**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**

Sujal Kumeriya – 01/09/2025

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Session** | **2025-26 (ODD)** | | **Course Name** | **HPC Lab** | |
| **Semester** | **7** | | **Course Code** | 22ADS706 | |
| **Roll No** | 64 | | **Name of Student** | Sujal Kumeriya | |
|  |  | |  |  |  |
| Practical Number | | 2 | | | |
| Course Outcome | | 1. Understand and Apply Parallel Programming Concepts 2. Analyse and Improve Program Performance | | | |
| Aim | | Measuring Program Performance | | | |
| Problem Definition | | Measuring Program Performance | | | |
| Theory  (100 words) | | **Why measure performance?**  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.  **Common ways to measure program performance in Linux**  **HPC:**  A. Using Linux time command  B. Using built-in timing functions in code (e.g., OpenMP, MPI timing functions)  C. Using profiling tools (basic overview)  **Example: Measuring Performance of Matrix**  **Multiplication**  Step 1: Write the serial (single-threaded) matrix  multiplication code.  Step 2: Compile and run the serial program  Step 3: Add OpenMP parallelization and timing  Step 4: Compile and run the OpenMP version  Step 5: Compare results | | | |
| Code: | | 1. **Matmul\_serial.c**   #include <stdio.h>  #include <stdlib.h>  #include <time.h>  void matmul(int N, double \*A, double \*B, double \*C) {  for (int i = 0; i < N; i++)  for (int j = 0; j < N; j++) {  double sum = 0;  for (int k = 0; k < N; k++)  sum += A[i\*N+k] \* B[k\*N+j];  C[i\*N+j] = sum;  }  }  int main(int argc, char \*\*argv) {  if (argc < 2) {  printf("Usage: %s matrix\_size\n", argv[0]);  return 1;  }  int N = atoi(argv[1]);  double \*A = malloc(N\*N\*sizeof(double));  double \*B = malloc(N\*N\*sizeof(double));  double \*C = malloc(N\*N\*sizeof(double));  // Initialize matrices A and B  for (int i = 0; i < N\*N; i++) {  A[i] = 1.0;  B[i] = 2.0;  }  clock\_t start = clock();  matmul(N, A, B, C);  clock\_t end = clock();  double time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;  printf("Serial MatMul elapsed time: %f seconds\n", time\_spent);  free(A); free(B); free(C);  return 0;  }   1. **Matmul\_openmp.c**   #include <stdio.h>  #include <stdlib.h>  #include <omp.h>  void matmul(int N, double \*A, double \*B, double \*C) {  #pragma omp parallel for collapse(2)  for (int i = 0; i < N; i++)  for (int j = 0; j < N; j++) {  double sum = 0;  for (int k = 0; k < N; k++)  sum += A[i\*N+k] \* B[k\*N+j];  C[i\*N+j] = sum;  }  }  int main(int argc, char \*\*argv) {  if (argc < 2) {  printf("Usage: %s matrix\_size\n", argv[0]);  return 1;  }  int N = atoi(argv[1]);  double \*A = malloc(N\*N\*sizeof(double));  double \*B = malloc(N\*N\*sizeof(double));  double \*C = malloc(N\*N\*sizeof(double));  for (int i = 0; i < N\*N; i++) {  A[i] = 1.0;  B[i] = 2.0;  }  double start = omp\_get\_wtime();  matmul(N, A, B, C);  double end = omp\_get\_wtime();  printf("OpenMP MatMul elapsed time: %f seconds\n", end - start);  free(A); free(B); free(C);  return 0;  } | | | |
| Output | | A screenshot of a computer  AI-generated content may be incorrect.  A screenshot of a computer  AI-generated content may be incorrect. | | | |
| Output Analysis | | The matmul\_serial file executes in 0.4 seconds, whereas the matmul\_openmp files executes in 0.2. There’s a significate improvement in execution time. | | | |
| Link of student Github profile where lab assignment has been uploaded | | **https://github.com/sujalkumeriya59/HPC\_LAB** | | | |
| Conclusion | | Using OpenMP drastically improve the performance of a program. | | | |
| Plag Report (Similarity index < 12%) | | **A screenshot of a chat  AI-generated content may be incorrect.** | | | |
| Date | | 01/09/2025 | | | |