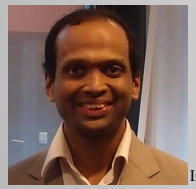
Faculty Information

Sanjay Rajopadhye (http://www.cs.colostate.edu/~svr/CV/vita-feb17.pdf) is professor in the Computer Science and in the Electrical Departments at Colorado State University. He teaches CS560, "Foundations of Fine-grain Parallelism" (see http://www.cs.colostate.edu/~cs560/Fall2015) and "Parallel Programming," on a regular basis. Some modules of this course will be drawn from CS475, and the others come from CS560.

Rajopadhye is one of the inventors of the polyhedral model. Spanning more than 30 years of theoretical, academic research, polyhedral compilation has become a mature technology today. The problem of automatic parallelization can today be solved for a class of loop programs called Affine Control Loops (ACLs). The technology to parallelize such programs/program-fragments is now incorporated into production compilers. There are numerous research tools, such as the PLUTO system from OSU, the dHPF project at Rice University, the PIPS project at ENSMP, the tools of the PARKAS group in INRIA, Paris, Loopo, and Chill. More importantly, the polyhedral model is now included in a number of production compilers. IBM's XL compiler family now adopts this model as the intermediate representation for loops; the 4.4.0 release of gcc now includes this as part of the GRAPHITE project; LLVM uses this representation in the Polly set of tools; Reservoir Labs also uses it in their compilation engine, Par4All and Silkan, the commercial branch of PIPS also use this model.

In Rajopadhye's research group Mélange, efforts for polyhedral compilation and analysis are embodied in AlphaZ, an open source system, developed using rigorous model driven software engineering (MDE) techniques that allow it to be cleanly extensible. In addition to the core technology for equational program analysis and transformation, it has tools for code generation, as well as tools to integrate the polyhedral kernel into complete applications through systematic wrapper generation. The code generators produce parallel, tiled shared memory (OpenMP) code, as well as parametrically tiled MPI code. There are tools to perform automatic complexity minimization, to serialize reductions, and a number of parallel schedulers, together with a formal verifier that checks the legality of a proposed transformation.



Ramakrishna Upadrasta graduated with a PhD from University of Paris-Sud, France, and INRIA, Paris, where he worked under Prof. Albert Cohen. His 2013 dissertation was titled "Sub-Polyhedral Compilation using (Unit-)Two-Variables-Per-Inequality Polyhedra",in which he addressed the problem of improvement of scalability of algorithms that are used in loop-parallelization. His thesis used techniques from polyhedral compilation, abstract interpretation, combinatorial optimization and graph theory.
Upadrasta has a M.S from Colorado State University, USA, an M.Tech in Indian Institute of Science (IISc), Bangalore and a B.E in Electrical and Electronics Engineering from Andhra University, Visakhapatnam. Earlier, he was a visiting scientist at IISc, a research engineer at INRIA, Paris, a research scholar at Lawrence Livermore National Laboratories, USA and a compiler engineer in Hewlett Packard.

Upadrasta has been at IIT Hyderabad (IITH) since March-2014. At IITH, Upadrasta's "Scalable Compilers for Heterogeneous Architectures" group works on the LLVM compiler infrastructure as well as Polly that uses Polyhedral Compilation techniques for efficient code-generation for CPUs and GPUs. At IITH, Upadrasta regularly teaches undergraduate Compilers, and graduate level courses on Compiler Optimizations and Introduction to Compiler Engineering using LLVM.

Topics to be Covered

- Parallel computing using GPUs
- NVIDIA CUDA
- Performance analysis
- Performance tuning

Course Details

The course is organized as five 4-hour modules, to be delivered as a sequence of lectures and a set of hands-on lab sessions. Each module will also include associated assignments and tutorials to be completed before the start of the next module (there is some flexibility, depending on constraints of the local coordinator).
The modules will cover the following topics::

[1 module] GPU programming in CUDA
[2 modules] Optimizing and tuning GPU programs
[2 modules] Advanced optimization: tiling, bottleneck analysis (roofline)

Course Coordinator

Dr. Ramakrishna Upadrasta
Course Coordinator
Department of Computer Science and Engineering, IIT Hyderabad
Kandi, Sangareddy. 502 285
Telangana, India
Tel: (+91) 40-23018445
Email: gian.pga@iith.ac.in

GLOBAL INITIATIVE ON ACADEMIC NETWORK

GIAN

(5 Day Course)

Programming GPUs & Accelerators
A Principled, Quantitative Approach

March 5th-9th 2018

Under the aegis of
MINISTRY OF HUMAN RESOURCE DEVELOPMENT

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Organized by
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Kandi, Sangareddy. 502 285
Telangana

Objectives

The challenge facing computer systems researchers is to deliver on the so-called, "*Moore's Law of expectation*," that computing systems must get *better*, *cheaper*, and *faster*, at an exponential rate. To date, we have always delivered on this promise, but a number of factors make the current version more challenging. Since the end of Dennard scaling over a decade ago, power and energy have become *de-facto* constraints, and led to greatly increased on-chip parallelism in the form of multi-cores, many-cores, GPUs, and *accelerators*.

In the past, programmers were shielded from details of processor evolution. A single architectural abstraction (the von Neumann machine), and a single programming/algorithmic abstraction (the random access machine RAM) allowed a clean separation of algorithmic, programming and architectural advances. These abstractions are no longer valid, since multi- and many-cores end "La-Z-Boy Programming" approach, where programmers simply wait for the next generation of processors. Programmers, library writers, and application developers must be aware of parallelism and memory locality at multiple levels, or risk losing many of the Moore's Law gains of modern architectures.

However, parallel programming is difficult, especially as architectures are evolving rapidly. More importantly, the associated runtime systems, and programming languages, libraries and frameworks are also evolving. The above evolution raises a number of issues that this course aims to tackle.

- The short term goal is to learn the methods of writing parallel applications for two classes of modern architectures: multi-core microprocessors, and many-core accelerators (GPUs, Intel Xeon Phi, etc.) Programmers need to learn to develop highly tuned applications that can best exploit the emerging architectures of today.
- The medium term objective is to do this in a principled manner so that the skills can be easily transferred to other contexts and platforms that the student is likely to encounter in the future.
- Finally, the long-term goal is to enable foundational research to render the first two challenges moot. This can be achieved through automatic compilation and code generation tools, and will enable the "return to La-Z-Boy Programming." It uses a quantitative approach based on a mathematical formalism called the *Polyhedral Model*.

The *polyhedral model* provides, for an important and precisely defined class of computations, a basis for (i) describing/formalizing them, (ii) analyzing them using quantitative performance metrics, and (iii) transforming them to produce efficient parallelization.

Schedule of the Course

Dates: March 5th--9th 2018
Total no. of Days: 5

Lecture Schedule (Tentative)

Day 1: Getting Started with GPUs and CUDA. Reductions/Scans (with commutative and non-commutative) operators.
Day 2: CUDA next steps (matrix multiplication and its tuning)
Day 3: Tuning/optimization: bank conflicts, (more matrix multiplication)
Day 4: A full kernel: Back Propagation Learning
Day 5: Advanced Topics, tiling, autotuning (0/1 Knapsack problem)

Registration Fee (All modules)

	Registration Fee Last Date (23-Feb-2018)
Participants from abroad	US $500
Industry/Research Organizations	Rs. 30000/-
Academic Institutions:	Rs. 10000/-
Students	Rs. 1000/-

The above fee include **all** instructional materials, computer use for tutorials, 24 hr free internet facility. The participants will be provided with single bedded accommodation on payment basis.

Payments should be made in the form of a

- Electronic Fund Transfer : Name of the Bank: State Bank of India, IIT Kandi, Hyderabad, India. Branch code: 014182 SWIFT Code: SBIN0014182 (Within India) Account No.:30859878032 (Current A/c) Remittance from abroad using SWIFT code SBININBB762, IMCR CODE:502002528
- DD in favor of Registrar, IIT Hyderabad, Payable to SBI, IIT Kandi Branch, IFS Code: SBIN0014182.

The DD, a copy of ID proof issued by the organization mentioned in the registration form, together with registration form should be sent to the course coordinator address mentioned overleaf.

GIAN

A 5 Days Course
on
Programming GPUs & Accelerators
A Principled, Quantitative Approach

Registration Form

March 5th-9th 2018

Title: Mr. / Ms. / Mrs. / Dr.

Name: _____

Date of Birth: _____

Designation: _____

Organization*: _____

Mailing Address: _____

_____

Phone: _____

Email: _____

Registration Fee Enclosed (Select one)*:

Rs. 1000 / Rs. 10000 / Rs. 30000 / USD 500

Draft/Electronic fund transfer Details.

_____

_____

Accommodation Needed: Yes/No

* Proof to be submitted