FDA125

Advanced Parallel Programming:Models, Languages, Algorithms (CUGS)

Lectures:

32 hours.

Recommended for

Graduate (CUGS, ECSEL, …) students interested in the areas of parallel computer architecture, parallel programming, compiler construction, or algorithms and complexity.
Interested undergraduate students are also welcome.

The course was last given:

This is a new course. It complements the existing graduate course FDA101 on Parallel Programming (4p), which is aliased to the undergraduate course TDDB78.

Goals

The course emphasizes fundamental aspects of parallel programming such as parallel architectures and programming models, performance models, parallel complexity classes, parallel algorithmic paradigms, parallelization strategies, and the design and implementation of parallel programming languages. Practical exercises help to apply the theoretical concepts of the course to solve concrete problems in different parallel programming models.

Prerequisites

Data structures and algorithms (e.g. TDDB57) are absolutely required; knowledge in complexity theory and compiler construction is useful. Processprogramming (e.g. TDDB63/68/72) and Parallel Programming (e.g. TDDB78 or TANA77) are useful but not required. Most of the contents of FDA101, i.e. TDDB78, will be summarized during this course.
Programming in C is necessary for the practical exercises.

Organization

Lectures, student presentations of research papers, programming exercises.

Contents

I. General introduction: Parallel computer architectures (*).

II. Basic theory of parallel computation: PRAM model. Time, work, cost. NC. Speedup and Amdahl's Law (*), Self-simulation and Brent's Theorem, Scalability and Gustafssons Law (*). Fundamental PRAM algorithms (reduction, parallel prefix, list ranking). PRAM variants, simulation results and separation theorems.

III. PRAM emulations on distributed-memory architectures: Hashed address space, Pipelining and Multithreading. Ranade's Fluent Machine and its cost-effective massively parallel realization in hardware. Low-level synchronization mechanisms.

IV. Message passing models: Delay model, classical BSP model of Valiant, BSP-model of McColl, LogP, LogGP.

V. Distributed shared memory realizations in CC-NUMA architectures and software DSM emulations. Memory consistency models.

VI. Parallel programming languages and environments: Fork, BSPlib, MPI (*), OpenMP (*), HPF (*), Split-C, NESL, ZPL, Linda, …

VII. Parallel algorithmic paradigms and programming techniques, with example problems: Parallel loops, static and dynamic loop scheduling. Data parallelism (*). Parallel divide-and-conquer, recursive doubling. Synchronization mechanisms in asynchronous computations and parallel data structures. Parallel task queues. Synchronous and asynchronous pipelining. Domain decomposition and irregular parallelism.

VIII. Generic parallel programming with skeletons. Skeleton-based programming environments.

IX. Compiling for parallel computers: Dependence analysis, loop transformations, loop parallelization, idiom recognition. Code generation from dataparallel languages for message-passing architectures. Message agglomeration. Optimization of data layout. Static and dynamic remapping of arrays. Prefetching and fuzzy barriers. Dynamic parallelization. Instruction-level parallelism and software pipelining. Granularity. Clustering and scheduling algorithms for various models.

X. More parallel algorithms: Multiprefix computations. Searching, merging and sorting. Integer sorting in constant time. List ranking. Lowest common ancestors and tree contraction. Connected components of graphs.

XI. Other issues as time permits.

Literature

Keller, Kessler, Träff: Practical PRAM Programming. Wiley Interscience, 2000. ISBN 0-471-35351-5.

Further literature to be annonced later.

Teachers

Christoph Kessler, Welf Löwe.

Examiner

Christoph Kessler.

Schedule

Spring 2003.

Examination

MUN1 Oral examination (4p) (3p in connection with FDA101/TDDB78)
PRE1 Presentation with written summary (1p)
UPG1 Programming exercise (1p).

Credit

6 credits (5p in connection with FDA101/TDDB78).

Comments

There is a partial overlap in contents with FDA101 Parallel Programming / TDDB78 that corresponds to about 1 point. This means that we can give only 5 points (3p for MUN1, 1p for PRE1, 1p for UPG1) to those who got points for FDA101 / TDDB78 earlier. Accordingly, some of the

lectures are optional for those students; these are marked by an asterisk (*) in the Contents section above.

A main difference to FDA101 / TDDB78 / TANA77 is not only in depth but also in scope: While FDA101 emphasize scientific computing and numerical applications, this course focuses on theory and non-numerical algorithms.