SYLLABUS

Parallel and Distributed Programming

University year 2025/2026

1. Information regarding the programme

1.1. Higher education institution	Babeş Bolyai University
1.2. Faculty	Faculty of Mathematics and Computer Science
1.3. Department	Department of Computer Science
1.4. Field of study	Computer Science
1.5. Study cycle	Bachelor
1.6. Study programme/Qualification	Computer Science
1.7. Form of education	Full time

2. Information regarding the discipline

2.1. Name of the discipline	**Parallel and Distributed Programming**					Discipline code	**MLE5077**
2.2. Course coordinator					Lect. PhD. Radu Lupșa		
2.3. Seminar coordinator					Lect. PhD. Radu Lupșa		
2.4. Year of study	3	2.5. Semester	5	2.6. Type of evaluation	E	2.7. Discipline regime	Compulsory

3. Total estimated time (hours/semester of didactic activities)

3.1. Hours per week	**5**	of which: 3.2 course	**2**	3.3 seminar/laboratory/project	**0/2/1**
3.4. Total hours in the curriculum	70	of which: 3.5 course	28	3.6 seminar/laboratory/project	**42**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					10
Additional documentation (in libraries, on electronic platforms, field documentation)					10
Preparation for seminars/labs, homework, papers, portfolios and essays					20
Tutorship					10
Evaluations					5
Other activities:					-
3.7. Total individual study hours	**55**				
3.8. Total hours per semester	**125**				
3.9. Number of ECTS credits	**5**				

4. Prerequisites (if necessary)

4.1. curriculum	Programming Fundamentals, Object Oriented Programming, Data Structures and Algorithms, Operating Systems
4.2. competencies	Programming abilities

5. Conditions (if necessary)

5.1. for the course	Lecture room with videoprojector
5.2. for the seminar /lab activities	Room with videoprojector; computers with IDEs for C++, Python, Java and C#

6.1. Specific competencies acquired [1]

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

Professional/essential competencies	• use of theoretical foundations of computer science as well as of formal models • use of software tools in an interdisciplinary context
Transversal competencies	• application of organized and efficient work rules, of responsible attitudes towards the didactic-scientific field, to bring creative value to own potential, with respect for professional ethics principles and norms • use of efficient methods and techniques to learn, inform, research and develop the abilities to bring value to knowledge, to adapt at the requirements of a dynamical society and to communicate efficiently in Romanian language and in an international language

6.2. Learning outcomes

Knowledge	The graduate has the necessary knowledge for using computers, developing software programs and applications, information processing. The graduate has knowledge related to programming, mathematics, engineering and technology and has the skills to use them to create complex information technology systems.
Skills	The graduate has the necessary skills for computer program design and software systems analysis. The graduate has the ability to apply general rules to specific problems and produce relevant solutions.
Responsibility and autonomy:	The graduate is able to identify complex problems and examine related issues to develop solving options and implement solutions. The graduate is able to combine diverse information to formulate solutions and generate ideas for developing new products and applications.

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	• Aquiring the main concepts of concurrent, parallel and distributed programming; • Basics of communication between processes and threads, on the same machine or on distinct machines; • Knowing basic techniques of parallel programming; • Knowing and using parallel application design patterns • Knowing and using the existing frameworks for developing parallel and distributed applications
7.2 Specific objective of the discipline	• Parallel architectures and parallel programming systems • Know how to use parallel programming techniques in problem solving • Know how to evaluate the performance increase obtained by parallelization • Ability to work independent or in a team in order to solve problems in a parallel and/or distributed context

8. Content

8.1 Course	Teaching methods	Remarks
C1. General introduction. Necessity to use parallelism. Concurrent vs parallel vs distributed computing	Exposure: description, explanation, examples, debate.	
C2. Parallel architectures: pipeline, vectorial machines, grid and cluster computing.	Exposure: description, explanation, examples, debate.	
C3. Threads. Race conditions, mutual exclusion, deadlocks. Synchronization primitives.	Exposure: description, explanation, examples, debate.	
C4. Producer-consumer parallelism. Low-level primitives (condition variables) and high-level mechanisms (futures, producer-consumer queues)	Exposure: description, explanation, examples, debate.	
C5-C6. Asynchronous programming. Futures with continuations. Coroutines.	Exposure: description, explanation, examples, debate.	
C7. Basic parallel algorithms.	Exposure: description, explanation, examples, debate.	
C8. Recursive decomposition and parallel explore algorithms.	Exposure: description, explanation, examples, debate.	
C9. Distributed programming using MPI	Exposure: description, explanation, examples, debate.	
C10. Distributed recursive decomposition and parallel explore.	Exposure: description, explanation, examples, debate.	
C11. Distributed protocols. Lamport clocks.	Exposure: description, explanation, examples, debate.	
C12. Distributed shared memory.	Exposure: description, explanation, examples, debate.	
C13. GPGPU programming. OpenCL.	Exposure: description, explanation, examples, debate.	
C14. Fault tolerance	Exposure: description, explanation, examples, debate.	

Bibliography

- http://www.cs.ubbcluj.ro/~rlupsa/edu/pdp/
- Ian Foster. Designing and Building Parallel Programs, Addison-Wesley 1995.
- Michael McCool, Arch Robinson, James Reinders, Structured Parallel Programming: Patterns for Efficient Computation," Morgan Kaufmann,, 2012 .
- Berna L. Massingill, Timothy G. Mattson, and Beverly A. Sanders,Addison A Pattern Language for Parallel Programming. Wesley Software Patterns Series, 2004.
- Grama, A. Gupta, G. Karypis, V. Kumar. Introduction to Parallel Computing, Addison Wesley, 2003.
- D. Grigoras. Calculul Paralel. De la sisteme la programarea aplicatiilor. Computer Libris Agora, 2000.
- V. Niculescu. Calcul Paralel. Proiectare si dezvoltare formala a programelor paralele. Presa Univ. Clujana, 2006.
- D.B. Skillicorn, D. Talia. Models and Languages for Parallel Computation. ACM Computer Surveys, 30(2) pg.123-136, June 1998.
- B. Wilkinson, M. Allen, Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice Hall, 2002
- E.F. Van de Velde. Concurrent Scientific Computing. Spring-Verlag, New-York Inc. 1994.
- Boian F.M. Ferdean C.M., Boian R.F., Dragos R.C. Programare concurenta pe platforme Unix, Windows, Java. Ed. Albastra, grupul Microinformatica, Cluj, 2002 .
- OpenMP Tutorials
- MPI Tutorials
- OpenCL Tutorials

8.2 Seminar / laboratory	Teaching methods	Remarks
L1. Introduction	Dialogue, debate, examples, guided discovery.	
L2-L3. Synchronization primitives.	Dialogue, debate, examples, guided discovery.	
L4. Producer-consumer parallelism.	Dialogue, debate, examples, guided discovery.	
L5-L6. Asynchronous programming	Dialogue, debate, examples,	

	guided discovery.	
L7. Basic parallel algorithms	Dialogue, debate, examples, guided discovery.	
L8. Recursive decomposition	Dialogue, debate, examples, guided discovery.	
L9. Parallel explore	Dialogue, debate, examples, guided discovery.	
L10. Basic distributed algorithms with MPI	Dialogue, debate, examples, guided discovery.	
L11. Recursive decomposition and parallel explore with MPI	Dialogue, debate, examples, guided discovery.	
L12. Distributed shared memory.	Dialogue, debate, examples, guided discovery.	
L13. OpenCL	Dialogue, debate, examples, guided discovery.	
L14. Finalizing lab activities	Dialogue, debate, examples, guided discovery.	

Bibliography

- Eckel, B., Thinking in Java, 4th Edition, New York: Prentice Hall, 2006.
- Larman, C.: Applying UML and Design Patterns: An Introduction to OO Analysis and Design, Berlin: Prentice Hall, 2004.
- Fowler, M., Patterns of Enterprise Application Architecture, Addison-Wesley, 2002.
- E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns – Elements of Reusable Object Oriented Software, Ed. Addison Wesley, 1994.
- Walls, Craig, Spring in Action, Third Edition, Ed. O'Reilley, 2011.
- Kent Beck, Test Driven Development: By Example, Ed. Addison-Wesley Professional, 2002.
- http://download.oracle.com/javase/tutorial/
- http://msdn.microsoft.com/en-us/library/aa288436%28v=vs.71%29.aspx
- http://www.cs.ubbcluj.ro/~rlupsa/edu/pdp/

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- The course follows ACM and IEEE recommendations for computer science studies
- The course is part of the curricula in all major universities, both local and abroad
- The software companies consider the course content important for acquiring advanced programming abilities.

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	Knowing basic concepts	Written exam	50%
	Applying theoretical knowledge in problem solving	Semester project	20%
10.5 Seminar/laboratory	Applying theoretical knowledge in problem solving	Evaluation of lab assignments	30%
10.6 Minimum standard of performance			
• At least 12 out of 14 attendances at the labs			

- At least grade 5 (out of 10) for the written exam
- At least grade 5 (out of 10) for the final average.

11. Labels ODD (Sustainable Development Goals)[2]

Not applicable.

Date:
...

Signature of course coordinator

.....

Signature of seminar coordinator

.....

Date of approval:
...

Signature of the head of department

Assoc.prof.phd. Adrian STERCA

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable*.".