Parallel and functional programming techniques - opis przedmiotu

Informacje ogólne

Nazwa przedmiotu	Parallel and functional programming techniques
Kod przedmiotu	11.3-WE-INFD-PaFPT-Er
Wydział	Wydział Informatyki, Elektrotechniki i Automatyki
Kierunek	Informatyka
Profil	ogólnoakademicki
Rodzaj studiów	Program Erasmus drugiego stopnia
Semestr rozpoczęcia	semestr zimowy 2020/2021

Informacje o przedmiocie

Semestr	3
Liczba punktów ECTS do zdobycia	5
Typ przedmiotu	obieralny
Język nauczania	angielski
Sylabus opracował	• dr hab. inż. Marek Sawerwain, prof. UZ

Formy zajęć

Forma zajęć	Liczba godzin w semestrze (stacjonarne)	Liczba godzin w tygodniu (stacjonarne)	Liczba godzin w semestrze (niestacjonarne)	Liczba godzin w tygodniu (niestacjonarne)	Forma zaliczenia
Wykład	15	1	-	-	Zaliczenie na ocenę
Laboratorium	15	1	-	-	Zaliczenie na ocenę
Projekt	15	1	-	-	Zaliczenie na ocenę

Cel przedmiotu

- 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.

Wymagania wstępne

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

Zakres tematyczny

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.

Metody kształcenia

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

Efekty uczenia się i metody weryfikacji osiągania efektów uczenia się

Opis efektu	Symbole efektów	Metody weryfikacji	Forma zajęć

Opis efektu	Symbole efektów	Metody weryfikacji	Forma zajęć
Knows and understands the basics of functional paradigm and the role of imperative constructions in functional languages.		• sprawdzian z progami punktowymi	• Wykład • Laboratorium
Knows model of parallel programming used in modern hardware -- software computation systems.		• sprawdzian z progami punktowymi	• Wykład • Laboratorium
Can use available functional programming features to reduce implementation time of selected problems in computer science.		• sprawdzian z progami punktowymi	• Wykład
Can use existing libraries which supports parallel programming techniques for CPU and GPU.		• projekt • sprawozdanie z projektu	• Laboratorium • Projekt
Is aware of dynamic development of parallel programming techniques and rising role of functional programming.		• sprawdzian z progami punktowymi	• Wykład

Warunki zaliczenia

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%.

Literatura podstawowa

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.

Literatura uzupełniająca

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.

Uwagi

Zmodyfikowane przez dr hab. inż. Marek Sawerwain, prof. UZ (ostatnia modyfikacja: 26-04-2020 23:01)