Target Audience:

This tutorial is targeted primarily at application developers, computer/computational scientists, and graduate students interested in performance optimization issues and/or compilers for high-performance computing on general-purpose multicores and GPUs. Knowledge of C programming will be assumed; basic knowledge of processor architectures will be assumed; no prior parallel programming experience or familiarity with source-to-source transformations will be assumed.

Brief Description:

On-chip parallelism with multiple cores is now ubiquitous. Because of power and cooling constraints, recent performance improvements in both general-purpose and special-purpose processors have come primarily from increased on-chip parallelism from multiple cores rather than increased clock rates. Parallelism is therefore of considerable interest to a much broader group than developers of parallel applications for high-end supercomputers. Several programming environments have recently emerged in response to the need to develop applications for graphics processing units (GPUs) and multicore processors. This tutorial will address the following topics:
* What are the currently available programming models and API's for explicit parallel programming of multi-core CPUs and GPUs?

* What are the fundamental issues in achieving a significant fraction of peak performance with multicore CPUs and GPUs?

* What are some of the current efforts at providing more convenient high-level frameworks for programming GPUs? What are the compiler optimization challenges that these frameworks address?

TUTORIAL OUTLINE:

The format will consists of lectures and demos (demos if full-day). We can do either a half-day version or a full-day version, by essentially going over the same topics listed below but adjusting the depth of coverage.

1. Introduction
   - Multicore architectures and accelerators
   - Issues in performance and energy
   - GPUs
   - Explicitly managed memories
   - Programming Models:
     * Multi-threaded SIMT
     * Short-vector SIMD
     * Higher level models

2. GPU architectures and programming
   - GPU architectures
   - General-purpose computation on GPUs
   - Programming models and idioms
   - GPU programming models/environments:
     * OpenCL
     * CUDA
     * Directive-based programming
     * PGI Accelerator
     * CAPS/HMPP
     * OpenACC
     * Code examples on GPUs
     * Examples of CPU vs. GPU performance

3. Compiler optimizations and tuning for multicore
   - Brief review of data dependences, transformations
   - Polyhedral models, tiling, parametric tiling
   - Discussion of Pluto compiler for multicores
   - Locality and parallelism optimizations
- SIMDization
- Compiler-driven tuning

4. Compiler optimizations and tuning for GPUs
   - Performance characterization
   - Optimizing memory accesses
   - Multi-level parallelism exploitation
   - Performance models and empirical search
   - Compiler-driven tuning
   - Software managed memory hierarchies
   - Compilation strategies for GPUs in
     * Pluto (0.6.2 for GPUs)
     * R-Stream
     * Par4All
     * PGI Accelerator
   - Examples of application optimization

SHORT BIOGRAPHIES OF TUTORIAL SPEAKERS:

J. (Ram) Ramanujam received the B. Tech. degree in Electrical Engineering from the Indian Institute of Technology, Madras, India in 1983, and his M.S. and Ph. D. degrees in Computer Science from The Ohio State University in 1987 and 1990 respectively. He is currently the John E. and Beatrice L. Ritter Distinguished Professor in the Department of Electrical and Computer Engineering at Louisiana State University (LSU). In addition, he holds a joint faculty appointment with the LSU Center for Computation and Technology. His research interests are in compilers and runtime systems for high-performance computing, domain-specific languages and compilers for parallel computing, embedded systems, and high-level hardware synthesis. He has participated in several NSF-funded projects including the Tensor Contraction Engine (TCE) and the Pluto project for automatic parallelization. Additional details can be found at http://www.ece.lsu.edu/jxr/.

P. (Saday) Sadayappan received the B. Tech. degree from the Indian Institute of Technology, Madras, India, and an M.S. and Ph. D. from the State University of New York at Stony Brook, all in Electrical Engineering. He is currently a Professor in the Department of Computer Science and Engineering at The Ohio State University. His research interests include compiler/runtime optimization for parallel computing, and domain-specific languages for high-performance scientific computing. He has led several NSF-funded projects including the Tensor Contraction Engine and the Pluto project for automatic parallelization. Additional details can be found at http://www.cse.ohio-state.edu/~saday/.