Approval: 10th Senate Meeting

Course Name:	Design of Concurrent Software
Course code:	CS546
Credits:	3-0-0-3
Prerequisites:	programming & data structures; basics of operating systems, basics of networks.
Intended for:	BTech, MTech, MSc, MS/PhD in the area.
Elective/Core:	Discipline elective for BTech CSE and EE, free elective for others
Semester:	Any

Preamble:
In concurrent software, several statements or blocks of code execute simultaneously. This is in contrast to a sequential programme where exactly one instruction executes at a time. Concurrency serves two purposes: to handle asynchronous events (eg, mouse click, arrival of network packet, change of temperature) or to improve performance by making use of multiple CPUs. The CPUs could be on one node with shared memory (parallel hardware), or they could be independent nodes, each with its own memory and disk, connected by a network (distributed hardware).

The objectives of this course are to learn the theory and practice necessary for writing efficient parallel solutions for scientific and engineering computation, and efficient concurrent solutions for Big Data processing.

Course Outline:
After a review of parallel and distributed hardware, we will study the principles of concurrency and examine techniques for designing efficient concurrent software. The course will involve theory, implementation using current languages (MPI, Java and Map Reduce) and evaluation of software performance.

Modules:

1. Introduction: Need for concurrent programs; the critical section problem; parallel and distributed architectures.
2. Programming models: levels of parallelism; data distribution for arrays; shared variables and message-passing; processes and threads.
3. Performance evaluation: metrics for parallel programs; design of experiments, measurement techniques, confidence levels.
4. Parallel programming: the MPI message-passing model; point-to-point and collective communication modes; process groups; MPI and Pthreads; testing for correctness and for performance; debugging. Optionally: CUDA/OpenCL for GPU programming.
5. Concurrency in Java: Java memory model; threads; RMI; locking; scalability; selected concurrency design patterns.
6. Big Data Processing: the Map Reduce programming model; Map Reduce architecture and implementations; Map Reduce algorithm design; limitations of the Map Reduce model, extensions to solve these.

Textbooks:

1. T. Rauber & G. Rünger, *Parallel Programming for Multicore and Cluster Systems*, Springer, 2007.
2. B. Goetz *et al*., *Java Concurrency in Practice*, Pearson, 2006.
3. J. Lin & C. Dyer, *Data-Intensive Text Processing with Map Reduce,* Morgan & Claypool, 2010

References:

1. E.D. Lazowska *et al*., *Quantitative System Performance*, Prentice-Hall, 1984.
2. K.S. Trivedi, *Probability and Statistics with Reliability, Queueing and Computer Science Applications*, Prentice-Hall, 1982.
3. Doug Lea, *Concurrent Programming in Java: Design Principles and Patterns*, 2nd ed., Pearson, 2000.
4. M. Subramanian, *Network Management: Principles and Practice*, 2nd ed., Pearson, 2009. (Chap 9.4)
5. D. Kirk and W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach,* 2nd ed., Morgan Kaufmann, 2012.