Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

COURSE DESCRIPTION – ACADEMIC YEAR 2021/2022

Course title	**Introduction to Parallel Computing**
Course code	73049
Scientific sector	INF/01
Degree	Master in Computational Data Science (LM-18)
Semester	1
Year	2
Credits	6
Modular	No

Total lecturing hours	40
Total lab hours	20
Attendance	Attendance is not compulsory, but strongly recommended. Students who are unable to follow all lectures and labs are encouraged to interact with the lecturer.
Prerequisites	Good knowledge of the following subjects is expected: • Programming • Computer Systems • Algorithms and Data Structures
Course page	https://ole.unibz.it/

Specific educational objectives	The course belongs to the type "caratterizzanti – discipline informatiche" in the curricula "Data Analytics" and "Data Management". Students will acquire a deep knowledge of how to design faster and efficient applications by exploiting modern parallel architectures (e.g., GPUs). Under such a light, students will acquire professional skills and knowledge in parallel computing by understanding the most advanced techniques that researchers have developed in the last years.

Lecturer	Flavio Vella
Contact	Flavio.Vella@unibz.it, Piazza Domenicani, 3 – Office POS 3.13
Scientific sector of lecturer	INF/01
Teaching language	English
Office hours	Arrange beforehand by email
Lecturing Assistant (if any)	--
Contact LA	--
Office hours LA	--
List of topics	• Introduction to parallel and distributed systems • Shared memory model • Distributed memory model • Principle and design of parallel algorithms • Selection of parallel algorithms • Performance Analysis, optimization and tuning
Teaching format	• Frontal lectures • Lab supported by the lecturer and Teaching Assistant (if any) In the lectures, new concepts and techniques are introduced by

	presentation on the blackboard and multimedia material (slides and videos). In the labs, students will: ● use the tools that will be used during the course and project development; ● solve simple exercise and discuss the solution; ● start to prepare the final project.

Learning outcomes	Knowledge and understanding: • D1.1 - Knowledge of the key concepts and technologies of data science disciplines • D1.3 - Knowledge of principles, methods and techniques for processing data in order to make them usable for practical purposes, and understanding of the challenges in this field • D1.4 - Sound basic knowledge of storing, querying and managing large amounts of data and the associated languages, tools and systems • D1.5 - Knowledge of principles and models for the representation, management and processing of complex and heterogeneous data Applying knowledge and understanding: • D2.1 - Practical application and evaluation of tools and techniques in the field of data science • D2.2 - Ability to address and solve a problem using scientific methods Making judgments • D3.2 - Ability to autonomously select the documentation (in the form of books, web, magazines, etc.) needed to keep up to date in a given sector Communication skills • D4.1 - Ability to use English at an advanced level with particular reference to disciplinary terminology Learning skills • D5.1 - Ability to autonomously extend the knowledge acquired during the course of study • D5.2 - Ability to autonomously keep oneself up to date with the developments of the most important areas of data science • D5.3 - Ability to deal with problems in a systematic and creative way and to acquire problem solving techniques

Assessment	The assessment is based on a final project that will be assigned during the semester. It will consist of solving a particular problem by using parallel computing techniques. The project will be developed by a group of two students (at most). Specifically, the teams have to: ● release the code by providing the instruction for result reproducibility; ● write a short scientific document describing the solution, the methodology, the technology they adopted for solving the problem;

	• prepare a short oral presentation of the project. After the presentation will follow a Q&A session to assess the knowledge of the candidate and its contribution to the project. The assessment is based on the **individual** contribution of each team member.
Assessment language	English
Assessment Typology	Monocratic
Evaluation criteria and criteria for awarding marks	The final mark is composed by evaluating the project in terms of originality of the methods adopted/designed, results obtained and quality and clarity of the presentation which includes code, document and oral presentation. Specifically: • Code assessment (20% of the final mark). The software should follow the best practice for code writing. The experiments must be replicable. • Document assessment (40% of the final mark). Ability to introduce a problem. Ability to report the state of the art. Ability to describe the methodology adopted. Ability to comment and report the results obtained. The originality and the soundness of the solution will be also considered in the evaluation. • Quality and clarity of the oral presentation in a fixed time (40% of the final mark). Ability to answer to possible questions to the project and the related topics addressed during the course that can arise during the presentation.

Required readings	There is not a single book that cover all the topics that will be presented during the course. • Introduction to parallel computing 2nd edition (Grama, Karpis, Kumar, Gupta) • Computer Architecture: a quantitative approach. 6th ed (Hennessy, Patterson) • The Art of Multiprocessor Programming (Herlihy, Shavit) • Programming Massively Parallel Processors: A Hands-on Approach 3rd Edition (Kirk, Hwu) Subject Librarian: David Gebhardi, David.Gebhardi@unibz.it
Supplementary readings	• CUDA C++ Best Practices Guide and CUDA C++ Programming Guide; • C++ for OpenCL Programming Language; • CUDA by examples (Sanders and Kandrot). • Patterns for parallel programming (Timothy G. Mattson) • Parallel Computer Architecture. A Hardware / Software Approach (David Culler)
Software used	Programming languages: C/C++ or Python. Compilers GCC and NVCC Other software/frameworks: CUDAToolkit, OpenMPI or OpenCL. Github.