NSF/TCPP Curriculum Initiative in Parallel and Distributed Computing (PDC) – Core Topics for Undergraduates

Coordinator: Sushil K. Prasad - Georgia State University
Panel Members: Almadena Chtchelkanova– NSF
Andrew Lumsdaine - Indiana University
Manish Parashar, - Rutgers
Yves Robert - INRIA, France
Arnold Rosenberg – Northeastern University
Alan Sussman - University of Maryland
Michael Wrinn, Intel

Curriculum Initiative Website: http://www.cs.gsu.edu/~tcpp/curriculum/index.php
SC-12 Panel, Nov 13, 2012, Room 355-BC
Panel Agenda

- **Introduction to NSF/TCPP Curriculum Initiative:**  
  Sushil Prasad – Georgia State
- **NSF Perspective:**  
  Almadena Chtchelkanova – NSF

Panel Questions:
- **Q1. Taking forward to higher level courses:** How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?
- **Q2. What is best?:** What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?
- **Q3. Down to the masses:** How can PDC topics be taught in the context of a computer literacy course?
Curriculum Working Group

• Chtchelkanova, Almadena - NSF
• Dehne, Frank - University of Carleton, Canada
• Gouda, Mohamed - University of Texas, Austin, NSF
• Gupta, Anshul - IBM T.J. Watson Research Center
• JaJa, Joseph - University of Maryland
• Kant, Krishna - NSF, Intel
• La Salle, Anita - NSF
• LeBlanc, Richard, University of Seattle
• Lumsdaine, Andrew - Indiana University
• Padua, David- University of Illinois at Urbana-Champaign
• Parashar, Manish- Rutgers

• Prasad, Sushil- Georgia State University
• Prasanna, Viktor- University of Southern California
• Robert, Yves- INRIA, France
• Rosenberg, Arnold- Northeastern and Colorado State University
• Sahni, Sartaj- University of Florida
• Shirazi, Behrooz- Washington State University
• Sussman, Alan - University of Maryland
• Weems, Chip, University of Massachussets
• Wu, Jie - Temple University
Why now?

• Computing Landscape has changed
  – Mass marketing of multi-cores
  – General purpose GPUs even in laptops (and handhelds)
• A student with even a Bachelors in Computer Science (CS) or Computer Engineering (CE) must acquire skill sets to develop parallel software
  – No longer instruction in parallel and distributed computing primarily for research or high-end specialized computing
  – Industry is filling the curriculum gap with their preferred hardware/software platforms and “training” curriculums as alternatives with an eye toward mass market.
Stakeholders

- CS/CE Students
- Educators – teaching core courses as well as PDC electives
- Universities and Colleges
- Employers
- Developers
- Vendors
- Authors
- Researchers
- NSF and other funding agencies
- IEEE Technical Committees/Societies, ACM SIGs,
- ACM/IEEE Curriculum Task Force
Current State of Practice

• Students and Educators
  – CS/CE students have no well-defined expectation of what skill set in parallel/distributed computing (PDC) they must graduate with.
  – Educators teaching PDC courses struggle to choose topics, language, software/hardware platform, and balance of theory, algorithm, architecture, programming techniques…
  – Textbooks selection has increasingly become problematic each year, as authors cannot keep up; no single book seems sufficient
  – Industry promotes whatever best suits their latest hardware/software platforms.
  – The big picture is getting extremely difficult to capture.
Curriculum Planning Workshops at DC (Feb-10)

• Goals
  – setup mechanism and processes which would provide periodic curricular guidelines
  – employ the mechanism to develop sample curriculums

• Agenda:
  – Review and Scope
  – Formulate Mechanism and Processes
  – Preliminary Curriculum Planning
    • Core Curriculum
    • Introductory and advanced courses
  – Impact Assessment and Evaluation Plan

Main Outcomes

- Priority: Core curriculum revision at undergraduate level
- Preliminary Core Curriculum Topics
- Sample Intro and Advanced Course Curriculums
Weekly Meetings on Core Curriculum (May-Dec’10; Aug’11-Feb’12)

**Goal:** Propose core curriculum for CS/CS graduates

- **Every individual** CS/CE undergraduate must be at the proposed level of knowledge as a result of their *required* coursework

**Process:** For each topic and subtopic

1. Assign **Bloom’s classification**
   - K= Know the term (basic literacy)
   - C = Comprehend so as to paraphrase/illustrate
   - A = Apply it in some way (requires operational command)

2. Write **learning outcomes**
3. Identify core CS/CE courses impacted
4. Assign number of hours
5. Write suggestions for “how to teach”
<table>
<thead>
<tr>
<th>Algorithms Topics</th>
<th>Bloom#</th>
<th>Course</th>
<th>Learning Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithmic problems</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>broadcast</td>
<td>C/A</td>
<td>Data Struc/Algo</td>
<td>The important thing here is to emphasize the parallel/distributed aspects of the topic</td>
</tr>
<tr>
<td>multicast</td>
<td>K/C</td>
<td>Data Struc/Algo</td>
<td>represents method of exchanging information - one-to-all broadcast (by recursive doubling)</td>
</tr>
<tr>
<td>scatter/gather</td>
<td>C/A</td>
<td>Data Structures/Algo</td>
<td>Illustrate macro-communications on rings, 2D-grids and trees</td>
</tr>
<tr>
<td>gossip</td>
<td>N</td>
<td></td>
<td>Not in core</td>
</tr>
<tr>
<td>Asynchrony</td>
<td>K</td>
<td>CS2</td>
<td>asynchrony as exhibited on a distributed platform, existence of race conditions</td>
</tr>
<tr>
<td>Synchronization</td>
<td>K</td>
<td>CS2, Data Struc/Algo</td>
<td>aware of methods of controlling race condition,</td>
</tr>
<tr>
<td>Sorting</td>
<td>C</td>
<td>CS2, Data Struc/Algo</td>
<td>parallel merge sort,</td>
</tr>
<tr>
<td>Selection</td>
<td>K</td>
<td>CS2, Data Struc/Algo</td>
<td>min/max, know that selection can be accomplished by sorting</td>
</tr>
</tbody>
</table>

4 Curriculum Areas
Architecture, Programming, Algorithms, Cross-cutting
Early Adopter Program

• Total 80 institutions worldwide
  – Spring-11: 16 institutions; Fall’11: 18;
  – Spring-12: 21; Fall-12: 25 institutions
  – Most from US (4 year to research institutions);
    • some from South America, A few from Europe, fewer from Asia (India, China).

• Fall-13 round of competition: Deadline June 30, 2013
  – NSF funded Cash Award/Stipend up to $2500/proposal
  – Which course(s), topics, evaluation plan?

• Instructors for core CS/CS courses such as CS1/2, Systems, Data Structures and Algorithms – department-wide multi-course multi-semester adoption preferred
  – Elective courses; graduate courses
EduPar Workshop Series

– EduPar-11 at Alaska, IPDPS-2011
  • Receive feedback from the Adopters
  • Stimulate discussion of curricular and other educational issues.
– EduPar-12 at Shanghai, IPDPS-2012
  • A regular satellite workshop
– *EduPar-13* will be at Boston in May 2013
Current Activities

– Curriculum Revision and Formal Curriculum Release
  • Revision through Fall 2011 and Spring/Summer 2012
  • Formal release by Dec 2012

– Educational Resource Website

– Interface to the Broader Community
  • ACM/IEEE taskforce for CS Curriculum revision CS-2013.
Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER)

- Develop **PDC core curricula** flexible enough for a broad range of programs and institutions; collaborate with all stakeholders
- Develop, collect, and synthesize **pedagogical and instructional materials** for teaching PDC curriculum topics*
- Facilitate access to state-of-the-art **hardware and software resources** for PDC instruction and training by instructors and students*
- Organize Early Adopter Competitions and EduPar workshops, and related **events***

* Call for contributions
Acknowledgements

- NSF: Primary Sponsor
- Intel: Early Adopters
- IBM: EduPar Workshop
- NVIDIA: Early Adopters
Panel Agenda

- **Introduction to NSF/TCPP Curriculum Initiative:**
  Sushil Prasad – Georgia State

- **NSF Perspective:**
  Almadena Chtchelkanova – NSF

Panel Questions:
- **Q1. Taking forward to higher level courses:** How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?
- **Q2. What is best?:** What software or hardware platform, or **programming language**, are best for lower level undergraduate core courses to introduce parallelism?
- **Q3. Down to the masses:** How can PDC topics be taught in the context of a **computer literacy course**?
NSF/TCPP Curriculum in Parallel and Distributed Computing

NSF Perspective

Almadena Chtchelkanova
Program Director, NSF

SC-12 HPC Educators Program Chair
NSF-Supported Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER)

- Develop **PDC core curricula** flexible enough for a broad range of programs and institutions; collaborate with all stakeholders
- Develop, collect, and synthesize **pedagogical and instructional materials** for teaching PDC curriculum topics
- Facilitate access to state-of-the-art **hardware and software resources** for PDC instruction and training by instructors and students
- Organize Early Adopter Competitions and EduPar workshops, and related **events**
Panel Questions

• **Q1. Taking forward to higher level courses:** How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• **Q2. What is best?:** What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• **Q3. Down to the masses:** How can PDC topics be taught in the context of a computer literacy course?
PDC Algorithmics that “Stays with You”

Arnold L. Rosenberg
Northeastern University
Boston, MA 02115, USA

≈≈≈≈≈≈≈≈

OR

≈≈≈≈≈≈≈≈

A Proposal for Teaching PDC Algorithmics in the Core Curriculum

Arnold L. Rosenberg
Northeastern University
Boston, MA 02115, USA
The Proposal in a Nutshell

PRESENT ALGORITHMS VIA DEPENDENCY GRAPHS, AS WELL AS DETAILED SPECIFICATIONS

THIS STRESSES WHY AN ALGORITHM IS PARALLELIZABLE, AS PART OF ITS SPECIFICATION/ANALYSIS

WE PRESENT TWO EXAMPLES THAT ILLUSTRATE THIS APPROACH.
Butterfly-Like Computations, 1

- The comparator transformation

\[ y_0 = \min(x_0, x_1) \quad \text{and} \quad y_1 = \max(x_0, x_1) \]

leads to comparator-based sorting algorithms

- The convolution transformation

\[ y_0 = x_0 + \omega x_1 \quad \text{and} \quad y_1 = x_0 - \omega x_1 \]

(\(\omega\) is a constant associated with this specific butterfly building block) leads to convolutional algorithms

—such as the FFT or polynomial multiplication
Butterfly-like computations, 2

Composing butterflies to obtain complete computations (sorters or convoluters):

Exposing the parallelism:
The Parallel-Prefix/Scan Operator, 1

The *-parallel prefix of \((x_1, x_2, \ldots, x_n)\) (* is a binary associative operation):

\[
\begin{align*}
y_1 &= x_1 \\
y_2 &= x_1 \ast x_2 \\
y_3 &= x_1 \ast x_2 \ast x_3 \\
&\vdots \\
y_n &= x_1 \ast x_2 \ast \cdots \ast x_{n-1} \ast x_n
\end{align*}
\]

can be computed in \(O(\log n)\) parallel steps (via \(n\)-fold parallel computation):

\[
\begin{align*}
\text{for } j = 0 \text{ to } \lfloor \log_2(n-1) \rfloor \text{ do} \\
\text{for } i = 2^j \text{ to } n-1 \text{ do in parallel} \\
&x_i \leftarrow x_{i-2^j} \ast x_i \text{ do in parallel} \\
&y_i \leftarrow x_i
\end{align*}
\]
The Parallel-Prefix/Scan Operator, 2

The 8-input parallel-prefix dependency structure:

Exposing the parallelism:
Applying the Scan Operator: The Discrete Laplace Transform

The $n$-dimensional Discrete Laplace Transform (the Z-Transform) transforms $\langle x_0, x_1, \ldots, x_{n-1} \rangle$ to a vector $\langle y_0(\omega), y_1(\omega), \ldots, y_{m-1}(\omega) \rangle$ where

$$y_k(\omega) = x_0 + x_1 \omega^k + x_2 \omega^{2k} + \cdots + x_{n-1} \omega^{(n-1)k} = \sum_{i=0}^{n-1} x_i \omega^{ik}.$$ 

if $[y_0 = x_i \cdot \omega^{ik}]$ and $[y_1 = x_j \cdot \omega^{jk}]$ then $[z = y_0 + y_1]$
Applying the Scan Operator: Computing paths in a graph

Given a 9-node graph, presented via its $9 \times 9$ boolean adjacency matrix $A$. Compute a $9 \times 9$ matrix $M$ whose $(i, j)$ entry is $\vec{v}_{i,j} = \langle \beta_{i,j}^{(1)}, \ldots, \beta_{i,j}^{(8)} \rangle$

where $\beta_{i,j}^{(k)} = \begin{cases} 1 & \text{if there is a path of length } k \text{ between nodes } i \text{ and } j \\ 0 & \text{if there is no such path} \end{cases}$
Panel Questions

• Q1. Taking forward to higher level courses: How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• Q2. What is best?: What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• Q3. Down to the masses: How can PDC topics be taught in the context of a computer literacy course?
UTF-8
How do we present PDC concepts in lower level core curriculum?

- Motivate concepts using the real-world – build on their experiences
  - Most students are already exposed to concurrent, multitasking and connect world at work and play
    - WWW, C/S, GPUS, etc.

- Teach how to solve a parallel problem, rather than how to parallelize a solution to problem
  - Most concepts naturally follow, e.g., communication, synchronization, load-balancing

- Make problem-solving (coding) an integral part
Panel Questions

• **Q1. Taking forward to higher level courses:** How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• **Q2. What is best?:** What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• **Q3. Down to the masses:** How can PDC topics be taught in the context of a computer literacy course?
NSF/TCPP Curriculum Initiative in Parallel and Distributed Computing

Yves Robert

University of Tennessee Knoxville
ENS Lyon & Institut Universitaire de France

SC’2012 TCPP Panel
Introducing PDC material to undergraduates

Action

- Students are trained to think sequentially: one thing after the other, always
- Expose potential parallelism when exposing sequential algorithms

Examples

- Algorithms: divide-and-conquer algorithms (e.g. merge sort, discuss parallel complexity too)
- Architecture: pipelined CPUs (discuss throughput)

At the end of the day
Understand that everything (in geek life) is parallel in essence 😊
Best environment for beginners

On best language
- Never introduce a new language (would spend more time detailing its intrinsics than exposing PDC stuff)
- If it means Java, then go for Java

My favorite approach
- Keep using the white board to expose concepts (PRAM algorithms, scheduling, load-balancing)
- Provide students with skeletons (MPI-based) and let them write only core procedures
Panel Questions

• Q1. Taking forward to higher level courses: How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• Q&A

• Q2. What is best?: What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• Q3. Down to the masses: How can PDC topics be taught in the context of a computer literacy course?
Panel Questions

• **Q1.** Taking forward to higher level courses: How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• **Q2.** What is best?: What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• **Q3.** Down to the masses: How can PDC topics be taught in the context of a computer literacy course?
NSF/TCPP Curriculum Initiative in Parallel and Distributed Computing
SC12 Panel

Alan Sussman
UMIACS and Dept. of Computer Science
University of Maryland
Hardware Platform

- Shared memory, for sure
- Cluster, MPI, distributed memory and message passing too low level, so obscures thinking about parallel algorithms
- GPU, accelerators, etc. too messy, too heterogeneous, not yet standardized
- Shared memory is everywhere
  - supercomputers, desktops, laptops, mobile devices
Software Platform / Programming Language

• Whatever works for the instructor, and for the class topic
  – Java threads – OO programming
  – Posix threads – systems programming
  – OpenMP – scientific programming
  – OpenACC – GPU programming
  – Matlab – engineers
  – SQL – Database systems
  – Client/server – web apps
  – ...

Panel Questions

• Q1. Taking forward to higher level courses: How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• Q2. What is best?: What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• Q3. Down to the masses: How can PDC topics be taught in the context of a computer literacy course?
NSF/TCPP Curriculum Initiative in Parallel and Distributed Computing
SC12 Panel

Michael Wrinn
University Research Office
Intel Labs
Parallel Computing landscape, platforms

• Symmetric execution units, shared memory
  – Traditional, ubiquitous, intuitive
• Heterogeneous systems, “shared proximity”
  – Mobile devices (overtaking traditional clients)
  – SoC
  – Accelerators/GPU
  – Mixed core types
• Distributed execution units
  – Cluster/supercomputer/data center
  – Mobile applications (e.g. handset/cloud combo)
Parallel Computing landscape, software

• Imperative languages, shared state (threads)
  – OpenMP, Java threads, TBB, Posix, OpenCL etc

• Imperative languages, messaging
  – MPI

• Declarative languages
  – F#, ML, Haskell etc
  – Erlang (messaging)
Parallel Computing landscape

- Symmetric execution units, shared memory
  - Traditional, ubiquitous, intuitive

- Imperative languages, shared state (threads)
  - OpenMP, Java threads, TBB, Posix, OpenCL etc

- Heterogeneous systems, “shared proximity”
  - Mobile devices (overtaking traditional clients)
  - SoC
  - Accelerators/GPU
  - Mixed core types

- Imperative languages, messaging
  - MPI

- Declarative languages
  - F#, ML, Haskell etc
  - Erlang (messaging)

- Distributed execution units
  - Cluster/supercomputer/data center
  - Mobile applications (e.g. handset/cloud combo)
Panel Questions

• Q1. Taking forward to higher level courses: How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• Q2. What is best?: What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• Q3. Down to the masses: How can PDC topics be taught in the context of a computer literacy course?
NSF/TCPP Curriculum Initiative in Parallel and Distributed Computing

Andrew Lumsdaine
Computer Science Department
Indiana University

SC'12
November 12, 2012
What is the best x?

• Scheme
Culture Wars (What is the best X?)

• A common refrain in computer science undergrad education: “the choice of language is incidental – we want to teach concepts”

• And then the fights over language begin

• Two pragmatic approaches
  – Whoever is teaching the course should be empowered to teach it as best (s)he can
  – We need to keep students interested – meet students where they are

• The refrain is true
Panel Questions

• Q1. Taking forward to higher level courses: How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• Q2. What is best?: What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• Q&A

• Q3. Down to the masses: How can PDC topics be taught in the context of a computer literacy course?
Panel Questions

• Q1. Taking forward to higher level courses: How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• Q2. What is best?: What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• Q3. Down to the masses: How can PDC topics be taught in the context of a computer literacy course?
Computer Literacy

Any Sufficiently Advanced Technology is Indistinguishable from Magic

IT'S NEVER DONE THAT BEFORE!
A GUIDE TO TROUBLESHOOTING WINDOWS XP

JOHN ROSS
Parallelism for the masses

• Not just nice – and imperative
• All students we meet have been using computers all their lives
• They have deeply held intuitions about what computers are:
  – Interactive
  – Multitasking
  – Media rich
  – Connected
• And they all use client/server parallelism
  – And the cloud and peer-to-peer and GPUs and…
• A number of these things are direct results from PDC
• All students should know the difference between sufficiently advanced technology and magic
Panel Questions

• Q1. Taking forward to higher level courses: How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• Q2. What is best?: What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• Q3. Down to the masses: How can PDC topics be taught in the context of a computer literacy course?
Literacy for All

Arnold L. Rosenberg
Northeastern University
Boston, MA 02115, USA
Panel Questions

• Q1. Taking forward to higher level courses: How do we present PDC concepts in lower level core to maximize preparation for upper level concentrations?

• Q2. What is best?: What software or hardware platform, or programming language, are best for lower level undergraduate core courses to introduce parallelism?

• Q3. Down to the masses: How can PDC topics be taught in the context of a computer literacy course?

• Q&A
NSF/TCPP Curriculum Initiative in Parallel and Distributed Computing (PDC) – Core Topics for Undergraduates

Coordinator: Sushil K. Prasad - Georgia State University
Panel Members: Almadena Chtchelkanova – NSF
Andrew Lumsdaine - Indiana University
Manish Parashar, - Rutgers
Yves Robert - INRIA, France
Arnold Rosenberg – Northeastern University
Alan Sussman - University of Maryland
Michael Wrinn, Intel

Curriculum Initiative Website: