SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babeș 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**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Parallel and Distributed Programming** **Programare Paralelă și Distribuită**				
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 Type of discipline	**Compulsory**
2.8 Code of the discipline		MLE5077					

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

3.1 Hours per week	6	Of which: 3.2 course	2	3.3 seminar/laboratory	1/2/1
3.4 Total hours in the curriculum	84	Of which: 3.5 course	28	3.6 seminar/laboratory	56
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					15
Additional documentation (in libraries, on electronic platforms, field documentation)					10
Preparation for seminars/labs, homework, papers, portfolios and essays					21
Tutorship					10
Evaluations					10
Other activities:					-

3.7 Total individual study hours	66
3.8 Total hours per semester	150
3.9 Number of ECTS credits	6

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	• Laboratory with workstations

6. Specific competencies acquired

Professional competencies	The student should prove that he has a good level of knowledge understanding of the field and he is capable of showing the knowledge and can use them in problem solving using parallel and distributing programming.
Transversal competencies	CT1 Application of organized and efficient work rules, of responsible attitudes towards the didactic and scientific domain, for the creative exploitation of their own potential according to the principles and rules of professional ethics CT3 Use of effective methods and techniques of learning, information, research and development of the capacity to exploit knowledge, to adapt to the requirements of a dynamic society and communication in Romanian language and in a foreign language.

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

7.1 General objective of the discipline	• Aquire 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
1. General introduction: • necessity to use parallelism • concurrent vs. parallel vs. distributed computing • levels of parallelism	Exposition, concepts, examples, case study.	
2. Parallel architectures: • Pipeline	Exposition, concepts, examples, case study.	

• Vectorial machines • Grid and cluster computers • Supercomputers		
3. : • Processes vs threads • Managing the processes/threads	Exposition, concepts, examples, case study.	
4. Concurrency concepts: • race conditions, critical sections, mutual exclusion, deadlock • synchronizations: monitors, semaphores	Exposition, concepts, examples, case study.	
5. Models of parallelism: • Implicit vs. explicit parallelism • data parallelism • message-passing • shared memory	Exposition, concepts, examples, case study.	
6. Parallel programming in shared memory: C pthreads, C++ threads, Java threads, OpenMP	Exposition, concepts, examples, case study.	
7. Performance evaluation for parallel programs: PRAM (Parallel Random Access Machine). Efficiency, cost, scalability.	Exposition, concepts, examples, case study.	
8. Parallel programming patterns: • master-slaves • task farm / work pool • divide et impera • pipeline	Exposition, concepts, examples, case study.	
9. Message passing parallel programs. MPI	Exposition, concepts, examples, case study.	
10. Phases in a parallel program: PCAM (Partition, Communication, Aggregation, Mapping): • task decomposition • domain (geometrical) decomposition • granularity • degree of parallelism • task dependency	Exposition, concepts, examples, case study.	
11. Parallel programs construction techniques: • divide et impera • binary tree • recursive double-back	Exposition, concepts, examples, case study.	
12. Data parallel programming	Exposition, concepts, examples, case study.	
13. GPGPU (General Processing on the Graphical Processing Unit): OpenCL, CUDA	Exposition, concepts, examples, case study.	
14. Distributed file systems	Exposition, concepts, examples, case study.	

Bibliography

http://www.cs.ubbcluj.ro/~rlupsa/edu/pdp/

1. Ian Foster. Designing and Building Parallel Programs, Addison-Wesley 1995.

2. Michael McCool, Arch Robinson, James Reinders, Structured Parallel Programming: Patterns for Efficient

Computation," Morgan Kaufmann,, 2012 .
3. Berna L. Massingill, Timothy G. Mattson, and Beverly A. Sanders,Addison A Pattern Language for Parallel
Programming. Wesley Software Patterns Series, 2004.
4. Grama, A. Gupta, G. Karypis, V. Kumar. Introduction to Parallel Computing, Addison Wesley, 2003.
5. D. Grigoras. Calculul Paralel. De la sisteme la programarea aplicatiilor. Computer Libris Agora, 2000.
6. V. Niculescu. Calcul Paralel. Proiectare si dezvoltare formala a programelor paralele. Presa Univ. Clujana, 2006.
7. D.B. Skillicorn, D. Talia. Models and Languages for Parallel Computation. ACM Computer Surveys, 30(2) pg.123-136,
June 1998.
8. B. Wilkinson, M. Allen, Parallel Programming Techniques and Applications Using Networked Workstations and Parallel
Computers, Prentice Hall, 2002
9. E.F. Van de Velde. Concurrent Scientific Computing. Spring-Verlag, New-York Inc. 1994.
10. 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 .
11. ***, OpenMP Tutorials
12. ***, MPI Tutorials
13. ***, CUDA Tutorials

8.2 Seminar / laboratory	Teaching methods	Remarks
S1 Threads vs processes		
S2 Concurrent programming		
S3 OpenMP		
S4 Parallel design patterns		
S5-S6 MPI		
S7 CUDA/OpenCL		
L1 Threads vs processes		
L2-L5 Concurrent programming C++, Java, C#		
L6-L7 OpenMP		
L8-L10 MPI		
L11-L14 CUDA/OpenCL		

Bibliography

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

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

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	Knowing basic concepts	written exam	40%
	Applying theoretical knowledge in problem solving	project	30%
10.5 Seminar/lab activities	Applying theoretical knowledge in problem solving	evaluation of lab assignments	30%
10.6 Minimum performance standards			
➢ At least 4.5 out of 10 for the written exam ➢ At least 4.5 out of 10 the average			

Date Signature of course coordinator Signature of seminar coordinator

....

Date of approval Signature of the head of department

....….....