CSEB432: HIGH PERFORMANCE COMPUTING

L: 3 T: 0 P: 2, Credits: 4

Course Outcomes (COs)

CO1: Understand fundamental principles and importance of high performance and parallel computing.
CO2: Apply parallel programming skills using C++, Python, and CUDA to solve computational problems.
CO3: Demonstrate practical skills in concurrent programming, GPU programming, memory management, and optimization.
CO4: Utilize advanced GPU and HPC libraries (cuBLAS, cuFFT, Thrust, cuDNN) for scientific, engineering, and AI/ML workloads.
CO5: Design, implement, and optimize large-scale scientific and enterprise computing solutions leveraging modern HPC platforms.
CO6: Utilize CUDA libraries to solve scientific and machine learning problems with GPU acceleration.

Unit 1: Foundations of High Performance and Parallel Programming
Course Overview & Expectations: Introduction to HPC, course structure, assessment, outcomes. Programming Background: Intro to C++ and Python, development tools (VS Code, GitHub).Concurrency Concepts: Why parallelism? Serial vs. parallel execution, Flynn's taxonomy.Basic Concurrent Programming Patterns: Producer-consumer, dining philosophers, sleeping barber — concepts and implementations. Hands-On: Simple C++ and Python concurrency labs, discussion prompts on general programming experience.

Unit 2: CPU Parallelism & Programming Models
CPU Parallelism in Modern Languages: Leveraging threads and cores in C++ and Python.
Concurrent Programming Structures: Syntax and design patterns for multi-threaded code, pitfalls, and practical considerations. Practical Labs: Python 3 and C++ parallel labs; assignments encouraging hands-on code. Discussion: Single vs. multithreaded programming, programming experience sharing.

Unit 3: GPU Fundamentals and CUDA Programming Essentials
GPU Hardware Basics: Integrated vs. dedicated GPUs, how to identify and interrogate local GPU hardware. Nvidia GPU Architectures and CUDA Ecosystem: Hardware generations, CUDA software layers, driver/runtime distinctions. CUDA Installation and Tools: Setting up CUDA environments on Linux, exploring NVCC and code compilation. First CUDA Programs: Syntax basics, code structure, IDE integration, kernel execution model. Hands-On: Compiling and running simple CUDA programs, exploring device queries and memory.

Unit 4: CUDA Programming — Data, Memory, and Performance
Threads, Blocks, and Grids: CUDA logical thread structure, multi-dimensional data processing, mapping algorithms. Host and Device Memory Management: CPU vs. GPU memory, allocations, and optimized transfer strategies. Specialized GPU Memory: Usage of shared, constant, and register memory — best practices for mutable and static data, thread communication, and caching. Performance Optimization: Understanding occupancy, memory bottlenecks, and practical memory management. Labs: Memory allocation, RGB to grayscale transformation, image processing, and real-world CUDA

assignments.

Unit 5: Advanced CUDA: Multi-GPU, Streams, and Events
Multiple CPUs/GPUs: Programming models for distributed and multi-GPU systems, inter-device communication, challenges. CUDA Streams & Events: Asynchronous workflows, event handling, enabling real-time and interactive applications. Scaling Up: Principles for enterprise-level deployments, resource management, canonical algorithms on multiple devices, Hands-On: Multiple CPU and GPU labs, stream/event assignments, discussions on distributed strategies.

Unit 6: GPU-Accelerated Libraries and Scientific Computing
CUDA Libraries for Computation: cuFFT (Fast Fourier Transforms), cuBLAS/NVBLAS, cuSPARSE, cuSOLVER for linear algebra. Thrust Library: Higher-level abstractions (vectors, sorting, reductions, transforms). cuDNN & Machine Learning: Accelerating deep learning, image, and signal processing with GPU libraries. Capstone Project: A large-scale HPC challenge involving CUDA, advanced libraries, and real-world datasets.Discussion & Assignments: Algorithm conversion to GPU, image/signal processing, neural networks in CUDA.

S. No.	Topic/Module	To be Completed by
1.	Introduction to Concurrent Programming with GPUs	Week 3
2.	Introduction to Parallel Programming with CUDA	Week 7
3.	CUDA at Scale for the Enterprise	Week 11
4.	CUDA Advanced Libraries	Week 14

Examination Category: 11 (Mid Term Examination: All MCQ and End Term Examination: All MCQ).
Weightage: Mid Term Examination 40% Marks and End Term Examination 60% Marks.
Continuous Assessment: Not Applicable (Only Mid Term and End Term exams will be conducted).

- **Eligibility Criteria for Mid-Term Examinations**

A student will be eligible for the Mid-Term Examination only if they successfully complete the following courses and submit the corresponding certificates before the end of the 7th week of the semester. The required courses are:

1. Introduction to Concurrent Programming with GPUs
2. Introduction to Parallel Programming with CUDA

Certificates of completion for both courses must be submitted on or before 10th September 2025; the courses are listed here**:**

1. **Introduction to Concurrent Programming with GPUs:** https://www.coursera.org/learn/introduction-to-concurrent-programming?specialization=gpu-programming.
2. **Introduction to Parallel Programming with CUDA:** https://www.coursera.org/learn/introduction-to-parallel-programming-with-cuda?specialization=gpu-programming.

- **Eligibility Criteria for End-Term Examinations**

To be eligible for the End-Term Examination, a student must successfully complete and submit Coursera completion certificates for all four courses included across the six units of the syllabus on or before 16th November 2025. The required courses are listed below:

1. **Introduction to Concurrent Programming with GPUs:** https://www.coursera.org/learn/introduction-to-concurrent-programming?specialization=gpu-programming
2. **Introduction to Parallel Programming with CUDA:** https://www.coursera.org/learn/introduction-to-parallel-programming-with-cuda?specialization=gpu-programming
3. **CUDA at Scale for the Enterprise:** https://www.coursera.org/learn/cuda-at-scale-for-the-enterprise?specialization=gpu-programming
4. **CUDA Advanced Libraries:** https://www.coursera.org/learn/cuda-advanced-libraries?specialization=gpu-programming.

Course link: https://www.coursera.org/specializations/gpu-programming