Course Outline Report

MC New Course: CS 006 - Introduction to Parallel, GPU, and Cloud Computation

COLLEGE:
Merritt College

ORIGINATOR: Brown, Courtney

DIVISION/DEPARTMENT:
Merritt - Division II/M - Technology

STATE CONTROL NUMBER: CCC000614952

DATES:
BOARD OF TRUSTEES APPROVAL DATE:
STATE APPROVAL DATE:
CURRICULUM COMMITTEE APPROVAL DATE:
REQUISITE VALIDATION:

CURRENT EFFECTIVE DATE: 01/01/2021

1. REQUESTED CREDIT CLASSIFICATION:

D - Credit - Degree Applicable

N - Not Basic Skills

1 - Program Applicable

2. DEPT/COURSE NO:

CS 006

3. COURSE TITLE:

Introduction to Parallel, GPU, and Cloud Computation

4. COURSE:

Course
MC New Course

TOP NO. 0707.10 - Computer Programming*

5. UNITS:

Variable No

Units (Min) 4.000

Min Total

Hours

Lecture Hours (Min) 3.000

52.5

Lab/Studio/Activity Hours (Min) 3.000

52.5

6. SELECTED TOPIC:

NO. OF TIMES OFFERED AS SELECTED TOPIC:

AVERAGE ENROLLMENT:

7. JUSTIFICATION FOR COURSE:

This course is for students who want to use Cloud computing or High Performance Computing as is common in STEM, biotechnology, physics, chemistry, and social science research. The ability to simulate problems using High Performance Computing (HPC) is becoming a requirement for successful research at National Laboratories such as Lawrence Livermore, and Lawrence Berkeley NERSC. This course prepares students for entry level jobs supporting scientific computation. It is a requiremnt for national Cybersecurity Education standards compliance, specifically Center For Academic Excellence-(CAE2Y-CD) and The National IA Education & Training Program (NIETP) standards. Alignment with these national standards brings articulation to any four-year cybersecurity program nation-wide. It also aligns with California STEM Core network workforce development opportunities for students. This course meets the requirements for a C++ & Parallel Programming Certificate.

8. COURSE/CATALOG DESCRIPTION

Creation of programs that use multiple processors to solve a problem: Use of more than one thread, process, core, and/or machine in the same program. Coordinated computation of multi-tasking and multi-threading through management of many computing elements working on the same problem. Programming for Cloud computation, Cluster computation, Big Data, Machine Learning, and High Performance Computing (HPC), multi-core processors and Graphics Processing Units (GPU).

9. OTHER CATALOG INFORMATION

a. Modular: No

If yes, how many modules:

b. Open entry/open exit: No

c. Grading Policy: Both Letter Grade or Pass/No Pass

d. Eligible for credit by Exam: No

e. Repeatable according to state guidelines: No

f. Required for degree/certificate (specify):

g. Meets GE/Transfer requirements (specify): See GE tab

h. C-ID Number:

Expiration Date:

i. Are there prerequisites/corequisites/recommended preparation for this course? Yes

10. LIST STUDENT PERFORMANCE OBJECTIVES (EXIT SKILLS):

If an Objective cannot be deleted, make sure a Content-Review found in the Content Validation Page is not using that objective.

Objectives

1. **Build parallel programs using class inheritance and polymorphism**
2. **Describe processes, threads, and the resources used by a running program including CPU cycles, file descriptors, input/output bandwidth, and storage**
3. **Write programs that use more than one thread, process or task concurrently to solve a problem, selecting data structures that maintain integrity under concurrent access.**
4. **Write programs that utilize parallel hardware using standards such as CUDA, OpenCL, and MPI.**

11. COURSE CONTENT:

LECTURE CONTENT:

A. Extension of single threaded programming concepts (12.5%)
B. Variable types, their underlying hardware counterparts, and cross-machine issues (12.5%)
C. Program execution and individual instances of computation (12.5%)
D. Control and coordination tools for programs, processes and threads (12.5%)
E. Introduction of classes and object as abstractions for control and coordination (12.5%)
F. Portability and standards for platform independent use of system services (12.5%)
G. Parallel computation hardware: local, remote, cloud (12.5%)
H. Polymorphism, inheritance and, class design for reusable software (12.5%)

LAB CONTENT:

Installation & Configuration of High Performance Computing Hardware Software Stack (12.5%)

Create portfolio of Serial and Parallel Programs (60%)

Analyze performance differences and measure changes in speed (15%)

Implement Programs using parallel communication and synchronization abstraction (12.5%)

12. METHODS OF INSTRUCTION (List methods used to present course content):

- Lecture
- Lab
- Observation and Demonstration
- Discussion
- Projects
- Directed Study
- Service Learning
- Threaded Discussions

Other Methods:

13. ASSIGNMENTS

Out-of-class Assignments (List all assignments, including library assignments. Requires two (2) hours of independent work outside of class for each unit/weekly lecture hour. Outside assignments are not required for lab-only courses, although they can be given.)

Override Outside Class Hours: No

Outside-of-Class Hours (Min) 6.000

Outside-of-Class Hours (Max) 0.000

Override Outside-of-Class Hours (Min) 0.000

Override Outside-of-Class Hours (Max) 0.000

Out of class Assignment
Design and implement programs to execute as single threaded, This portfolio program requires the student to design their own well-understood serial algorithm which they will convert to a parallel algorithm across several frameworks. Serial program execution is the default model of programming. Through this project the student learns about design of parallelize-able algorithms using a simple goal, like finding integers that are prime in a file containing several million integers.

Use classes, objects and methods to design, implement, test, and debug a parallel program. This portfolio program This program is a refactoring (reorganizing) if necessary of their serial programs. Object orientation and inheritance are used to have the same method used to invoke equivalent operations across different parallel architectures and libraries. The student learns how to enable side-by-side

comparison of the different parallel implementations using the serial implementation as a test oracle to detect implementation errors in the parallel algorithms. Identical invocation methods allow performance differences to be clearly visible and teaches the student how to hide complexity.

Use various Application Programming Interfaces (APIs) and to implement parallel programs on specific parallel devices The student refactors their code to execute on a massively parallel device like Graphics Processing Unit (GPU) or compute co-processor. Their new code must distribute the values to a co-processor, instruct the co-processor to perform the computation, and synchronize data transfer operations between the host and the co-processor. The student learns how to write a program that will detect and configure a parallel co-processor such as a GPU, configure it, and use it to perform a parallel version of a serial algorithm. You will develop the skills of communication and data transfer across the system bus and the best practice of checking co-processor results against a serial CPU implementation of the same algorithm.

14. STUDENT ASSESSMENT: (Grades are based on):

- ESSAY (Includes "blue book" exams and any written assignment of sufficient length and complexity to require students to select and organize ideas, to explain and support the ideas, and to demonstrate critical thinking skills.)
- COMPUTATION SKILLS
- NON-COMPUTATIONAL PROBLEM SOLVING (Critical thinking should be demonstrated by solving unfamiliar problems via various strategies.)
- SKILL DEMONSTRATION
- MULTIPLE CHOICE

OTHER (Describe):

15. TEXTS, READINGS, AND MATERIALS

A. Textbooks:
YesNo37
Lin, Calvin Snyder, Lawrence. *Principles of Parallel Programming*. 1 edition Addison-Wesley, 2009.
Barlas, Gerassimos. *Multicore and GPU Programming: an Integrated Approach*. 2 edition Morgan Kaufmann, 2022.
Sehr, Viktor and Andrist, Bjorn . *C++ High Performance*. 2 edition Packt Publishing Ltd., 2020.

Freely available software and Software Development Kits; CUDA SDK, OpenCL, Parallel Tools Eclipse, Gnu Compilers, Gnu Debuggers, Web based wiki and resources, Message Passing Interface (MPI), and more.

*Date is required: Transfer institutions require current publication date(s) within 5 years of outline addition/update.

B. Additional Resources:
Library/LRC Materials and Services:

The instructor, in consultation with a librarian, has reviewed the materials and services of the College

Library/LRC in the subject areas related to the proposed new/updated course
Print Materials were reviewed? Yes

Non-Print Materials were reviewed? No

Online Materials were reviewed? Yes

Services were reviewed? Yes

Specific materials and/or services needed have been identified and discussed. Librarian comments:

C. Readings listed in A and B above are: (See definition of college level):
YesNo39
Primarily college level

16. DESIGNATE OCCUPATIONAL CODE:

B - Advanced Occupational

17. LEVEL BELOW TRANSFER:

Y - Not applicable

18. CALIFORNIA CLASSIFICATION CODE:

Y - Credit Course

19. NON CREDIT COURSE CATEGORY:

Y - Not Applicable, Credit course

20. FUNDING AGENCY CATEGORY:

A - This course was primarily developed using Economic Development funds

REQUISITES AND ADVISORIES

PREREQUISITE:
CS 002 Control Structures and Objects and 003 Data Structures, Software Architectures, and Algorithms and CS 005 Digital Architectures for Computation

STUDENT LEARNING OUTCOMES

1. **Identify and select among methods of hiding complexity when implementing parallel programs.**

   Projects are assigned and student work is assessed base on implementation using object oriented methodology.

2. **Plan for the division and distribution of of work among processes, threads, and compute resources.**

   Student must construct narrative for quiz/exam essay questions, program comments, online forums and chats.

3. **Identify and resolve conflicts in concurrently executing programs**

   Assignment that requires selecting data structures that maintain integrity under concurrent access, and protection of critical sections of code.

4. **Select strategies for allocating compute resources to running program**

   Assignment requires estimation of CPU cycles, file descriptors, input/output bandwidth, and storage used by a parallel program.