IPS Academy, Institute of Engineering & Science

(A UGC Autonomous Institute, Affiliated to RGPV, Bhopal)
Scheme Based on AICTE Flexible Curriculum

Bachelor of Technology (B.Tech.)

Department of Computer Science & Engineering-AIML

Honors Certification in High Performance Computing

(For Students of CSE-AIML)

Sr. No.	Course Type	Course Code	Course Name	Teaching Scheme			Credits
				L	T	P	
1.	PCC	HOCL 01	Introduction to Linux	2	1	2	4
2.	PCC	HOCL 02	Parallel Programming	2	1	2	4
3.	PCC	HOCL 03	GPU with CUDA	2	1	2	4
4.	PCC	HOCL 04	High Performance Computing	2	1	0	3
Total Academic Engagement and Credits				8	4	6	15
				18			

IPS Academy, Institute of Engineering & Science

(A UGC Autonomous Institute, Affiliated to RGPV, Bhopal)
Scheme Based on AICTE Flexible Curriculum

Bachelor of Technology (B.Tech.)

Department of Computer Science & Engineering-AIML

PCC- HOCL 01	**Introduction to Linux**	**3L: 0T:2P (5hrs.)**	**Credits:04**

Prerequisite: Operating System.

Course Objective: The objective of this course is to understand fundamental Linux concepts.

Course Contents: (40 hrs.)

Module 1: **(06 hrs.)**

What is Linux, Everyday use of Linux, Unix vs. Linux. Differentiate between Unix and Linux, explain the everyday uses of Linux and identify key characteristics of the Linux operating system.

Module 2: **(08 hrs.)**

What is Oracle Virtual Box, Downloading and Installing Oracle Virtual Box, Creating virtual machine, Linux Distributions, Different way to install Linux, Downloading and Installing Linux (CentOS), Redhat Linux installation Linux Desktop (GUI), Virtual Machine Management Linux vs.

Module 3: **(08 hrs.)**

Accessing Linux system Download and install Putty, New Network Commands (ifconfig and ip), Connect Linux VM via Putty, Important Things to Remember in Linux. Introduction to File System, File system structure description, File system navigation commands, File System Paths, Directory listing overview, Creating Files and Directories, Linux File Types, Finding Files and Directories (find, locate) Difference between find and locate command, Changing Password, Wildcard (*, $, ^), Soft and Hard Links (ln), How to open image file through GUI.

Module 4: **(10 hrs.)**

Linux Fundamentals Commands Syntax, File Permissions (chmod), File Ownership (chown, chgrp), Getting Help (man, whatis etc.), TAB completion and up arrow keys, Adding text to file, Standard output to a file (tee command), Pipes (|), File Maintenance Commands, File Display Commands, Filters / Text Processing Commands (cut, sort, grep, awk, uniq, wc), Compare Files (diff, cmp), Compress and un-compress files/directories (tar, gzip, gunzip),Truncate file size (truncate), Combining and Splitting Files (cat and split), Linux vs. Windows Commands.

Module 5: **(08 hrs.)**

Shell Scripting Linux Kernel, What is a Shell, Types of Shells, Shell scripting, Basic Shell scripts, If-then scripts, For loop scripts, Do-while scripts, Case statement scripts, Aliases, Command history.

Course Outcomes:

1. Understand the basic concepts of Linux and differentiate it from other operating systems.
2. Understand the concept of virtualization and Oracle Virtual Box.
3. Access a Linux system remotely using Putty.
4. Understand Linux command syntax.
5. Understand the role of the Linux kernel and shell.

List of Text / Reference Books:

1. The Linux Command Line: A Complete Introduction by William E. Shotts Jr.
2. The Linux Programming Interface: A Linux and UNIX System Programming Handbook by Michael Kerrisk.
3. Linux Kernel Development by Robert Love.
4. UNIX and Linux System Administration Handbook by Evi Nemeth.
5. Advanced Programming in the UNIX Environment by W. Richard Stevens.
6. The Linux Programming Interface *by Michael Kerrisk*

List of Experiment

1. Write a program to copy a file using read and write system calls. Compare its performance with the cp command.
2. Use truncate or ftruncate to change the size of a file.
3. Create a program that forks a child process using fork. The parent and child processes should execute different code.
4. Use wait and waitpid to synchronize parent and child processes. Experiment with different wait options.
5. Use the kill command or the kill system call to send signals to other processes.
6. Create a pipeline of processes using pipe and communicate between them using read and write.
7. Create and use FIFOs to communicate between unrelated processes.
8. Create multiple threads using pthread_create and manage their termination using pthread_join.
9. Experiment with terminal I/O in non-canonical mode, reading characters as they are typed without waiting for a newline.
10. Write a program to create a daemon process that runs in the background.

IPS Academy, Institute of Engineering & Science

(A UGC Autonomous Institute, Affiliated to RGPV, Bhopal)
Scheme Based on AICTE Flexible Curriculum

Bachelor of Technology (B.Tech.)

Department of Computer Science & Engineering-AIML

PCC- HOCL 02	Parallel Programming	3L: 0T:0P (3hrs.)	Credits:03

Pre requisite(s):

Course Objective: The objective of this course is to provide Comprehend fundamental concepts of parallel computing.

Course Contents: (40 Hours)

Module 1: **(06 hrs)**

Parallel Computing, Von Neumann architecture, Processes, Multitasking and threads, Cache mappings, Virtual Memory, SIMD system, MIMD System, Cache coherence, Shared memory versus Distributed Memory.

Module 2: **(08 hrs)**

Parallel Software: Caveats, Coordinating the processes/threads. Performance: Speedup and efficiency, Amdahl's law, Scalability.

Module 3: **(08 hrs)**

Distributed Memory with MPI: Introduction, The Trapezoidal Rule in MPI, Dealing with I/O, Collective Communication, MPI derived Data types.

Module 4: **(08 hrs)**

Shared Memory programming with Pthreads, Processes, Threads and Pthreads, Matrix–vector multiplication, Critical section, Producer consumer synchronization and Semaphores, Thread Safety.

Module 5: **(10 hrs)**

Shared Memory programming with OpenMp: Introduction, The Trapezoidal Rule, The parallel for Directive, Scheduling Loops.

Course Outcomes:

After completion of the course the student will be able to

1. Understand the fundamental concepts of parallel computing,
2. Identify the challenges in parallel software development and analyze the performance of parallel programs.
3. Develop parallel programs using MPI for distributed memory systems.
4. Implement shared memory parallel programs.
5. Utilize OpenMP for shared memory programming.

Recommended Books:

1. Parallel Programming in OpenMP by Rohit Chandra.
2. Making OpenMP Simple Again by Tim Mattson, Helen He, Alice Koniges
3. Parallel Programming in C with MPI and OpenMP by Michael J. Quinn
4. Programming Your GPU with OpenMP by Tom Deakin and Tim Mattson
5. High Performance Parallel Runtimes by Michael Klemm and Jim Cownie
6. Using OpenMP – The Next Step by Ruud van der Pas, Eric Stotzer and Christian Terboven

IPS Academy, Institute of Engineering & Science

(A UGC Autonomous Institute, Affiliated to RGPV, Bhopal)

Scheme Based on AICTE Flexible Curriculum

Bachelor of Technology (B.Tech.)

Department of Computer Science & Engineering-AIML

PCC - HOCL 03	GPU with CUDA	3L: 0T:2P (5 hrs.)	Credits:04

Prerequisite: Linux, C/C++ Programming.

Course Objective: The objective of this course is to understand fundamental GPU architecture and parallel programming concepts.

Course Contents: (40 Hours)

Module 1: **(08 hrs)**

Introduction: History, GPU Architecture, Clock speeds, CPU / GPU comparisons, Heterogeneity, Accelerators, Parallel Programming, CUDA OpenCL / OpenACC, Kernels Launch parameters, Thread hierarchy, Warps/Wavefronts, Threadblocks/Workgroups, Streaming multiprocessors, 1D/2D/3D thread mapping, Device properties, Simple Programs.

Module 2: **(06 hrs)**

Memory: Memory hierarchy, DRAM / global, local / shared, private / local, textures, Constant Memory, Pointers, Parameter Passing, Arrays and dynamic Memory, Multi-dimensional Arrays, Memory Allocation, Memory copying across devices, Programs with matrices, Performance evaluation with different memories

Module 3: **(10 hrs)**

Synchronization: Memory Consistency, Barriers (local versus global), Atomics, Memory fence. Prefix sum, Reduction. Programs for concurrent Data Structures such as Worklists, Linked-lists. Synchronization across CPU and GPU Functions: Device functions, Host functions, Kernels functions, Using libraries (such as Thrust), and developing libraries.

Module 4: **(10 hrs)**

Support: Debugging GPU Programs. Profiling, Profile tools, Performance aspects Streams: Asynchronous processing, tasks, Task-dependence, Overlapped data transfers, Default Stream, Synchronization with streams. Events, Event-basedSynchronization - Overlapping data transfer and kernel execution, pitfalls.

Module 5: **(6 hrs)**

Advanced topics: Dynamic parallelism, Unified Virtual Memory, Multi-GPU processing, Peer access, Heterogeneous processing

Course Outcomes:

1. Define terminology commonly used in parallel computing, such as efficiency and speedup.
2. Describe common GPU architectures and programming models.
3. Implement efficient algorithms for common application kernels, such as matrix multiplication.
4. Develop an efficient parallel algorithm to solve it
5. Implement an efficient and correct code to solve it, analyze its performance, and give convincing written and oral presentations explaining the achievements.

List of Text Books/ Reference Books:

1. David Kirk and Wen-mei Hwu, Programming Massively Parallel Processors: A Hands-On Approach
2. Shane Cook, CUDA Programming: A Developer's Guide to Parallel Computing with GPUs, Morgan Kaufman

List of Experiment

1. Write CUDA code to compute the squares of the first N integers.
2. Write CUDA code to determine the data transfer bandwidth from host to device.
3. Write CUDA code to determine data transfer bandwidth from device to host.
4. Write CUDA code to determine data transfer bandwidth from host to device using pinned memory.
5. Write CUDA code to determine data transfer bandwidth from device to host using pinned memory.
6. Write CUDA code to determine kernel creation overhead.
7. Write code for matrix multiplication using shared memory and compare its performance with CPU code.
8. Implement matrix multiplication on the CPU and GPU

IPS Academy, Institute of Engineering & Science

(A UGC Autonomous Institute, Affiliated to RGPV, Bhopal)

Scheme Based on AICTE Flexible Curriculum

Bachelor of Technology (B.Tech.)

Department of Computer Science & Engineering-AIML

PCC - HOCL 04	High Performance Computing	3L: 0T:2P (5 hrs.)	Credits:04

Prerequisite: NA

Course Objective:

The objective of this course is to provide an understanding of the technologies and the standards relating to the High Performance Computing.

Course Contents: (40 Hours)

Module 1: **(06 hrs)**

Parallel Programming & Computing – Introduction: Era of Computing, Parallel Computing, Multiprocessors and Multicomputer Architectures, Scalar VS Vector Processing, Multi vector and Superscalar Machines, Pipelined Processors, SIMD Computers, Conditions of parallelism, Program flow mechanisms, Types of Parallelism – ILP, PLP, LLP, Program Partitioning and scheduling.

Module 2: **(10 hrs)**

Introduction to High Performance Computing, Scalable Parallel Computer Architectures, towards low-cost computing, Network of Workstations project by Berkeley, Cluster Computing Architecture, Components, Cluster Middleware and SSI, Need of Resource Management and Scheduling, Programming Environments.

Module 3: **(08 hrs)**

Clustering Models, Clustering Architectures, Clustering Architectures key factors, types of clusters, Mission critical Vs Business Critical Applications, Fault Detection and Masking Algorithms, Check pointing, Heartbeats, Watchdog Timers, Fault recovery through Failover and Failback Concepts.

Module 4: **(10 hrs)**

High Speed Networks & Message Passing ,Introduction to High-Speed Networks, Lightweight Messaging Systems, Xpress Transport Protocol, Software RAID and Parallel File systems, Load Balancing Over Networks – Algorithms and Applications, Job Scheduling approaches and Resource Management in Cluster.

Module 5: **(06 hrs)**

CUDA Programming: Introduction to CUDA architecture for parallel processing, CUDA Parallelism Model, Foundations of Shared Memory, Introduction to CUDA-C, Parallel programming in CUDA-C, Thread Cooperation and Execution Efficiency, Constants memory and events, memory management, CUDA C on multiple GPUs, Hashing and Natural Parallelism, Scheduling and Work Distribution, Atomics, Barriers and Progress, Transactional Memory

Course Outcomes:

1. Understand the fundamental concepts of parallel computing.
2. Understand High Performance Computing (HPC) system architectures and various computational models.
3. Apply parallel execution models and methodologies for parallel programming and parallel applications development.
4. Design and implement compute intensive applications on HPC platform.
5. Learn basics of CUDA programming.

List of Text Books/ Reference Books:

1. Rajkumar, High Performance Cluster Computing: Architectures and Systems, Vol. 1 Pearson Education.
2. Georg Hager and Gerhard Wellein, Introduction to High Performance Computing for Scientists and Engineers, CRC Press.
3. Kai Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw Hill International Editions.
4. "Introduction to High Performance Computing for Scientists and Engineers" by Georg Hager and Gerhard Wellein.
5. "Parallel Programming in C with MPI and OpenMP" by Michael J. Quinn.

List of Experiments

1. Write a simple MPI program to print "Hello, World!" from each process
2. Implement a program where two processes exchange data using MPI_Send and MPI_Recv.
3. Implement a program that uses MPI_Bcast to broadcast data from one process to all others.
4. Implement a program that uses MPI_Reduce to combine data from all processes on one process (e.g., sum, max, min).
5. Write an MPI program and use MPI_Wtime to measure the execution time for different numbers of processes.
6. Write an OpenMP program that uses both shared and private variables.
7. Implement a program that requires synchronization between threads using OpenMP constructs such as #pragma omp critical, #pragma omp atomic, and #pragma omp barrier.
8. Use OpenMP's reduction clause to efficiently combine the results of a parallel loop (e.g., sum, product).
9. Implement a parallel sorting algorithm (e.g., parallel merge sort, parallel quicksort) using either MPI or OpenMP.
10. Create and use derived datatypes (e.g., MPI_Struct) to exchange non-contiguous data between processes.