University of Texas at El Paso
Course Syllabus

COURESE DESCRIPTION

Dept., Number	CS 4175	**Course Title**	Parallel Computing
Approval Date	01/15/2020	**Course Coordinator**	

CATALOG DESCRIPTION

The course covers fundamentals of parallel computing, including principles of parallel decomposition, processes communication and coordination, architecture, parallel algorithms, analysis, and programming.

TEXT BOOK

TBD

COURSE OUTCOMES

Level 1: Knowledge and Comprehension:

Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. Upon successful completion of this course, students will able to:

a. State the goals of parallelism.
b. Distinguish between parallelism and concurrency.
c. Distinguish data races from higher level races.
d. Explain when and why multicast or event-based messaging can be preferable to alternatives.
e. Describe at least one design technique for avoiding liveness failures in programs using multiple locks or semaphores.
f. Describe the relative merits of optimistic versus conservative concurrency control under different rates of contention among updates.
g. Give an example of a scenario in which an attempted optimistic update may never complete.
h. Define “critical path”, “work”, and “span”.
i. Define “speed-up” and explain the notion of an algorithm’s scalability in this regard.
j. Characterize features of a workload that allow or prevent it from being naturally parallelized.
k. Provide an example of a problem that fits the producer-consumer paradigm.
l. Explain the differences between shared and distributed memory.
m. Describe the SMP architecture and note its key features.
n. Characterize the kinds of tasks that are a natural match for SIMD machines.
o. Describe the advantages and limitations of GPUs vs. CPUs.

Level 2: Application and Analysis:

Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details.

Upon successful completion of this course, students will be able to:

a. Identify opportunities to partition a serial program into independent parallel modules.
b. Parallelize an algorithm by applying task-based decomposition.
c. Parallelize an algorithm by applying data-parallel decomposition.
d. Write a program using actors and/or reactive processes.
e. Use mutual exclusion to avoid a given race condition.
f. Use semaphores or condition variables to block threads until a necessary precondition holds.
g. Give an example of a scenario in which blocking message sends can deadlock
h. Write a program that correctly terminates when all of a set of concurrent tasks have completed.
i. Use a properly synchronized queue to buffer data passed among activities.
j. Write a test program that can reveal a concurrent programming error; for example, missing an update when two activities both try to increment a variable.
k. Compute the work and span, and determine the critical path with respect to a parallel execution diagram.
l. Identify independent tasks in a program that may be parallelized.
m. Implement a parallel divide-and-conquer (and/or graph algorithm) and empirically measure its performance relative to its sequential analog.
n. Decompose a problem (e.g., counting the number of occurrences of some word in a document) via map and reduce operations.