Course Motivation and Organization

Course Motivation:

The Information Revolution, started more than 20 years ago, was enabled by the explosive growth of semiconductor integration (and internet), and is governed by Moore's law: microprocessor transistor density doubles every 19-24 months and cost effectiveness keeps pace, translating into similarly increasing performance.

In the 90s, parallel systems were eclipsed by ``killer-micro'': the ever-increasing speed of the *Single CPU* provided the path of least resistence. In mid 2000s and after, power complexity and the ever-growing memory wall have reversed the trend towards *multi/many-cores*.   Massively parallel hardware is now mainstream and lays the the path towards higher performance. We are now at an important juncture: unless commodity programming can harness and unleash hardware parallelism, The Information Revolution will soon come to an end!

Course Organization:

The course aims at studying the deep synergy between parallel harware and programming, and, as such, the course is organized on three tracks:

Hardware Track: In order to write software that utilizes the hardware well, we need first to understand the levels at which hardware exposes parallelism, the constraints at each level, the main tradeoffs and the potential bottlenecks. Therefore, we will study the design space of the critical components of parallel hardware: *processor, memory hierarchy and interconnect*.

For example, the processor exhibits three level of parallelism: at pipeline level, denoted Instruction Level Parallelism (ILP), at intra-core level, e.g., hardware multi threading, and at inter-core level, e.g., multi/many cores.   Interconnect topologies dictate the layout of the parallel system, e.g., cores and caches, and the routing strategies between cores or between cores and memory. Since a memory location may be present in multiple caches, coherence protocols are needed to ensure system's consistency.

Software Track reviews several programming models, ranging from high to lower level, that make application parallelism explicit, and studies optimizations that enable better utilization of the parallel hardware.   We start by reviewing the purely functional map-reduce
programming style that expresses a clean and clear semantics of parallelism rooted in the mathematical structure of (list) homomorphisms.   We examine how such a high-level program can be translated to a lower-level parallel API, such as OpenMP, and several

transformations that improve the locality of reference and/or the degree of parallelism.

Finally, we examine the software-hardware duality: various optimizations can be implemented
at both levels, but exhibiting a different set of weaknesses and benefits. One such example
is the exploitation of instruction-level parallelism. Another one is thread-level speculation, in which different iterations of a loop are executed out of order even if the loop is not (statically) proven parallel; but violation to the program semantics are tracked and fixed at runtime.

CUDA/Lab Track: teaches GPGPU hardware specifics, the CUDA programming model, and applies various optimizations learned in the software track.

The table below gives a ***tentative scheduling*** of lectures and labs
(they may change or be reordered):

Week	**Date**	**Contents**	Cosmin Oancea (C) Troels Henriksen (T)
36	01/09	List Homomorphisms: Map-Reduce Programming (SFT)	C
	03/09	Flattening Nested Parallelism to Bulk Operations (SFT)	C
	03/09	LAB: Introduction to CUDA API, Simple Map	C
37	08/09	Introduction: Hardware Trends, Vector Machines Processor Architecture: Mostly-Static Pipelines (HWD)	C
	10/09	Static Optimization of ILP: Hardware and Software	C
	10/09	LAB: Basic Blocks: Reduce & Scan in CUDA	C
38	15/09	Reasoning About & Optimizing Parallelism (SFT)	C
	17/09	Memory Hierarchy & Cache Coherence Protocols (HWD)	C
	17/09	LAB: Matrix Multiplication in CUDA	C
39	22/09	Interconnect Topologies (HWD)	C
	24/09	Data Dependence Analysis & Vectorization (SFT)	C
	24/09	LAB: CUDA Optimizations	C
40	29/09	Optimizing Locality of Reference (SFT)	C
	01/10	Dynamic Pipeline Design, Tomasulo Alg (HWD)	C
	01/10	LAB: to be announced	C
41	05/10	***Project is Published***	
	06/10	Matrix Multiplication in MPI (SFT)	C
	08/10	Software Thread-Level Speculation (SFT+HWD)	C
	08/10	Help With Project	C
43	20/10	Extra Topic	
	22/10	Extra Topic	
	22/10	LAB: help with project	

In this course, all lectures will be held in English / Alle forelæsninger i dette kursus holdes på engelsk.