AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Code:	UBPJO-229	Module name:	Introduction to CUDA and OpenCL		
Academic year:	2017/2018	Semester:	Fall	ECTS credits:	5
Programme:	Physics and Applied Computer Science				
Course homepage:	http://home.agh.edu.pl/~szumlak	Lecture language:	English		
Responsible teacher:	dr hab. inż, prof. AGH Szumlak Tomasz (szumlak@agh.edu.pl)				
Academic teachers:	dr hab. inż, prof. AGH Szumlak Tomasz (szumlak@agh.edu.pl)				

Module summary

This lecutre aims at providing a basic knowledge and skills on parallel programming using GPU. We introduce two frameworks NVIDIA CUDA SDK and AMD APP-ICD SDK.

Description of learning outcomes for module

MLO code	Student after module completion has the knowledge/ knows how to/is able to	Method of learning outcomes verification (form of completion)
Social competence		
M_K001	A student can present his/her results and discuss them	Activity during classes
Skills		
M_U001	A student can write complete programs, using extended C/C++ language. Can use a professional framework provided by NVIDIA. Can proficiently use IDE environment, nvidia compiler and debugger.	Project, Test
M_U002	A student can work as a part of a team and can interact properly with his/her team-mates.	Project
Knowledge		
M_W001	A student gains knowledge on advanced topics related to the massively parallel programming using GPU.	Participation in a discussion, Report, Activity during classes

FLO matrix in relation to forms of classes

MLO code	Student after module completion has the knowledge/ knows how to/is able to	Form of classes										
		Lectures	Auditorium classes	Laboratory classes	Project classes	Conversation seminar	Seminar classes	Practical classes	Fieldwork classes	Workshops	Others	E-learning
Social competence												
M_K001	A student can present his/her results and discuss them	-	-	-	+	-	-	-	-	-	-	-
Skills												
M_U001	A student can write complete programs, using extended C/C++ language. Can use a professional framework provided by NVIDIA. Can proficiently use IDE environment, nvidia compiler and debugger.	-	-	-	+	-	-	-	-	-	-	-
M_U002	A student can work as a part of a team and can interact properly with his/her team-mates.	-	-	-	+	-	-	-	-	-	-	-
Knowledge												
M_W001	A student gains knowledge on advanced topics related to the massively parallel programming using GPU.	+	-	+	+	-	-	-	-	-	-	-

Module content

Lectures

Introduction (2 h)

A generic introduction to multicore/many-core hardware and programming techniques. Students will be given a short overview of the current trends in both hardware available on the market (CPUs and GPUs) as well as software solutions provided for developers. Basic ideas of heterogeneous parallel computing.

Introduction to CUDA and OpenCL - how to approach (3 h)

During this lecture we introduce, at a high level for the time being, how to use the tools for developers and be able to write and run programs using CUDA and OpenCL. For this lecture we focus on NVIDIA CUDA SDK and AMD APP SDK frameworks. The first one, as the name suggests is dedicated for NVIDIA GPUs and has been introduced and being maintained by NVIDIA company. For the OpenCL world situation is a bit more complex and there is more than one way of using OpenCL (Khronos does not provide any ready-to-use software just specs). I decided to introduce the AMD IDE but beside that one there is a nice programming environment provided by Intel: Intel® SDK for OpenCL™. You can also choose this one if you prefer.

Programming and execution models - CUDA (4 h)

In general one can say, that CUDA is a parallel computing platform and a programming model. Also, it provides an extension to the standard C language which allows, in turn, to write and run massively parallel algorithms using CPU/GPU. Thanks to CUDA it is very easy to code your programs – it is almost like using C language. With CUDA extensions you can provide software for a diverse set of systems like tablets/phones, desktops, workstations and HPC clusters and last but not least embedded devices. Programming model can be viewed as an abstraction of computer architectures – it is a bridge between your application and its concrete implementation on available hardware. By learning the programming model we will be able to execute parallel code (so called kernels) on GPU, organise grids of threads and manage your devices.
On other hand, in order to understand how to optimise your software you need to gain knowledge on how respective instructions are actually executed on the available devices. This is a domain of the execution model, which exposes an high level (abstract) view of the GPU massively parallel architecture and threat concurrency.

GPU Memory - its hierarchy and management with CUDA (4 h)
Optimisation of the kernel performance is not only related with the threads management. It also has to do with a device memory. During these lectures you will learn how to manage the memory. We use so called global memory to create better applications by exploring its access patterns and maximising its throughput. We introduce the notion of unified memory, which is a great help for a developer. Also, the shared memory will be introduced and discussed in detail. We explain how to create and use cache to make your code even better.

GPU accelerated libraries (2 h)
At the end we discuss selected parallel libraries that can be used to speed up your applications. They can easily be employed as ‘atoms’ to build more and more complex programs. By analysing their behaviour we can learn new levels of parallelism. Here, we mainly focus on very generic features and look at linear algebra and random number generation. These libraries should be treated as state-of-the-art products that are highly optimised by large number of experts working with massively parallel devices.

Laboratory classes

Introduction to NVIDIA CUDA SDK (4 h)
Using our dedicated server we introduce NVIDIA CUDA SDK – from starting it through writing code, compiling it and executing. We also use profiling tools to analyse the execution time and performance.

Introduction to AMD SDK - OpenCL (4 h)
Similarly, we will have a look at IDE for developing parallel software using OpenCL programming model.

Code samples - a good way forward in self-teaching (4 h)
Both NVIDIA and AMD provied a large number of code samples. These are provided by proffessionals and are an excellent source for improving your programming skills. We are going to have a deep look into it.

Creating grids and optimising their performance (4 h)
The first step into creating efficient and robust programs is to understand algorithms. By exposing the type of parallelism one can create a thread layout that is the best for a given problem. We try to understand this and check how different thread grids can impact performance of your programs.

Memory management - how to gain in execution time (4 h)
Threads and their layouts are not the whole story! Memory is also important. Here, we try to understand the CUDA memory model and manage the memory of a device.

Using shared memory (4 h)
Communication between threads can be realised using the shared memory – which is one of the most important components of the GPU. We learn how it is related to streaming multiprocessors and how it could be exploited in your programs.

Project classes

To be defined!
Concrete topics very much depend on students! If you are interested in GPU you could get a more ambitious project. The group will be split into groups (2 up to 4 students). Each group will be asked to select a team leader and work together to finish the project. The topics will be hand out after we finish lectures – you are going to have plenty of time to comfortably complete it.

To be defined!
Concrete topics very much depend on students! If you are interested in GPU you could get a more ambitious project. The group will be split into groups (2 up to 4 students). Each group will be asked to select a team leader and work together to finish the project. The topics will be hand out after we finish lectures – you are going to have plenty of time to comfortably complete it.

Method of calculating the final grade
The final grade will depend on your performance during the computer laboratories and the final project: final_grade=0.5 • lab_grade + 0.5 • project_grade. NOTE! You need to have both labs and project passed to get an overall passing grade!!!

Prerequisites and additional requirements
Since this is the intro lecture there are minimal requirements regarding basic skills with using linux OS and programming in C language.

Recommended literature and teaching resources
Depertamental Library (Physics and Applied Computer Science) is quite well stocked with books pertaining to parallel programming. We have the following, that can be used to study the problem:
1. “CUDA for Engineers: An Introduction to High-Performance Parallel Computing”
Duane Storti, Mete Yurtoglu
Addison Wesley, ISBN-13: 978-0134177410
2. “CUDA Programming: A Developer’s Guide to Parallel Computing with GPUs (Applications of GPU Computing Series)”
Shane Cook
Morgan Kaufmann, ISBN-13: 978-0124159334
3. “Programming Massively Parallel Processors: A Hands-on Approach”
David B. Kirk, Wen-mei W. Hwu
Morgan Kaufmann; 2 edition (14 Dec. 2012), ISBN-13: 978-0124159921

Scientific publications of module course instructors related to the topic of the module
1. The LHCb Collaboration, “Measurement of the track reconstruction efficiency at LHCb”, JINST 10 (2015) P02007
2. The LHCb VELO Group, “Performance of the LHCb Vertex Locator”, JINST 9 (2014) P09007
3. The LHCb VELO Group, “Radiation damage in the LHCb Vertex Locator”, JINST 8 (2013) P08002

Additional information

The labs will be quite agressively paced – please attend them! This classes are compulsory and you are allowed to skip only one lab during the whole semester. Also, there will be very hard to catch up if you miss a class.

Student workload (ECTS credits balance)

Student activity form	Student workload
Participation in laboratory classes	22 h
Participation in project classes	8 h
Completion of a project	35 h
Preparation for classes	20 h
Realization of independently performed tasks	25 h
Examination or Final test	2 h
Contact hours	2 h
Participation in lectures	15 h
Summary student workload	129 h
Module ECTS credits	5 ECTS