MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

HIGH PERFORMANCE COMPUTING SYSTEMS LAB – I

I SEM M.Tech.

Computer Science and Engineering

Prepared by

Dr. N Gopalakrishna Kini
Professor, Dept. of CSE
Manipal Institute of Technology, Manipal

Department of Computer Science and Engineering

MANIPAL INSTITUTE OF TECHNOLOGY, MAHE, MANIPAL

2023

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that Ms./Mr. ……

Reg. No. …… has satisfactorily completed the Computing Lab-1 of First Sem., M Tech [CSE] Degree at MIT, Manipal, in the academic year 2019-2020.

Date: ……

Signature

Faculty in Charge

CONTENTS

LAB NO. WEEK	TITLE
	Course Objectives and Outcomes
	Evaluation plan
	Instructions to the Students
1	OpenMP programming
2	Point to Point Communications in MPI
3	Collective communications in MPI : USE OF MPI_Bcast, MPI_Scatter and MPI_Gather
4	Collective communications in MPI (Contd..): USE OF MPI_Reduce( ), MPI_Scan( )
5	Mini project in MPI
6	CUDA programs on Vectors
7	CUDA programs on Strings
8	CUDA programs on Matrices
9	CUDA programs on Sorting
10	Mini project in CUDA
11	Lab Examination
12	Demo of Mini project

Note:

Online exercise means work to be carried on during the lab hour.

Offline exercise stands for work to be carried on from 'Lab No. Week *n*' to 'Lab No. Week *n* + 1' during off time.

Course Objectives

- Learn to write OpenMP programs, and understand the appropriate usage of MPI point to point, collective communication and time computation.
- Use CUDA APIs to write host and kernel code for different parallel patterns
- To develop the skills of designing parallel algorithms and implement small projects using different parallel programming environment

Course Outcomes

At the end of this course, students will be able to

- Develop OpenMP codes to run it on shared memory system.
- Implement MPI programs using point-to-point and collective communication primitives.
- Develop CUDA programs for different parallel applications
- Demonstrate the skill in parallel programming by developing mini projects

Evaluation plan

- Internal Assessment Marks : 60%
  - Continuous Evaluation : 40%

    The assessment will depend on punctuality, program execution, maintaining the observation note and answering the questions in viva voce.

  - Project Evaluation + Paper submission to Conference / journal : 20%
- End semester assessment of 2 hours: 40 %
- Marks distribution:

Week number	Evaluation Marks
Week 1 & Week 2	10
Week 3 & Week 4	10
Week 6 & Week 7	10
Week 8 & Week 9	10
Week 11 & Week 12 (Mini project evaluation)	20

End Sem assessment: i)Program writeup ii)Program execution	 20 20
Total Marks	100

INSTRUCTIONS TO THE STUDENTS

Pre- Lab Session Instructions

1. Students should carry the Lab Manual Book and the required stationery to every lab session
2. Be in time and follow the institution dress code
3. Must Sign in the log register provided
4. Make sure to occupy the allotted seat and answer the attendance
5. Adhere to the rules and maintain the decorum
6. Students must come prepared for the lab in advance

In- Lab Session Instructions

- Follow the instructions on the allotted exercises
- Show the program and results to the instructors on completion of experiments
- On receiving approval from the instructor, copy the program and results in the Lab record
- Prescribed textbooks and class notes can be kept ready for reference if required

General Instructions for the exercises in Lab

- Implement the given exercise individually and not in a group.
- The programs should meet the following criteria:
    - Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
    - Observation book should be complete with program, proper input output clearly showing the parallel execution in each process.
    - For comparing time of execution consider very large data size as input.
- Plagiarism (copying from others) is strictly prohibited and would invite severe penalty in evaluation.
- In case a student misses a lab class, he/ she must ensure that the experiment is completed during the repetition lab with the permission of the faculty concerned.
- Questions for lab examination are not necessarily limited to the questions in the manual, but may involve some variations and / or combinations of the questions.

THE STUDENTS SHOULD NOT

- Bring mobile phones or any other electronic gadgets to the lab.
- Go out of the lab without permission.

Lab No. Week 1

OpenMP PROGRAMMING

1. Write an OpenMP program to use variables as shared or private.
2. Write an OpenMP program to sum the respective elements of the two arrays
   i) using number of threads equal to the number of CPU cores.
   ii) using number of threads irrespective of number of CPU cores.
3. Write an OpenMP program to find the sum of integers from 1 to N
   i) using parallel for loop.
   ii) using reduction clause in parallel for loop.
4. Write an OpenMP program to parallelize nested for loop for any program.

Offline exercise: A group of TWO students will be formed and start doing a literature survey on a conference / journal paper to be implemented as a mini project in MPI. For literature survey student need to refer MIT library portal.

Lab No. Week 2

MPI PROGRAMMING

USE OF POINT TO POINT COMMUNICATION

1. Write a program in MPI to simulate simple calculator. Perform each operation using different processes in parallel.

2. Write a MPI program using synchronous send. The sender process sends a word to the receiver. The second process receives the word, toggles each letter of the word and sends it back to the first process. Both process use synchronous send operations.

3. Write a MPI program to add an array of size *N* using two processes. Print the result in the root process.

4. Write a MPI program to read *N* elements of an array in the root. Search a number in this array using root and another process. Print the result in the root.

5. Write a MPI program to find the prime numbers between 1 and 100 using two processes.

Online exercise: The group will submit the hardcopy of the conference / journal paper selected by them for implementation to the lab faculty.
Offline exercise: Group will continue literature survey of supporting papers of the paper they submit.

Lab No. Week 3

MPI PROGRAMMING

USE OF MPI_Bcast, MPI_Scatter and MPI_Gather

1) Write a MPI program to read N values in the root process. Root process sends one value to each process. Every process receives it prints the factorial of that number. Use N number of processes.
2) Write a MPI program to read a value M and N x M elements in the root process. Root process sends M elements to each process. Each process finds average of M elements it received and sends these average values to root. Root collects all the values and finds the total average. Use N number of processes.
3) Write a MPI Program to read two strings S1 and S2 of same length in the root process. Using *N* process including the root (string length is evenly divisible by N), produce the concatenated resultant string as shown below. Display the resultant string in the root process.

   Eg: String S1: string String S2: length Resultant String : slternigntgh

4) Write a program to read a value M and N x M number of elements in the root. Using N processes do the following task. Find the square of first M numbers, Find the cube of next M numbers and so on. Print the results in the root.

Offline exercise: The group will start implementing the same conference / journal paper as a mini project in MPI.

Lab No. Week 4

MPI PROGRAMMING

USE OF MPI_Reduce( ), MPI_Scan( )

1) Write a MPI program using N processes to find 1! + 2! +…..+N!. Use MPI_Scan. Calculate the amount of time taken by each process and the whole program.
2) Write a MPI program to read a 3 x 3 matrix. Enter an element to be searched in the root process. Find the number of occurrences of this element in the matrix using three processes.
3) Write a MPI program to read 4 x 4 matrix display the following output using four processes

I/p matrix:					O/p matrix:				
	1	2	3	4		1	2	3	4
	1	2	3	1		2	4	6	5
	1	1	1	1		3	5	7	6
	2	1	2	1		5	6	9	7

4) Write a MPI program to read a word of length N. Using N processes including the root get output word with the pattern as shown in example. Display the resultant output word in the root. Calculate the amount of time taken by each process and the whole program.
   Eg: Input : PCAP Output : PCCAAAPPPP

Online and offline exercise: After completing above programs, the group will continue working with mini project in MPI.

Lab No. Week 5

Mini project in MPI

Online and Offline exercise: The group will continue with the implementation of mini project.

Lab No. Week 6

CUDA PROGRAMMING

CUDA programs on Vectors

1) Write a CUDA program to read two arrays A and B of same size N. Find the sum of corresponding array elements. Store the result in array C.
2) Write a CUDA program which takes N number of decimal values as input. It converts these values into their corresponding octal values and stores the result in another array in parallel.
3) Write a CUDA program which takes an integer array with N number of values and it modifies that array by swapping alternative elements in that same array in parallel.

Online exercise: A Demo of mini project implemented is to be given by the group to the Lab faculty.

Offline exercise: The same group of TWO students start doing a literature survey on a conference / journal paper to be implemented as a mini project in CUDA. For literature survey student need to refer MIT library portal.

OR

the same paper implemented in MPI can also be extended as mini project in CUDA.

Lab No. Week 7

CUDA PROGRAMMING

CUDA programs on Strings

1) Write a CUDA program which reads a string as input and it toggles every character of this string in parallel. Store the resultant string in another character array. Also find the execution time taken by kernel and the whole program.
2) Write a CUDA program which reads a string as input and produces resultant string which is having characters obtained by reversing ASCII value of corresponding character in the original string.
3) Write a CUDA program that takes a string S as input and one integer value N. Produces output string N times as follows in parallel:
   I/p: S = Hello N = 3 O/p String: HelloHelloHello
4) Write a CUDA program which reads a string consisting of N words and reverse each word of it in parallel.

Online and offline exercise: The group will start implementing the paper as a mini project in CUDA.

Lab No. Week 8

CUDA PROGRAMMING

CUDA programs on Matrices

1. Write a CUDA program to read matrix A of size M x N, read matrix B of size N x P. Perform C = A x B. Produce a resultant matrix C of size M x P in parallel. . Also find the execution time taken by kernel and the whole program.
2. Write a program in CUDA to add two Matrices using
   a. Each row of resultant matrix to be computed by one thread.
   b. Each column of resultant matrix to be computed by one thread.
   c. Each element of resultant matrix to be computed by one thread.
3. Write a CUDA program to read a matrix A of size N x N. It replaces the principal diagonal elements with zero. Elements above the principal diagonal by their factorial and elements below the principal diagonal by their sum of digits.
4. Write a program in CUDA to implement matrix addition and multiplication using 2D grids and 2D blocks.

Online and offline exercise: The group will continue implementing the paper as a mini project in CUDA.

Offline exercise: Download the IEEE paper template. The group will prepare a technical paper on their mini project according to the IEEE paper format.

Lab No. Week 9

CUDA PROGRAMMING

CUDA programs on Sorting

1. Write a program in CUDA to sort a given string using selection sort.
2. Write a program in CUDA to sort each word of a given string using selection sort.
3. Write a CUDA program to read an integer array of N numbers. Sort this array using odd-even transposition sorting. (Use 2 kernels).

Online and offline exercise: The group will continue implementing the paper as a mini project in CUDA.
Offline exercise: The group will continue with the preparation of the technical paper on their mini project. The group will consult with the faculty to communicate this paper to a conference / journal.

Lab No. Week 10

Mini project in CUDA

Online and Offline exercise: The group will continue with the implementation of mini project in CUDA.
Offline exercise: Plagiarism check is to be done for student paper. Faculty will submit this paper to a conference / journal as student paper. Approval from lab faculty is required before submitting the paper for a conference / journal.

Lab No. Week 11

Lab Examination.

Online exercise: Follow up of paper submitted to Conference / journal.

Lab No. Week 12

A Demo of mini project implemented in CUDA is to be given by the group to the Lab faculty.

Online exercise: Follow up of paper submitted to Conference / journal.

References:

1. Michael J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw Hill, 2003.
2. David B. Kirk, Wen-mei W. Hwu, Pragramming Massively Parallel Processors, A Hands-on Approach, (2e),Elsevier, 2012.