Course guide
2703127 - HPC - High Performance Computing

Last modified: 30/01/2026

Unit in charge:	Barcelona School of Informatics
Teaching unit:	701 - DAC - Department of Computer Architecture.
Degree:	BACHELOR'S DEGREE IN BIOINFORMATICS (Syllabus 2024). (Compulsory subject).

Academic year: 2025 **ECTS Credits:** 6.0 **Languages:** English

LECTURER

Coordinating lecturer:	JOSE RAMON HERRERO ZARAGOZA
Others:	Segon quadrimestre: CHRISTIAN GUZMAN RUIZ - 11, 12 JOSE RAMON HERRERO ZARAGOZA - 11, 12 SANDRA ADRIANA MENDEZ VALERIO - 11, 12 MIQUEL ANGEL SENAR ROSELL - 11, 12

PRIOR SKILLS

To follow this course satisfactorily, participants must have a solid foundation in sequential imperative programming and fundamental data structures. Knowledge of the C programming language with a specific focus on pointers and dynamic memory management is a must. Familiarity with the Linux/Unix command line and basic build tools is needed to navigate the lab environment effectively. Additionally, students should be comfortable with the manipulation of mathematical expressions, as this skill is necessary to understand the theoretical concepts and performance models introduced throughout the sessions. However, no prior knowledge of parallel programming (MPI, OpenMP, or OpenACC) is required.

LEARNING RESULTS

Knowledges:
K3. Identify the mathematical foundations, computational theories, algorithmic schemes and principles of information organisation relevant to modelling biological systems and efficiently solving bioinformatics problems through the design of computational tools.
K4. Integrate the concepts offered by the most widely used programming languages in life sciences to model and optimise data structures and build efficient algorithms, relating them to each other and to their application cases.

Skills:
S7. Implement programming methods and data analysis based on the development of working hypotheses within the area of study.
S8. Make and defend reasoned decisions when solving problems in biology and, in appropriate fields, the health sciences, computer science and experimental sciences.

Competences:
C6. Identify and overcome gaps in one’s knowledge by thinking critically and choosing the best approach to extending one’s knowledge.

TEACHING METHODOLOGY

Theoretical classes amount to two hours per week. They introduce all the knowledge, techniques, concepts needed to solve problems and lab assignments. Some problems will be solved during these theoretical sessions. The student is expected to work on the rest as personal work using a collection of problems. Additionally, two hours of laboratory sessions are also held weekly; active participation and performance during the laboratory sessions will be valued (work during the session, advancing as far as possible in order to achieve the objectives of each session).
Students must work autonomously prior to the sessions in order to prepare for the contact sessions and use time well. Afterwards, they have to work autonomously in order to practice and consolidate concepts and abilities developed throughout the course.
For the practical assignments and the problems in this course, mainly the C programming language and the MPI, OpenMP and OpenACC parallel programming models will be used. The practical laboratory sessions will be done in a parallel machine running the Linux operating system. The access to such cluster will be done using the student's laptop. Thus, the students must make sure that they can connect to a Linux cluster from their laptop.

LEARNING OBJECTIVES OF THE SUBJECT

1.Ability to formulate simple performance models given a parallelization strategy for an application, that allows an estimation of the influence of major architectural aspects: number of processing elements, data access cost and cost of interaction between processing elements, among others.
2.Ability to identify the different types of parallelism that can be exploited in a computer architecture (ILP, TLP, and DLP within a processor, multiprocessor and multicomputer) and describe its principles of operation.
3.Ability to compile and execute a parallel program, using the essential command-line tools to measure the execution time.
4.Ability to choose the most appropriate decomposition strategy to express parallelism in an application (tasks, data).
5.Ability to apply the basic techniques to synchronize parallel execution, avoiding race conditions and deadlock and enabling the overlap between computation and interaction, among others.
6.Ability to program in MPI the parallel version of a sequential application.
7.Ability to program in OpenMP the parallel version of a sequential application.
8.Ability to program in OpenACC the parallel version of a sequential application.
9.Ability to measure, using instrumentation, visualization and analysis tools, the performance achieved with the implementation of a parallel application and to detect factors that limit this performance: granularity of tasks, equitable load and interaction between tasks, among others.

STUDY LOAD

Type	Hours	Percentage
Hours small group	30,0	20.00
Hours large group	30,0	20.00
Self study	90,0	60.00

Total learning time: 150 h

CONTENTS

Introduction to high performance computing and parallel computing

Description:
Introduction to high performance computing, parallel architectures and parallel computing

Understanding Paralallelism

Description:
Theoretical concepts. Performance metrics.

Distributed memory programming with MPI

Description:
Distributed memory architectures: programming with Message Passing Interface (MPI)

Performance modeling

Description:
Ability to formulate simple performance models given a parallelization strategy for an application, that allows an estimation of the influence of major architectural aspects: number of processing elements, data access cost and cost of interaction between processing elements, among others.

Shared memory programming with OpenMP

Description:
Shared memory architectures: programming with OpenMP

Cluster Computing and Batch Queue Systems

Description:
Cluster Computing and Batch Queue Systems: SLURM

Python for HPC

Description:
Python for High Performance Computing

GPU computing and OpenACC

Description:
Computation with Graphics Processing Units (GPUs) using OpenACC

ACTIVITIES

Theoretical expository lectures

Description:
To take ownership of their learning process. This involves attending classes, labs, and tutorials punctually and participating actively in discussions. Students are responsible for managing their time effectively to complete all required assignments, readings, and projects by the specified deadlines. They must also adhere to the university's academic honesty policies, seek clarification when material is unclear, and respect the learning environment by engaging with peers and the teacher in a professional manner.

Specific objectives:
1, 2, 4, 5, 6, 7, 8

Full-or-part-time: 72h
Theory classes: 27h
Self study: 45h

Practical Hands-on sessions

Description:
The student must develop practical assignments using a parallel cluster. In such environment, the student's duty includes active execution, problem-solving, and resource management. They must:

- Prepare in advance by reviewing the theoretical concepts and any pre-lab readings or initial setup instructions before the lab session.

- Execute the tasks independently, carefully documenting the procedures, inputs, and outputs, including any errors encountered and how they were resolved.

- Develop practical debugging skills and utilize available technical resources (documentation, specific software tools) to solve technical challenges before seeking instructor help.

- Submit working, well-documented code/reports that follow the specified formatting, version control, and naming conventions.

Specific objectives:
3, 4, 5, 6, 7, 8, 9

Full-or-part-time: 73h
Practical classes: 28h
Self study: 45h

Theory Exam 1st part (TE1)

Description:
Theory Exam 1st part (TE1)

Specific objectives:
1, 2, 4, 5, 6, 9

Full-or-part-time: 1h 30m
Guided activities: 1h 30m

Laboratory Exam 1st part (LE1)

Description:
Laboratory Exam 1st part (LE1)

Specific objectives:
3, 4, 5, 6, 9

Full-or-part-time: 1h
Guided activities: 1h

Theory Exam 2nd part (TE2)

Description:
Theory Exam 2nd part (TE2)

Specific objectives:
2, 4, 5, 7, 8

Full-or-part-time: 1h 30m
Guided activities: 1h 30m

Laboratory Exam 2nd part (LE2)

Description:
Laboratory Exam 2nd part (LE2)

Specific objectives:
3, 4, 7, 8, 9

Full-or-part-time: 1h
Guided activities: 1h

GRADING SYSTEM

The course grade will be computed as the weighted average of 6 grades, (3 grades for each of the parts of the course):

CG = 0.05 * LS1 + 0.15 * LE1 + 0.3 * TE1 + 0.05 * LS2 + 0.15 * LE2 + 0.3 * TE2

Where:

LS_i relates to the submission of the deliverables of the practical assignments for the i-th part.

LE_i is the laboratory exam for the i-th part.

TE_i is the theory exam for the i-th part.

IMPORTANT: Completion and presentation of all laboratory follow-up reports (LS_i) is a necessary condition to pass the subject. Only reports with a minimum of content are considered a report prepared and presented. Empty reports or with only the questions, for example, are not considered completed or submitted.

Students will pass the course if they have completed all exams and obtained a course grade (CG) equal to or greater than 5 over 10.

Students who fail the course with a final average grade above 3 over 10, will be allowed to take a recovery exam for the theoretical part. The recovery exam will consist of two parts: there will be a retake exam for each of the 2 parts (TRE1 and/or TRE2) for students who failed that specific part. In the re-evaluation exam only the theoretical part of the course can be retaken. The grade TRE_i will then replace the grade obtained in the regular course for the corresponding theory exam TE_i.

BIBLIOGRAPHY

Basic:
- Grama, Ananth. Introduction to parallel computing. 2nd ed. Pearson Addison Wesley, 2003. ISBN 9780201648652.
- Hager, Georg; Wellein, Gerhard. Introduction to high performance computing for scientists and engineers. CRC Press, cop. 2011. ISBN 9781439811924.
- Gropp, William; Lusk, Ewing; Skjellum, Anthony. Using MPI : portable parallel programming with the Message-Passing-Interface. Third edition. MIT Press, [2014]. ISBN 9780262326605.
- Chapman, Barbara; Jost, Gabriele; Pas, Ruud van der. Using OpenMP : portable shared memory parallel programming. MIT Press, cop. 2008. ISBN 9780262533027.

Complementary:
- McCool, Michael; Robison, Arch D; Reinders, James. Structured parallel programming : patterns for efficient computation. 1st edition. Elsevier/Morgan Kaufmann, [2012]. ISBN 9786613689603.
- Gropp, William; Lusk, Ewing; Skjellum, Anthony. Using MPI : portable parallel programming with the Message-Passing-Interface. Third edition. MIT Press, [2014]. ISBN 9780262326605.
- Eijkhout, Victor. The Science of Computing (formerly: Introduction to High-Performance Scientific Computing). CC-BY 4.0 license. 2022.
- Eijkhout, Victor. Parallel Programming in MPI and OpenMP The Art of HPC, volume 2. CC-BY 4.0 license. 2022.

RESOURCES

Hyperlink:
- https://computing.llnl.gov/tutorials/mpi/- https://hpc-tutorials.llnl.gov/openmp/- https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial- https://slurm.schedmd.com/- https://www.mpi-forum.org/docs/- https://www.openacc.org/- https://www.openmp.org/resources/