The Nineteenth International Conference on

Parallel Architectures and Compilation Techniques (PACT 2010)

Vienna, Austria, September 11-15, 2010

WORKSHOP AND TUTORIAL PROGRAM

Workshops & Tutorials at PACT 2010

Workshops and tutorials take place on Saturday, September 11 and Sunday, September 12.

Program of Saturday, September 11

On Saturday we have a tutorial and two workshops in parallel and (optionally) a dinner in the evening:

Tutorial 1:

Programming Ct – Porting Applications to Future Multicore

Michael Klemm and *Peter Hinsbeeck (Intel)*

Saturday, September 11, Sitzungssaal, 8:30 am – 5:00 pm

Intel® Ct Technology supports a high-level, generalized and portable programming model for data-parallel programming. It simplifies the efficient parallelization of computations over large data sets. Programmers do not need to focus on the implementation details of their data-parallel program, but instead can express a program's algorithms in terms of operations on collections of data. Ct's deterministic semantics avoid race conditions and deadlocks by design, improving reliability and maintainability, and can be used for both rapid prototyping and production-stable codes.

Ct manages the complexity of mapping the high-level description of the program's operations onto an efficient implementation by employing JIT compilation techniques. Its internal JIT compiler dynamically optimizes a program to whatever hardware is used for execution, automatically emitting vectorized and multi-threaded code appropriate for that hardware's microarchitecture. With Ct's JIT compiler it becomes possible to efficiently execute the program on multiple computing platforms (e.g. Intel® SSE, Intel® AVX) without recompiling the application. Ct's JIT compiler is also the key to support upcoming execution environments without the need to recompile a program: updating the Ct library alone, without recompilation of the application, will suffice to enable future platforms.

In this tutorial, we introduce to the participants the programming model and the execution environment of Intel® Ct Technology. We provide an in-depth guide to the basic building blocks of the Ct language: scalar types, dense and sparse vector data types, vector operations, elemental functions, and control flow. We present how Ct fits into an application's control flow and can be used to express different levels of abstraction. Based on real-world scientific codes and other examples, we then show how to construct data-parallel algorithms from these basic building blocks. We demonstrate how to smoothly move an existing sequential code base to a parallel code base. In addition, we illustrate how to make use of external libraries such as the Intel® Math Kernel Library. We close the tutorial with a live demonstration of performance and scalability analysis as well as performance optimization of Ct applications.

Topics Covered:

The topics covered will include the following items:

- Introduction to Ct and its goals
- Getting started with Ct
- The Ct Language
  - Data types
  - Operators
  - Control flow constructs
- Example programs
- Ct Dynamic Engine and JIT compiler
- Porting applications to Ct
- Utilizing external libraries
- VTune performance analysis and performance tuning (demo)

Workshop 1:

GPUs and Scientific Applications (GPUScA)

Eduard Mehofer, Markus Schordan, Dan Quinlan and *Beniamino Di Martino*

Clubraum, Saturday, September 11

8:45 **9:00**	**Opening & Welcome**
9:00 **10:00**	**Keynote Address: *Vivek Sarkar*** Towards a Portable Execution Model for Extreme Scale Multicore Systems
10:30	**Applications with strong algorithmic aspects** Improving the GPU-based Collision Check Procedure for Distributed Crowd Simulations *Guillermo Vigueras, Juan M. Orduña, Miguel Lozano, Jose M. Cecilia, and Jose M. García* Fast GPU perspective grid construction and triangle tracing for Exhaustive Ray Tracing of Highly Coherent Rays *Lancelot Perrotte and Guillaume Saupin* Solving Planted Motif Problem on GPU *Naga Shailaja Dasari, Ranjan Desh, and Zubair M*
12:00	
13:30	**Applications with strong domain aspects** Scalability of Color-Based Segmentation of Football Players over GPUs *Miguel Angel Montañes, Enrique F. Torres, Jesus Martinez, and J. Elias Herrero* Fluid Simulation With CUDA Using the Lattice Boltzmann Method *Andreas Monitzer* A Framework for GPU Accelerated Deformable Object Modeling *Aria Shahingohar and Roy Eagleson*
15:00	
15:30	**Parallel programming technology** ViennaCL - A High Level Linear Algebra Library for GPUs and Multi-Core CPUs *Karl Rupp, Florian Rudolf, and Josef Weinbub* Dynamic Work Scheduling for GPU Systems *Miguel Angel Lastras-Montaño, Maged M. Michael, and J. Alan Bivens*
16:30	
16:30 **16:45**	**Closing Remarks**

Workshop 2:

Parallel Architectures and Bioinspired Algorithms (WPABA)

Jose L. Risco-Martín, Francisco Fernández and *Juan Lanchares*

Museumszimmer, Saturday, September 11

8:30 **8:40**	Welcome and introduction
8:40 **9:00**	A Parallel Memetic Algorithm for Workload Distribution in Dynamic Multi-Agents Systems *David Millá and J. Ignacio Hidalgo*
9:00 **9:25**	Communication-focussed approach for real-time neural simulation *Paul Fox and Simon Moore*
9:30 **9:55**	Effective Mutation Operator for Nurse Scheduling by Cooperative GA and Its Parallel Processing *Makoto Ohki*
10:30 **10:55**	Hybridizing Memetic Algorithms and Particle Filters for Visual Tracking on GPU *Raul Cabido, Antonio Sanz and Juan José Pantrigo Fernández*
11:00 **11:25**	P System Simulations on Massively Parallel Architectures *José María Cecilia, José Manuel García, Ginés D. Guerrero, Miguel A. Martínez del Amor, Mario J. Pérez-Jiménez and Manuel Ujaldon*
11:30 **11:55**	GPU-Accelerated Genetic Algorithms *Rajvi Shah, P J Narayanan and Kishore Kothapalli*

Workshop Dinner

7:00 **open end**	For a casual dinner on Saturday we have booked a table at the **Wirtshaus Biergarten Zattl** (Freyung 6, 1010 Vienna). Workshop and tutorial participants are welcome to join. Note, the dinner is not included in the fees. You have to pay for your own consumption.

Program of Sunday, September 12

On Sunday we have a workshop in the morning and a tutorial in the afternoon:

Workshop 3:

Programming Models for Emerging Architectures (PMEA)

Xavier Martorell, Rosa M. Badia, Marc Gonzàlez and *Alejandro Duran*

Sitzungssaal, Sunday, September 12

8:30 **9:30**	**Keynote: *Calin Cascaval*** Power Programming
9:30 **10:00**	Mapping a Class of Applications in Heterogeneous Multithreading Architectures *J. Lucena, O. Plata, N. Guil*
10:30	Enjing - a JIT Backend for CUDA Devices *M. Bentsen, B. Vinter*
	Combining Processor Virtualization and Component-Based Engineering in C for Many-Core Heterogeneous Embedded MPSoCs *E. Rohou, A. Ornstein, A. Özcan, M. Cornero*
12:00	QoS-ocMPI: QoS-aware on-chip Message Passing Library for NoC-based Many-Core MPSoCs *J. Joven, F. Angiolini, D. Castells-Rufas, G. De Micheli, J. Carrabina-Bordoll*

Tutorial 3:

Automatic Parallelization Techniques and the Cetus Source-to-Source Compiler Infrastructure

Rudi Eigenmann and *Sam Midkiff (Purdue University)*

Sitzungssaal, Sunday, September 12, 1:30 pm – 5:00 pm

The increased importance of parallelism has made parallelizing compiler technology, and easy-to-learn and use compiler infrastructures implementing this technology, increasingly important to researchers, developers and students in the fields of computer architecture, compilers and high performance applications. This tutorial will cover basic parallelizing compiler technology, including dependence analysis, dependence breaking transformations, optimizing transformations, and limits on compiler technology. We will then describe the Cetus source-to-source restructuring compiler infrastructure for C programs which is already used by a substantial number of research projects around the world. Cetus is a freely available, open source community compiler developed with support from the National Science Foundation. Its main distinction from related infrastructure efforts is its focus on high-level source-to-source translation for C programs and abstract internal representation. These features have already proven to enable highly efficient design and implementation of new compilation techniques. The tutorial aims to reach a wider audience and provide guidance for the use of the resource and its advanced optimization techniques. These techniques include new symbolic analysis methods, such as range analysis, automatic parallelization for multicores, and optimizations for heterogeneous multicores.

Topics Covered:

The first part of the tutorial will consist of an introduction to parallelizing compiler technology. Topics include:

- Dependence analysis and its relation to parallelization;
- Dependence eliminating transformations (i.e. privatization and expansion, forward substitution)
- Program optimizations, including vectorization and parallelization, parallelization of fully and partially parallel loops;
- Issues raised by optimizing explicitly parallel programs;
- Limits on compiler technology, and techniques such as cloning and profiling to overcome these.

The second part of the tutorial introduces Cetus, a source-to-source restructuring compiler infrastructure for C programs. Cetus is a community resource developed in support by the National Science Foundation. The infrastructure is available at cetus.ecn.purdue.edu. The tutorial will cover:

- Introduction to Cetus' capabilities;
- Internal abstract program representation;
- Optimization and analysis passes currently available in Cetus, including symbolic analysis methods, such as range analysis, automatic parallelization for multicores, and optimizations for heterogeneous multicores;
- Roadmap of ongoing and future enhancements.