**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**

Shreyas Chaurey – 24/10/2025

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Session** | **2025-26 (ODD)** | | **Course Name** | **HPC Lab** | |
| **Semester** | **7** | | **Course Code** | 22ADS706 | |
| **Roll No** | 61 | | **Name of Student** | Shreyas Chaurey | |
|  |  | |  |  |  |
| Practical Number | | 5 | | | |
| Course Outcome | | 1. Understand and Apply Parallel Programming Concepts 2. Analyse and Improve Program Performance | | | |
| Aim | | Basics of MPI Programming | | | |
| Problem Definition | | Basics of MPI Programming | | | |
| Theory  (100 words) | | Message Passing Interface (MPI) is a standard for parallel programming, specifically designed for high-performance computing and distributed memory systems. It allows processes to communicate with each other by passing messages, which is crucial for working in systems with multiple processors or nodes.  **Basics of MPI Programming**  In MPI programming, processes are distributed across multiple nodes or machines, and they communicate via message-passing mechanisms.  **Key Concepts in MPI:**  **Processes:** Each process is a separate running program with its own memory space.  **Rank:** The unique identifier for each process within a communicator.  **Communicators:** Defines which group of processes can communicate with each other.  **Point-to-Point Communication:** Involves direct communication between two processes.  **Collective Communication:** Involves communication between multiple processes (broadcast, reduce, etc.).  **Types of MPI Communication**  **Point-to-Point:**  MPI\_Send / MPI\_Recv for sending and receiving messages between two processes.  **Collective Operations:**  MPI\_Bcast, MPI\_Reduce, MPI\_Gather, etc., to perform communication over groups of processes.  **Steps to Perform MPI Programming Practically**   1. **Install MPI:** sudo yum install openmpi openmpi-devel; export PATH=$PATH:/usr/lib64/openmpi/bin/ 2. **Write a Basic MPI Program (Example: Hello World)** 3. **Compile the MPI Program:** mpicc hello.c -o hello 4. **Run the program:** mpirun -np 4 ./hello | | | |
| Code: | | **hello.c**  #include <stdio.h>  #include <mpi.h>  int main(int argc, char\* argv[]) {      int rank, size;      // Initialize MPI      MPI\_Init(&argc, &argv);        // Get the rank and size of the communicator      MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);  // Get rank of the process      MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);  // Get total number of processes      // Print "Hello World" from each process      printf("Hello from process %d of %d\n", rank, size);      // Finalize MPI      MPI\_Finalize();      return 0;  } | | | |
| Output | | A screenshot of a computer program  AI-generated content may be incorrect.  A screenshot of a computer  AI-generated content may be incorrect. | | | |
| Output Analysis | | Openmpi executes the hello program using with 4 number of processes. | | | |
| Link of student Github profile where lab assignment has been uploaded | |  | | | |
| Conclusion | | The **message-passing paradigm** **(MPI)** is a viable and effective method for developing **distributed-memory parallel programs**. The practical gives us a theoretical understanding of how independent processes communicate and synchronize to solve a common problem, laying the groundwork for tackling more complex parallel computations and exploring performance optimization in future work. | | | |
| Plag Report (Similarity index < 12%) | |  | | | |
| Date | | 24/10/2025 | | | |