INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

HIGH PERFORMANCE COMPUTING								
VII Semester: CSE								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
		L	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
ACSE27	**Core**	3	0	0	3	40	60	100
Contact Classes: 48	**Tutorial Classes: Nil**	**Practical Classes: Nil**				**Total Classes: 48**		
Prerequisite: Operating systems								

I. COURSE OVERVIEW:

This course provides a comprehensive introduction to High-Performance Computing (HPC), covering the principles, architectures, and techniques used to solve computationally intensive problems. Students will explore parallel computing models, distributed computing, and GPU acceleration to optimize performance in scientific computing, engineering, and data-intensive applications. The course also includes hands-on experience with MPI (Message Passing Interface), Open MP, and CUDA programming.

II. COURSES OBJECTIVES:

The students will try to learn

I. *The* parallel computing principles, algorithm design methodologies including partitioning, task scheduling, load balancing, and structured parallel models used in modern parallel computers.
II. *The* performance modeling fundamentals such as scalability, overhead, bandwidth, efficiency, and to understand algorithmic case studies like shortest path solutions derived from methods including Floyd's algorithm.
III. *The* C/C++-based concurrency constructs, synchronization, locality, communication, and remote operations supported in development environments provided by vendors such as IBM.
IV. *The* message-passing programming models and practical bindings using libraries like MPI and performance analysis ecosystems such as Pablo.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

CO1 Design and implement parallel algorithms (sorting, graph, reduction, network-aware methods) while selecting suitable partitioning and scheduling strategies.
CO2 Model and evaluate parallel system performance using parameters including scalability, overhead, bandwidth, speedup, and efficiency.
CO3 Develop concurrent programs in C/C++ using synchronization, mutual exclusion, and data transfer mechanisms with locality-aware thread/processor placement.
CO4 Apply message-passing programming using C and Fortran bindings with global and asynchronous operations through libraries such as MPI.
CO5 Analyze interconnection network impact including efficiency trade-offs in hypercube-style communication using models like Hypercube algorithms.
CO6 Use structured performance profiling and analysis workflows through tools such as Pablo, instrumentation traces, data collection, transformation, and visualization.

IV. COURSE CONTENT:

Week 1: Introduction to Parallel Computing

1. Study parallel computers and computation.
2. Identify characteristics of parallel architectures.
3. Analyze differences between serial and parallel execution using sample programs.

Week 2: Parallel Machine and Programming Models

1. Implement basic parallel programming models (shared memory vs distributed memory).
2. Study PRAM, SIMD, MIMD models through simple examples.

Week 3: Parallel Algorithm Design Methodology

1. Implement examples of parallel algorithms.
2. Apply partitioning, communication, agglomeration, and mapping (PCAM) methodology.

Week 4: Load Balancing and Task Scheduling

1. Implement static and dynamic load balancing algorithms.
2. Analyze task scheduling strategies and their performance.

Week 5: Case Studies on Parallel Algorithms

1. Implement random number generation in parallel.
2. Study hypercube algorithms with small-scale simulation.

Week 6: Vector and Matrix Operations

1. Implement parallel vector reduction.
2. Implement parallel matrix transposition.
3. Analyze execution time improvement.

Week 7: Parallel Sorting Algorithms

1. Implement parallel merge sort.
2. Compare performance with serial merge sort.

Week 8: Performance Modeling Fundamentals

1. Define execution time, speedup, efficiency.
2. Develop basic performance models.

Week 9: Scalability and Overhead Analysis

1. Study scalability, bandwidth, communication overheads.
2. Analyze I/O impact on parallel performance.

Week 10: Performance Modeling Case Studies

1. Implement shortest path algorithms.
2. Implement Floyd’s algorithm.
3. Implement Dijkstra’s algorithm.
4. Perform modular design and performance evaluation.

Week 11: Parallel Computing Development Tools – C/C++

1. Review C/C++ for parallel computing.
2. Implement concurrency using threads.
3. Study locality, processor objects, and global pointers.

Week 12: Synchronization and Communication

1. Implement synchronization primitives (mutex, semaphore).
2. Implement asynchronous communication and remote operations.
3. Analyze determinism and mapping issues.

Week 13: Fortran and Data Parallelism

1. Study Fortran M and High Performance Fortran (HPF).
2. Implement data parallelism and data distribution.
3. Analyze modularity and performance issues.

Week 14: Message Passing and Performance Tools

1. Implement MPI programs using C/Fortran (point-to-point and collective communication).
2. Study performance analysis tools: Paragraph, Upshot, Pablo, Gauge, ParaIDE, IBM Parallel Environment.
3. Perform performance data collection, transformation, and visualization.

V. TEXT BOOKS:

1. Ion Foster, ―Designing and Building Parallel Programs‖, Addison Wesley, 1st Edition, 2003.

VI. REFERENCE BOOKS:

2. Arjen Markus, "Modern Fortran in Practice", Cambridge University Press, 1 st Edition, 2012.
3. Charles H. Koelbe, "High Performance Fortran Handbook", MIT Press, 1 st Edition, 1993.
4. Michael J. Quinn, "Parallel Programming in C with MPI and Open MPI", Tata McGraw-Hill Publishing Company Ltd, 1st Edition, 2003.

VII. ELECTRONICS RESOURCES:

1. http://www.drdobbs.com/parallel/designing-parallel-algorithms-part-1/223100878.
2. http://searchcloudapplications.techtarget.com/tip/How-to-use-application-performance-modelingtechniques.
3. https://computing.llnl.gov/tutorials/parallel_comp/.

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech-talk topics
4. Open-ended experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper – II

9. Lecture notes
10. PowerPoint presentation
11. E-Learning Readiness Videos (ELRV)