Parallel and functional programming techniques - course description

General information

Course name	Parallel and functional programming techniques
Course ID	11.3-WE-INFD-PaFPT-Er
Faculty	Faculty of Computer Science, Electrical Engineering and Automatics
Field of study	Computer Science
Education profile	academic
Level of studies	Second-cycle Erasmus programme
Beginning semester	winter term 2021/2022

Course information

Semester	3
ECTS credits to win	5
Course type	optional
Teaching language	english
Author of syllabus	• dr hab. inż. Marek Sawerwain, prof. UZ

Classes forms

The class form	Hours per semester (full-time)	Hours per week (full-time)	Hours per semester (part-time)	Hours per week (part-time)	Form of assignment
Lecture	15	1	-	-	Credit with grade
Laboratory	15	1	-	-	Credit with grade
Project	15	1	-	-	Credit with grade

Aim of the course

- Familiarize students with basic information about parallel and functional programming techniques.
- To shape understanding and awareness of the role of parallel programming techniques as well as highlight the increasing role of functional programming.
- To give basic skills in creating parallel programs for multi-core systems based on traditional processors (CPU) as well as graphics multi-core processors of general use.
- Learning of the basic skills in the functional programming paradigm, and in particular: the role of functions and recursion, programming without side effect and the acquisition of skill to use the method of the lazy computations.

Prerequisites

Methods of Programming, Algorithms and Data Structures, Theoretical Foundations of Computer Science, Logic for Computer Scientists

Scope

Theory of computation models:models of parallel computations and complexity classes.

Programmer tools: available tools for parallel programming for CUDA and OpenCL technologies.

Basic operations: Parallel primitive operations.

Data Dependency: dependency and division of data, models of execution of parallels environments for CPU and GPU.

Programming paradigm: Functional paradigm and basic constructions in selected functional languages OCaml, F#, Scala.

Basic data types: Data types in functional programming, exceptions and objects.

High-class function: firstclass and higherorder functions, functional model of computations (in a form of simplified operational description).

Type system and imperative control flow instructions: type systems, and lazycomputations, imperative features in functional programming languages.

Teaching methods

Lecture: conventional lecture
Laboratory: laboratory exercises, group work
Project: project method, discussions and presentations

Learning outcomes and methods of theirs verification

Outcome description	Outcome symbols	Methods of verification	The class form
Knows and understands the basics of functional paradigm and the role of imperative constructions in functional languages.		• a test with score scale	• Lecture • Laboratory
Knows model of parallel programming used in modern hardware -- software computation systems.		• a test with score scale	• Lecture • Laboratory

Outcome description	Outcome symbols	Methods of verification	The class form
Can use available functional programming features to reduce implementation time of selected problems in computer science.		• a test with score scale	• Lecture
Can use existing libraries which supports parallel programming techniques for CPU and GPU.		• a project • sprawozdanie z projektu	• Laboratory • Project
Is aware of dynamic development of parallel programming techniques and rising role of functional programming.		• a test with score scale	• Lecture

Assignment conditions

Lecture - obtaining a positive grade in written exam.
Laboratory - the main condition to get a pass are sufficient marks for all exercises and tests conducted during the semester.
Project - a condition of pass is to obtain positive marks from all project tasks and preparation written report of project.
Calculation of the final grade: = lecture 40% + laboratory 30% + project 30%.

Recommended reading

1. Mattson G.T., He Y., Koniges E.A.:The OpenMP Common Core: Making OpenMP Simple Again, MIT Press, 2019.
2. Han J., Sharma B.: Learn CUDA Programming: A beginner's guide to GPU programming and parallel computing with CUDA 10.x and C/C++, Packt Publishing, 2019.
3. Rauber T., Rünger G.: Parallel Programming for Multicore and Cluster Systems, Springer Berlin Heidelberg, 2013
4. Herlihy M., Shavit N.: The Art of Multiprocessor Programming, Morgan Kaufmann, 2012
5. Gaster B., Howes L., Kaeli D. R., Mistry P., Schaa D.: Heterogeneous Computing with OpenCL, Morgan Kaufmann, 2011.
6. Pacheco P.: An Introduction to Parallel Programming, Morgan Kaufmann, 2011.
7. Smith C.: Programming F#.: O'Reilly Media, Inc., Sebastopol, USA, 2010.
8. Sanders J., Kandrot E.: CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley Professional, 2010.
9. Pickering R.: Foundations of F#, Apress, USA, 2007.

Further reading

1. Syme D., Granicz A., Cisternino A.: Expert F# , Apress, USA, 2015.
2. Wen-mei W. Hwu, eds: GPU Computing Gems, Emerald Edition and Jade Edition, Morgan Kaufmann, 2011.
3. Farber R.: CUDA Application Design and Development, Morgan Kaufmann, 2011.
4. Harrop J.: F# for Scientists, John Wiley & Sons, Inc., Hoboken, New Jersey, USA, 2008.
5. Thomspon S.: Haskell - The Craft of Functional Programming, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1999.

Notes

-- no comments --

Modified by dr hab. inż. Marek Sawerwain, prof. UZ (last modification: 15-07-2021 23:47)