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 Ore-Attivita-frontale: 81.0

For enrolled in 2017/2018

Taught in 2018/2019

Course year 2

Language INGLESE

Curriculum PERCORSO COMUNE

Location Lecce

Semester Primo-Semestre

Exam type Orale

Assessment Voto-Finale

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 using slides made available to students via the Moodle platform, 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 of an exercise type in which you 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	Oral exam. Optionally, a student may be assigned a small project. During the exam the student is asked to illustrate theoretical topics in order to verify his/her knowledge and understanding of the selected topics. The student may also be asked to design a very simple algorithm in order to assess his/her ability to identify and use the relevant design techniques; alternatively, the student may be asked to analyze the complexity of a small code fragment.
OTHER USEFUL INFORMATION	**Office Hours** By appointment; contact the instructor by email or at the end of class meetings.
FULL SYLLABUS	Sequential A*lgorithms* Introduction. Order of growth. Analysis of algorithms. Decrease and conquer. Divide and conquer. Recurrences. Randomized algorithms. Transform and conquer. Dynamic programming. Greedy algorithms. Complexity and computability. NP-Completeness. *Parallel Algorithms* Introduction. The transition from sequential to parallel computing. Parallel complexity. 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.
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 (2004) Michael J. Quinn McGraw-Hill