UNIVERSITY OF GOTHENBURG

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DIT430 High Performance Parallel Programming, 7.5 credits

Parallell programmering för hög prestanda, 7,5 högskolepoäng

Second Cycle

Confirmation

This course syllabus was confirmed by Department of Computer Science and Engineering on 2019-02-08 to be valid from 2019-09-02, autumn semester of 2019.

Field of education: Science 100%
Department: Department of Computer Science and Engineering

Position in the educational system

The course is offered within several programmes. It is also a single subject course at the University of Gothenburg.

The course can be part of the following programmes: 1) Computer Science, Master's Programme (N2COS), 2) Applied Data Science Master's Programme (N2ADS) and 3) Software Engineering and Management Master's Programme (N2SOF)

Main field of studies	*Specialization*
Computer Science	AXX, Second cycle, in-depth level of the course cannot be classified

Entry requirements

To be eligible for the course, students should have successfully completed courses corresponding to 90 credits within the subject of Computer Science, Mathematics, Software Engineering, or equivalent, including a 7.5 credits course in machine-oriented programming (e.g., DIT151 Machine Oriented Programming, or equivalent).

Applicants must prove knowledge of English: English 6/English B or the equivalent level of an internationally recognized test, for example TOEFL, IELTS.

Learning outcomes

On successful completion of the course the student will be able to:

Knowledge and understanding

- List the different types of parallel computer architectures, programming models and paradigms, as well as different schemes for synchronization and communication.
- List the typical steps to parallelize a sequential algorithm
- List different methods for analyses methodologies of parallel program systems

Competence and skills

- Apply performance analysis methodologies to determine the bottlenecks in the execution of a parallel program
- Predict the upper limit to the performance of a parallel program

Judgement and approach

- Given a particular software, specify what performance bottlenecks are limiting the efficiency of parallel code and select appropriate strategies to overcome these bottlenecks given a certain software.
- Design energy-aware parallelization strategies based on a specific algorithm structure and computing system organization
- Argue for which performance analysis methods that are important given a specific context.

Course content

This course looks at parallel programming models, efficient programming methodologies and performance tools with the objective of developing highly efficient parallel programs.

The course consists of a set of lectures and laboratory sessions. The lectures start with an overview of parallel computer architectures and parallel programming models and paradigms. An important part of the discussion are mechanisms for synchronization and data exchange. Next, performance analysis of parallel programs is covered. The course proceeds with a discussion of tools and techniques for developing parallel programs in shared address spaces. This section covers popular programming environments such as pthreads and OpenMP. Next the course discusses the development of parallel programs for distributed address space. The focus in this part is on the Message Passing Interface (MPI). Finally, we discuss programming approaches for executing applications on accelerators such as GPUs. This part introduces the CUDA

(Compute Unified Device Architecture) programming environment.

The lectures are complemented with a set of laboratory sessions in which participants explore the topics introduced in the lectures. During the lab sessions, participants parallelize sample programs over a variety of parallel architectures, and use performance analysis tools to detect and remove bottlenecks in the parallel implementations of the programs.

Sub-courses

1. **Written exam** *(Tentamen),* 4.5 credits
   Grading scale: Pass with Distinction (VG), Pass (G) and Fail (U)
2. **Laboratory** *(Laborationer),* 3 credits
   Grading scale: Pass with Distinction (VG), Pass (G) and Fail (U)

Form of teaching

The teaching consists of theory-oriented lectures and lab sessions in which the participants develop code for different types of parallel computer systems.

Language of instruction: English

Assessment

The course is examined by an individual written exam that is carried out in an examination hall and a laboratory report written in groups of two.

If a student, who has failed the same examined component twice, wishes to change examiner before the next examination, a written application shall be sent to the department responsible for the course and shall be granted unless there are special reasons to the contrary (Chapter 6, Section 22 of Higher Education Ordinance).

In cases where a course has been discontinued or has undergone major changes, the student shall normally be guaranteed at least three examination occasions (including the ordinary examination) during a period of at least one year from the last time the course was given.

Grades

The grading scale comprises: Pass with Distinction (VG), Pass (G) and Fail (U).
A Pass grade (G) for the entire course requires at least a Pass grade for all sub-courses.
A Pass with Distinction grade (VG) for the entire course requires a Pass with Distinction (VG) on both sub-courses.

Course evaluation

The course is evaluated through meeting after the course between teachers and student representatives. Further, an anonymous questionnaire is used to ensure written information. The outcome of the evaluations serves to improve the course by indicating which parts could be added, improved, changed or removed.

Additional information

The course is a joint course together with Chalmers.

Course literature to be announced the latest 8 weeks prior to the start of the course.

Knowledge of concurrent programming is recommended, e.g., from the course DIT391 Principles of Concurrent Programming, or equivalent.