Universitat Autònoma de Barcelona

Advanced High Performance Computing

Code: 44730
ECTS Credits: 6

2025/2026

Degree	Type	Year
Research and Innovation in Computer based Science and Engineering	OP	1

Contact

Name: Juan Carlos Moure Lopez

Email: juancarlos.moure@uab.cat

Teachers

Nehir Sonmez Tekin

Teaching groups languages

You can view this information at the end of this document.

Prerequisites

Programming skills (preferible in C language)

Objectives and Contextualisation

The fundamental objective of the subject is that students acquire the capabilities of analysis, use and evaluation of computers, for the development of advanced applications.

The basic concepts that will be described in the theoretical sessions are: parallel processing in computers, hierarchical memories, performance evaluation, and the parallel programming languages and models of these systems. The most specific objectives are the following:

1. Understand the keys to obtain good performance from multi-core, multi-processor computers, and from computer accelerators (GPUs)

2. Identify the opportunities for parallelization in an algorithm or application, at different levels (instructions, iterations of a loop, calls to functions ...), and express it formally

3. Design the data structures of an application and the algorithms that access these data to favor the good performance of the memory hierarchy

4. Analyze, design and implement parallel algorithms with abstract computation patterns (reduction, transformation ...) under programming paradigms based on shared variables and synchronization; and with current parallel languages.

5. Verify the functionality and evaluate the performance of parallel applications, identifying performance bottlenecks

6. From performance analysis, select the right computer to an application and/or perform code optimizations that further improve its performance

Learning Outcomes

1. CA04 (Competence) Apply high-performance computing techniques to complex problems in various fields of science or engineering.
2. CA05 (Competence) Generate questions and hypotheses in the face of new research challenges and demonstrate originality in the way they approach and solve problems using high-performance computational techniques.
3. KA06 (Knowledge) Describe the most suitable algorithmic patterns for different high-performance computing environments.
4. KA07 (Knowledge) Describe the most common performance issues that computer applications may exhibit depending on the system architecture in which they are executed.
5. KA08 (Knowledge) Identify the most appropriate tech stacks on the basis of cost-performance criteria to develop high-performance applications that address problems in the fields of science and engineering.
6. SA08 (Skill) Identify the performance problems of complex applications by selecting and applying the appropriate profiling techniques and tools.
7. SA09 (Skill) Optimise applications adapted to different architectures: multicore systems, distributed systems, graphics accelerators (GPUs) and hybrid systems.
8. SA10 (Skill) Design innovative applications in the fields of science and engineering by applying theoretical models and using high-performance computing techniques and tools.

Content

1. Instruction-Level Parallelism: Data dependencies between instructions in a loop and dynamic reordering of its execution

2. Data-Level Parallelism: Analysis of the data parallelism of an algorithm and SIMD execution (vectorization)

3. Task Parallelism: multi-thread and multi-core execution; Shared Memory hierarchy and data coherence; synchronization between threads

4. Parallel Algorithms: Model of Parallelism with shared variables and synchronization. Parallel computing patterns: map, reduction, master/worker, divide & conquer and pipeline

5. Performance evaluation of Applications: Analysis of complexity, parallelism and locality: Performance metrics (elapsed time, IPC, bandwdth, arithmetic intensity); Total work and critical path of parallel execution

Activities and Methodology

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
Attendance at theory class and theoretical exercises	26	1.04	
Practices in LAB	16	0.64	

Type: Supervised		
Research work	15	0.6
Type: Autonomous		
Autonomous study, peport writting and and presentation development	41	1.64
Study and Problem Solving	50	2

Theory classes: the knowledge of the subject will be explained, illustrated with practical examples. The most important learning problems will be highlighted and will show how to complete and deepen the contents. Practical problems will be discussed and solved, and the teacher will detect the most common problems of understanding and reasoning and solve them for all students. The student's analysis and synthesis capacity, critical reasoning and problem-solving ability are promoted.

Practical case of autonomous study: throughout the course, students must design, plan, carry out, present and defend orally a practical work, indicating the objectives, the development, the results obtained with understandable diagrams, figures and graphs, the most noteworthy and, finally, the most relevant conclusions. A proactive and dynamic attitude is expected, the autonomous search for the knowledge that is necessary, and ambition with the objectives of the work.

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.

Assessment

Continous Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes
Individual Exam	20	1.5	0.06	KA06, KA07, KA08
LAB exercises: Programming and Performance Evaluation	50	0	0	CA04, CA05, KA07, SA08, SA09, SA10
Written report and Oral defense of the autonomous work	30	0.5	0.02	CA05, KA06, KA07, KA08

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: written examination of theory and problems; done individually without notes; 20% of the final grade; no minimum grade; can be recovered; at the end of the course
Activity B: lab exercises; done as a group; 50% of the final grade; minimum grade of 5 out of 10; no recovereable; during all the course
Activity C: written report and oral defense of autonomous work; done individually; 30% of the final grade; no minimum grade; can be recovered; at the end of the course

Recovery Process
The student can opt to recover activities A or C if he/she has done activities A, B and C

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, unauthorized use of IA (i.e., Copilot, ChatGPT or equivalents), 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 weightedaverage of the marks (and therefore the approved by compensation will not be possible).

Bibliography

Parallel Programming: Techniques and Applications using Networked Workstations and Parallel Computers. Barry Wilkinson. Prentice Hall, 1999.
Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Ian Foster. Addison Wesley, 1995.
Introduction to Parallel Computing. A. Grama et alter. Addison Wesley, Second Edition, 2003.
Computer Architecture: A Quantitative Approach. 5th Edition (https://bibcercador.uab.cat/permalink/34CSUC_UAB/15r2rl8/cdi_askewsholts_vlebooks_9780123838735) John Hennessy, David Patterson, Morgan Kaufmann (Elsevier) 2018 (Cap. 4 i 5)
Structured Parallel Programming: Patterns for efficient computation M. McCool, J. Reinders, A. Robison, Elsevier, 2012
Parallel Programming for Multicore and Cluster Systems T. Rauber, G. Rünger, Springer (Elsevier), 2010
Programming Massively Parallel Processors: A Hands-on Approach D. Kirk, & W.M. Hwu, Morgan Kaufmann (Elsevier), 2010

Software

C language. OpenMP and OpenACC extensions.

Groups and Languages

Please note that this information is provisional until 30 November 2025. You can check it through this link. To consult the language you will need to enter the CODE of the subject.

Name	Group	Language	Semester	Turn
(PLABm) Practical laboratories (master)	1	English	first semester	afternoon
(TEm) Theory (master)	1	English	first semester	afternoon