Universitat Autònoma de Barcelona

**2021/2022**

# Performance Engineering

Code: 104341
ECTS Credits: 6

| Degree | Type | Year | Semester |
|---|---|---|---|
| 2503758 Data Engineering | OB | 2 | 1 |

The proposed teaching and assessment methodology that appear in the guide may be subject to changes as a result of the restrictions to face-to-face class attendance imposed by the health authorities.

## Contact

Name: Juan Carlos Moure Lopez
Email: JuanCarlos.Moure@uab.cat

## Use of Languages

Principal working language: spanish (spa)
Some groups entirely in English: No
Some groups entirely in Catalan: No
Some groups entirely in Spanish: No

## Teachers

Juan Carlos Moure Lopez

## Prerequisites

It is highly recommended to have studied and passed the subjects of Fundamentals of Computer Science, Fundamentals of Programming and Advanced Programming

## Objectives and Contextualisation

1. Evaluate and analyze the performance of the execution of a program.
2. Understand the conversion from the source language (Python / C) to the assembler, identifying the transformations in the code.
3. Determine the data dependencies in the execution loops of the programs and use strategies to increase the parallelism at the level of instructions and data.
4. Understand the functioning of the memory hierarchy and use strategies to increase the locality of data accesses.

## Competences

- Design efficient algorithmic solutions to computational problems, implement them in the form of robust software developments which are structured and easy to maintain, and verify their validity.
- Make a critical evaluation of work carried out.
- Plan and manage the available time and resources.
- Search, select and manage information and knowledge responsibly.
- Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.

## Learning Outcomes

1. Analyse applications' performance, detect bottlenecks and apply possible optimisations.
2. Apply the basic principles of computer structure and programming.
3. Implement medium-difficulty algorithmic problems in a programming language.
4. Make a critical evaluation of work carried out.
5. Plan and manage the available time and resources.
6. Search, select and manage information and knowledge responsibly.
7. Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.

## Content

1. Performance Evaluation: tools to monitor and instrument the execution of a program.
2. Performance Analysis: computational complexity, factors determined by the processor architecture, and performance metrics.
3. Parallelism in the execution of Instructions: Pipeline and Vectorization.
4. Analysis of data dependencies in the execution loops.
5. Memory Hierarchy and Locality of data accesses: effect on performance.
6. Algorithmic optimization strategies: increase the parallelism and the locality of data accesses.

## Methodology

Theory classes: the knowledge of the subject will be explained. The basic concepts will be described and illustrated with practical examples. The most important learning problems will be highlighted and will show how to complete and deepen the contents. Practical cases will be discussed and the teacher will detect the most common comprehension and reasoning problems and solve them for all students.

Classes of Problems in Laboratory: cooperative activities of resolution of problems and of simple practical cases will be made, that will serve as support to the theory. After a previous individual work, the students will make a group sharing and solve their doubts. The teacher will detect the most common comprehension and reasoning problems and solve them in groups or for all students. The student's analysis and synthesis capacity, critical reasoning and problem-solving ability are promoted. During the session, the students should inform the teacher about their progress and the problems that may be encountered, and they will deliver a final document with the results and reflections on their activities.

Optimization Problems: throughout the course, exercises will be proposed to transform small pieces of code so that they run in less time. Students should apply the knowledge, methodologies and strategies learned and present the result by arguing the reasons for performance improvements. Students are expected to be proactive and dynamic, and to autonomously search for the necessary knowledge.

Annotation: Within the schedule set by the centre or degree programme, 15 minutes of one class will be reserved for students to evaluate their lecturers and their courses or modules through questionnaires.

## Activities

| Title | Hours | ECTS | Learning Outcomes |
|---|---|---|---|
| Type: Directed | | | |
| Resolution of practical problems in the laboratory | 26 | 1.04 | 1, 2, 4, 3, 5 |
| Theory classes | 26 | 1.04 | 6, 7 |
| Type: Supervised | | | |
| Performance Optimization of code snippets | 26 | 1.04 | 1, 2, 4, 6, 3, 5 |

| Type: Autonomous | | | |
|---|---|---|---|
| Autonomous study | 30 | 1.2 | 6, 5 |
| Previous preparation and documentation of work in the laboratory | 40 | 1.6 | 1, 4, 6, 5, 7 |

## Assessment

The dates for continuous evaluation and submission of works will be published on the Moodle platform of the UAB (Virtual Campus) and may be subject to programming changes for reasons of adaptation to possible incidents. Always be informed on the Moodle platform about these changes, as it is understood that it is the usual platform for exchanging information between teacher and students.

Evaluation Activities
Activity A: solving practical problems in the laboratory; it is done as a group; it represents 50% of the final grade; it is necessary to obtain a minimum score of 5 points out of 10; and it cannot be recovered.
Activity B: delivery of a report of activity A; it is done individually; assumes 15% of the final grade; it is not necessary to obtain a minimum grade; and it cannot be recovered.
Activity C: attendance and active participation in class; it is done individually; assumes 5% of the final grade; it is not necessary to obtain a minimum grade; and it cannot be recovered.
Activity D: written examination of theory and problems; it is done individually and without notes; assumes 30% of the final grade; it is necessary to obtain a minimum score of 5 points out of 10; and it can be recovered.

Apart from the evaluation activities described above, during the course it will be possible to propose other voluntary activities that can contribute to the final qualification of the course

Program of Evaluation Activities
Activities A, B and C are carried out throughout the course. Activity D is carried out at the end of the course, on the dates specified by the coordination.

Recovery Process
The student can present himself to the recovery whenever he/she has submitted to the activities A, B and D. According to the coordination of the Degree and the direction of the School of Engineering, only activity D can be recovered, it represents 30% of the final grade.

Review procedure
All evaluation activities can be reviewed in a place, date and time that will be published beforehand. Claims may be made onthe activity grade, which will be evaluated by the faculty responsible for the subject. If the student does not show up for this revision, this activity will not be reviewed later.

Qualifications
An honor grade (MH) will be awarded to students who reach a final grade equal to or greater than 9.00, up to a total of 5% of the students enrolled, prioritizing the part of the grade that corresponds to individual assessments (activities B, C and D).
If the student does not reach the minimum grade in any of the evaluation activities, the final numerical grade will be the minimum between 4.5 points and the weighted sum of the grades of all the activities.
A student will be considered non-assessable (NA) if he / she has not submitted to any of activities A, B and D.

Irregularities by students, copy and plagiarism
Without prejudice to other disciplinary measures deemed appropriate, will be scored with a zero the irregularities committed by a student that may lead to a variation of the rating of an act of evaluation. Therefore, copying, plagiarism, cheating, letting copy, etc. in any of the evaluation activities will involve suspending it with a zero. The evaluation activities qualified in this way and by this procedure will not be recoverable. If it is necessary to pass any of these assessment activities to pass the subject, then this subject will be suspended directly, with no opportunity to recover it in the same course. In the case of not passing the subject due to the fact that the student has committed irregularities in an evaluation act, the numerical note of the record will be the lower value between 3.0 and the weighted average of the marks (and therefore the approved by compensation will not be possible).

Evaluation of repeating students
There is no special treatment for repeating students.

## Assessment Activities

| Title | Weighting | Hours | ECTS | Learning Outcomes |
|---|---|---|---|---|
| Group work for solving practical problems in the laboratory + Answers + Reports | 50% | 0 | 0 | 1, 2, 6, 3, 5, 7 |
| Individual report on the resolution of practical problems in the laboratory | 15% | 0 | 0 | 1, 4, 6, 7 |
| Individual written examination of theory and problems | 30% | 2 | 0.08 | 1, 4, 7 |
| Participation in class/lab | 5% | 0 | 0 | 1, 4, 6, 5, 7 |

## Bibliography

Computer Architecture: A Quantitative Approach. 6th Edition
(https://cataleg.uab.cat/iii/encore/record/C__Rb2034006)
John Hennessy and David Patterson, Morgan Kaufmann (Elsevier) 2018 (Cap. 1-3)

Computer Systems: A Programmer's Perspective. 3rd Edition (North-American)
(https://csuc-uab.primo.exlibrisgroup.com/permalink/34CSUC_UAB/avjcib/alma991006392489706709)
Randall E. Bryant and David R. O'Hallaron, Addison Wesley, 2015 (Cap. 5 i 6)

Computer Organization and Design RISC-V Edition: The Hardware/Software Interface.
David Patterson and John Hennessy, Morgan Kauffman (Elsevier), 2017 (Cap. 4 i 5)

High Performance Python: practical performant programming for human.
Micha Gorelick and Ian Ozsvald, O'Reilly, 2014

Cython: a guide for Python programmers.
Kurt W. Smith, O'Reilly, 2015

Python High Performance programming. Boost the performance of your Python programs using advanced techniques
Gabriele Lanaro, Packt Publishing, 2013

## Software

None