COMPUTER ENGINEERING (LM55)

(Lecce - Università degli Studi)

Teaching PARALLEL ALGORITHMS

GenCod A003130

Owner professor Massimo CAFARO

Teaching in italian PARALLEL ALGORITHMS

Teaching PARALLEL ALGORITHMS

SSD code ING-INF/05

Reference course COMPUTER ENGINEERING

Course type Laurea Magistrale

Credits 9.0

Teaching hours Front activity hours: 81.0

For enrolled in 2019/2020

Taught in 2020/2021

Course year 2

Language ENGLISH

Curriculum PERCORSO COMUNE

Location Lecce

Semester First Semester

Exam type Oral

Assessment Final grade

Course timetable https://easyroom.unisalento.it/Orario

BRIEF COURSE DESCRIPTION

The course provides a modern introduction to design, analysis and implementation of sequential and parallel algorithms. In particular, the course is based on a pragmatic approach to parallel programming of message-passing algorithms through the C language and the MPI library.

REQUIREMENTS

Calculus I and II, Probability Theory. Programming skills and working knowledge of the C programming language.

COURSE AIMS

Knowledge and understanding. Students must have a solid background with a broad spectrum of basic knowledge of sequential and parallel algorithms:

· the students must have the basic cognitive tools to think analytically, creatively, critically and in an inquiring way, and have the abstraction and problem-solving skills needed to cope with complex systems;
· they must have a solid knowledge of the design and implementation of sequential and parallel efficient algorithms;
· they must have the tools for analysing the resources used by algorithms;
· they must have a catalogue of the most well-known and efficient sequential and parallel algorithms for basic computational problems.

Applying knowledge and understanding. After the course the student should be able to:

· Describe and use the main design techniques for sequential algorithms;
· Design, prove the correctness and analyze the computational complexity of sequential algorithms;
· Understand the differences among several algorithms solving the same problem and recognize which one is better under different conditions;
· Describe and use basic sequential algorithms;
· Describe and use basic data structures; know about the existence of advanced data structures;
· Understand the difference between sequential and parallel algorithms;
· Design, implement and analyze message-passing based parallel algorithms in C using the MPI library;
· Describe and use basic parallel algorithms.

Making judgements. Students are guided to learn critically everything that is explained to them in class, to compare different approaches to solving algorithmic problems, and to identify and propose, in an autonomous way, the most efficient solution they find.

Communication. It is essential that students are able to communicate with a varied and composite audience, not culturally homogeneous, in a clear, logical and effective way, using the methodological tools acquired and their scientific knowledge and, in particular, the specialty vocabulary. The course promotes the development of the following skills of the student: ability to expose in precise and formal terms an abstract model of concrete problems, identifying the salient features of them and discarding the nonessential ones; ability to describe and analyze an efficient solution to the problem in question.

Learning skills. Students must acquire the critical ability to relate, with originality and autonomy, to the typical problems of data mining and, in general, cultural issues related to other similar areas. They should be able to develop and apply independently the knowledge and methods learnt with a view to possible continuation of studies at higher (doctoral) level or in the broader perspective of cultural and professional self-improvement of lifelong learning. Therefore, students should be able to switch to exhibition forms other than the source texts in order to memorize, summarize for themselves and for others, and disseminate scientific knowledge.

TEACHING METHODOLOGY	The course aims to enable students to abstract formal algorithmic models and problems from concrete computational problems, and to design efficient algorithmic solutions for them. This will be done using the following teaching method. Every computational problem will be introduced, motivating it with concrete examples. The presentation of each topic will be divided into four parts: 1. Description of the actual computational problem. 2. Modelling the real problem by means of an abstract problem. 3. Resolution of the abstract problem through an algorithm obtained through the application of the general techniques of design of algorithms introduced in the course. 4. Analysis of the resources used by the algorithm. The course consists of frontal lessons, and classroom exercises. There will be theoretical lessons aimed at learning the basic techniques for the project and analysis of algorithms, and a part of lessons devoted to exercises in which we will illustrate, with plenty of examples, how the theoretical knowledge acquired can be used in order to solve algorithmic problems of practical interest and implement parallel algorithms in C language through the MPI library.

ASSESSMENT TYPE

The exam consists of a written test and a prototype implementation of a parallel software. The written test (3 hours, 21 points out of 30) covers theoretical topics related to the design and analysis of sequential and parallel algorithms in order to verify the student's knowledge and understanding of the materials. In order to pass the written test, students must obtain at least 14 points out of 21. The parallel software prototype (9 points out of 30) is meant to verify the practice of parallel programming and the ability of the student to implement theoretical parallel algorithms or to parallelize a sequential algorithm. In order to pass the parallel programming challenge, students must obtain at least 4 points out of 9. Both the written test and the prototype implementation of parallel software are mandatory. Students must pass both the written test and the parallel programming challenge.

WARNING: THE FOLLOWING ARE DETAILED INSTRUCTIONS RELATED TO ONLINE WRITTEN TESTS USING MICROSOFT TEAMS

Written test of
Parallel Algorithms

Microsoft Teams is used to verify the student's identity and to supervise the test; the teacher will select each student asking him to show his identification document. The exam takes place within a team that students access through registration by the teacher or through a link that will be communicated via email. Students are asked to enter the team at least 15 minutes prior to the scheduled start time in order to perform recognition. Each student will participate in the meeting through a device equipped with a microphone and a webcam that must remain on for the duration of the test, framing the student and the paper on which he/she writes. The teacher will check the regularity of the students' work, immediately cancelling the test in case of irregularities of any kind. For the distribution of the exam outline to the students, the teacher shares a pdf file in the TEAMS chat. The track is divided into 3 exercises for Parallel Algorithms, and the teacher releases to the students each of the exercises that compose it at fixed time intervals. The test is then carried out as follows:

1. The teacher releases the first exercise at the beginning of the roll call;
a. students print out the text of the exercise and begin drafting the paper on a sheet of paper clearly visible from the video capture device;
b. the teacher provides his/her estimate x of sufficient time to answer;
c. delivery of the paper related to the first exercise after x amount of time has elapsed;
d. Teacher waits 2 minutes for delivery;

2. Teacher delivers the second exercise;
a. students print out the text of the exercise and begin writing the paper on a sheet of paper clearly visible from the video capture device;
b. teacher gives his estimate y of sufficient time to answer;
c. delivery of the paper related to the second exercise after the time y has elapsed;
d. The teacher waits 2 minutes for the delivery;

3. Teacher releases the third exercise;
a. students print out the text of the exercise and begin writing the paper on a piece of paper that is clearly visible from the video capture device;
b. teacher gives his estimate z of sufficient time to answer;
c. delivery of the paper related to the second exercise after the time z has elapsed;
d. The teacher waits 2 minutes for the delivery;

4. The teacher informs the participating students that the roll call is closed.

For the delivery of the papers related to the individual exercises, the student is requested to take a photo of the relative sheets, including an image of the university booklet or, alternatively, an identification document. The photos should be placed in a directory called "Exercise_a_b_c_d" in which:
- a indicates the progressive number of the exercise (1, 2 or 3);
- b is PA (for Parallel Algorithms)
- c indicates the student's first name;
- d indicates the student's last name.

The directory must then be compressed, e.g. in zip format, and the resulting file must be delivered via team chat.
It is not possible to turn in an exercise before the allotted time and move on to the next exercise. The student who intends to abandon the test can give notice to the teacher at any time, and leave the team immediately afterwards. It is not possible to leave for any reason during the test. The student is required to ensure that he/she has everything necessary for the test (microphone, webcam, printer, camera, pen, adequate number of sheets of paper).

OTHER USEFUL INFORMATION

Office Hours
By appointment; contact the instructor by email or at the end of class meetings.

FULL SYLLABUS

Parallel Algorithms
Introduction. Parallel Architectures. Parallel algorithm design. Message-Passing Programming. Sieve of Erathostenes. Floyd all-pairs shortest path algorithm. Performance analysis. Matrix-vector multiplication. Document classification. Matrix multiplication. Linear Systems. Finite Difference Methods. Sorting.

Parallel Programming

Message-Passing programming using MPI.

Sequential Algorithms

Introduction. Order of growth. Analysis of algorithms. Decrease and conquer. Divide and conquer. Recurrences. Randomized algorithms. Transform and conquer. Dynamic programming. Greedy algorithms. Single Source Shortest Paths. Dijkstra algorithm. Breadth-First Search. Bellman-Ford Algorithm. Complexity and computability. NP-Completeness. The transition from sequential to parallel computing. Parallel complexity.

REFERENCE TEXT BOOKS

Introduction to Algorithms. Third edition. Cormen, Leiserson, Rivest, Stein. The MIT Press
Parallel Programming in C with MPI and OpenMP (International Edition). Michael J. Quinn. McGraw-Hill