

**MIT Art Design and Technology University**

**MIT School of Computing, Pune**

**Department of Computer Science and Engineering**

|  |
| --- |
| **Lab Report** |

# **Course- HPC Lab**

**Class - L.Y. (SEM-I), Core, AIEC**

**Name:**

**Roll No:**

**A.Y. 2024 - 2025**

**Lab Experiment List**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Name of Experiment** | **CO** |
| 1. | Familiarization with Linux commands | CO1 |
| 2. | Familiarization with SLURM commands | CO1 |
| 3. | Write an OpenMP program to print Hello world with thread ID. | CO2 |
| 4. | Write your first Parallel Program, with which you should be able to print your NAME from 4 underline cores. | CO2 |
| 5. | Write a C program utilizing OpenMP directives to demonstrate the behavior of the private clause.   * + - * The program should perform the following steps:       * Initialize OpenMP with 4 threads.       * Declare an integer variable val and initialize it to a value of 1234.       * Print the initial value of val outside the OpenMP parallel region.       * Enter an OpenMP parallel region using the omp parallel directive, with the firstprivate clause applied to the variable val.       * Inside the parallel region, each thread should print the current value of val, increment it by 1, and then print the updated value.       * Print the final value of val outside the parallel region. | CO2 |
| 6. | Write a C program utilizing OpenMP directives to demonstrate the behavior of the private clause.  Steps to follow :   * + - * Open text editor.       * write the below program in it.       * Save the file with .c extentation.       * Compile and execuate with given commands. | CO2 |
| 7. | Write a Parallel C program where the iterations of a loop should be scheduled statically across the team of threads. A thread should perform CHUNK iterations at a time before being scheduled for the next CHUNK of work. | CO2 |
| 8. | Write a Parallel C program which should print the series of 2 and 4. Make sure both should be executed by different threads. | CO2 |
| 9. | Write MPI Program to print "Hello World".  MPI program to send and receive Hello World messages from all other processes to a Root process and print the received messages. | CO3 |
| 10. | MPI program to send two numbers (array elements) per process to a Root process and print the received messages. | CO3 |
| 11. | MPI program to find sum of first N integers using any given number of processes. Example, N=10,000 and no. of processes can be 4 or 8 or 12 etc. | CO4 |
| 12. | MPI program to find sum of n integers on in which processors are arranged in ring topology using MPI point-to-point blocking communication library calls. | CO4 |
| 13. | Write a CUDA program to perform two matrix addition. | CO5 |
| 14. | Write a CUDA program to perform two matrix multiplication. | CO5 |

**List of Hardware / Software Requirements**

* **Hardware requirements:**

1. Laptop/Desktop machine
2. Internet facility

* **Software Requirements**

**Experiment No 1**

## **Experiment Title: Familiarization with Linux commands**

## Problem Statement:

## Familiarization with Linux commands

#### Objectives

#### Theory

#### Conclusion

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 2**

## **Experiment Title: Familiarization with SLURM commands**

## Problem Statement: Familiarization with SLURM commands

**Objectives**

Understand the Basics of SLURM: Gain a fundamental understanding of what SLURM (Simple Linux Utility for Resource Management) is, its purpose, and its components.

Learn Basic SLURM Commands: Familiarize with essential SLURM commands used for job submission, monitoring, and management.

Execute Jobs Using SLURM: Learn how to submit, monitor, and manage jobs using SLURM commands.

Understand Job Scheduling and Resource Allocation: Understand how SLURM schedules jobs and allocates resources.

Gain Practical Experience: Practice using SLURM commands in a real or simulated high-performance computing (HPC) environment.

**Theory**

SLURM (Simple Linux Utility for Resource Management) is an open-source, highly configurable job scheduler used to manage resources in large-scale compute clusters. It is designed to allocate compute resources, schedule jobs, and manage job queues efficiently.

**Key Components of SLURM:**

Slurmctld (Controller Daemon): Manages the entire cluster, handles job queues, and schedules jobs.

Slurmd (Compute Node Daemon): Runs on each compute node and manages job execution on that node.

Slurmdbd (Database Daemon): Manages the accounting and usage information in a database.

Slurm Commands: Tools for users to interact with the SLURM system.

Essential SLURM Commands:

sbatch: Submits a batch script to the SLURM scheduler.

Example: sbatch myscript.sh

salloc: Allocates resources for a job and runs a command in a new shell.

Example: salloc -N 1 -n 4

srun: Used to submit jobs for execution or to launch tasks within a job allocation.

Example: srun myprogram

scancel: Cancels a pending or running job.

Example: scancel job\_id

squeue: Shows the status of jobs in the queue.

Example: squeue -u username

sinfo: Provides information about nodes and partitions.

Example: sinfo

sacct: Displays accounting information for jobs.

Example: sacct -j job\_id

**Conclusion**

Familiarizing yourself with SLURM commands is crucial for efficiently managing and utilizing resources in a high-performance computing environment. Understanding the theory behind SLURM and practicing essential commands will allow you to effectively submit, monitor, and manage jobs on a compute cluster. As you gain more experience, you can explore advanced features of SLURM to optimize job scheduling and resource allocation, ultimately enhancing the performance and efficiency of your computational tasks.

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 3**

## **Experiment Title: OpenMP program 1**

## Problem Statement: Write an OpenMP program to print Hello world with thread ID.

## **Objectives**

### 

### Theory

#### 

#### 

#### Conclusion

## 

## Source Code and Output /Screenshots**:**

## ***(To be provided by the student)***

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 4**

## **Experiment Title: OpenMP Program 2**

Problem Statement: Write your first Parallel Program, with which you should be able to print your NAME from 4 underline cores.

**Objectives**

**Theory**

**Key Concepts**:

### 

### Conclusion

## Source Code and Output /Screenshots:

## ***(To be provided by the student)***

|  |
| --- |
| ***//Source code and output/screenshot should be available here*** |

## 

**Experiment No 5**

## **Experiment Title: OpenMP Program 3**

## Problem Statement:

Write a C program utilizing OpenMP directives to demonstrate the behavior of the private clause.

The program should perform the following steps:

* Initialize OpenMP with 4 threads.
* Declare an integer variable val and initialize it to a value of 1234.
* Print the initial value of val outside the OpenMP parallel region.
* Enter an OpenMP parallel region using the omp parallel directive, with the firstprivate clause applied to the variable val.
* Inside the parallel region, each thread should print the current value of val, increment it by 1, and then print the updated value.
* Print the final value of val outside the parallel region.

## **Objectives**

## **Theory**

## **Conclusion**

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 6**

## **Experiment Title: OpenMP Program 4**

## 

## Problem Statement:

Write a C program utilizing OpenMP directives to demonstrate the behavior of the private clause.

Steps to follow :

Open text editor.

write the below program in it.

Save the file with .c extentation.

Compile and execuate with given commands.

**Objectives**

**Theory**

**Conclusion**

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 7**

## **Experiment Title: OpenMP Program 5**

## 

## Problem Statement: Write a Parallel C program where the iterations of a loop should be scheduled statically across the team of threads. A thread should perform CHUNK iterations at a time before being scheduled for the next CHUNK of work.

**Objectives**

**Theory**

**Key Concepts:**

**Conclusion**

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 8**

## **Experiment Title: OpenMP Program 6**

## 

## Problem Statement:

## Write a Parallel C program which should print the series of 2 and 4. Make sure both should be executed by different threads.

## Objectives

## 

## Theory

Conclusion

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 9**

## **Experiment Title: MPI Program 1**

## 

## Problem Statement: Write MPI Program to print "Hello World".

## MPI program to send and receive Hello World messages from all other processes to a Root process and print the received messages.

### Objectives

### 

### Theory

### 

### Conclusion

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 10**

## **Experiment Title: MPI Program 2**

## 

## Problem Statement: MPI program to send two numbers (array elements) per process to a Root process and print the received messages.

### Objectives

### 

### Theory

### 

### Conclusion

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 11**

## **Experiment Title: MPI Program 3**

## 

## Problem Statement: MPI program to find sum of first N integers using any given number of processes. Example, N=10,000 and no. of processes can be 4 or 8 or 12 etc.

### Objectives

### 

### Theory

#### 

### 

### Conclusion

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 12**

## **Experiment Title: MPI Program 4**

## 

## Problem Statement: MPI program to find sum of n integers in which processors are arranged in ring topology using MPI point-to-point blocking communication library calls.

Objectives

### 

### Theory

### 

### Conclusion

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 13**

## **Experiment Title: CUDA Program 1**

## 

## Problem Statement: Write a CUDA program to perform two matrix additions.

Objectives

### 

### Theory

#### 

### Conclusion

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |

**Experiment No 14**

## **Experiment Title: CUDA Program 2**

## 

## Problem Statement: Write a CUDA program to perform two matrix multiplications.

Objectives

### 

### Theory

### 

### Conclusion

## Source Code and Output /Screenshots:

## *(To be provided by the student)*

|  |
| --- |
| *//Source code and output/screenshot should be available here* |