IEEE-TCPP Parallel and Distributed Computing Curriculum for Computer Science and Engineering Undergraduates

Sushil K Prasad

National Science Foundation
Georgia State University
Former Chair, IEEE Technical Committee on Parallel Processing (TCPP)

Sheikh Ghafoor, Tennessee Tech University
David Bunde, Knox College
Martina Barnas, Indiana University

EduPar-19, Reo de Janeiro, Brazil
TCPP Curriculum Initiative:
http://www.cs.gsu.edu/~tcpp/curriculum/
Outline

• IEEE-TCPP Curriculum – 30 mins
  – Why this curriculum initiative and what are the opportunities for the audience?
  – Key Activities and Milestones
    • ACM/IEEE 2013 CS Curriculum Taskforce
      – provided direct link to us for rigorous coverage
    – How was the curriculum formulated?
    – How is it getting evaluated?
  – Resources
    – Curriculum Revisions - Version 2
• Discussions – 30 mins
Who are we?

- 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 – George Mason University
- La Salle, Anita - NSF
- LeBlanc, Richard, Seattle University
- 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
- 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.
How was the curriculum formulated?

Why would they come?

*Field of Dreams (1989):* "If you build it, he will come"
Curriculum Planning Workshops at DC (Feb-10) and at Atlanta (April-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
Some Participants at the Planning Workshop, Washington DC, Feb 5-6, 2010
Weekly Tele-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”
### 4 Curriculum Areas
Architecture, Programming, Algorithms, Cross-cutting

<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>The important thing here is to emphasize the parallel/distributed aspects of the topic</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>represents method of exchanging information - one-to-all broadcast (by recursive doubling)</td>
</tr>
<tr>
<td>multicast</td>
<td>K/C</td>
<td>Data Struc/Algo</td>
<td>Illustrate macro-communications on rings, 2D-grids and trees</td>
</tr>
<tr>
<td>scatter/gather</td>
<td>C/A</td>
<td>Data Structures/Algorithms</td>
<td></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>
How is the Curriculum being evaluated?

Early Adopter Program
EduPar/EduHPC/Euro-EduPar Workshop series
Early Adopter Program

• Over 100 institutions worldwide
  – Spring-11: 16 institutions; Fall’11: 18;
  – Spring-12: 21; Fall-12: 25 institutions, Fall-13: 25 institutions, Fall-14: 25, Fall-15: 13
  – Most from US (4 year to research institutions, one high school)
  – Some from South America, a few from Europe, fewer from Asia (India, China, Indonesia, Singapore), Middle East

• **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

• **CyberTraining Training Workshops - Summer 2019**
  – NSF/Intel funded Stipend up to $1500-5000/proposal
  – *Instructor training + adoption plans*
Edu* 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 of IPDPS
- **EduPar-13** in Boston + **EduHPC** Workshop at SC-13 + BOF at SIGCSE-14
- EduHPC-14 @ SC-14, Nov – New Orleans; EduHPC-15 in Austin, EduHPC-16, EduHPC-17, EduHPC-18 in Dallas
- EduPar-15 @IPDPS, May, India; EduPar-16, Chicago, EduPar-17 in Orlando; EduPar-18 in Vancouver, **EduPar-19 @ IPDPS**
- **Euro-EduPar** Aug 2015; Euro-EduPar-2016, EEP-2017, EEP-18, **EEP-19**
- **EduHiPC 2018 @ HiPC in Banglore** – for India and the region
  - EduHiPC 2018 @ HiPC in Hyderabad in Dec’19
- **EduHPC-19 @ SC in Denver in Nov’19**
CDER: Center for Parallel and Distributed Computing Curriculum Development and Educational Resources

- Develop **PDC core curricula** flexible enough for a broad range of programs and institutions; collaborate with all stakeholders
  - Version I released Dec 2012; **beta version 1.9 May’19**.
- 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*
  - CDER cluster - Multi-core, GPU, Shared/Distributed Memory, Hadoop/Spark
    - **request accounts for Fall-19 and Summer-19**
- Organize Early Adopter Competitions and EduPar workshops, and related events
Upload and Search Course Material

- **Type:**
  - Slides, Syllabus, Tutorial, Video
  - Animation, Article, Award, Blog, Book, Competition
  - Course Template, Course Module, Data
  - Hardware Access, Software/Tools
  - Proposal, Report

- **Courses:**
  - CS1, CS2, Systems, Data Structures and Algorithms, ...

- **NSF/TCPP Topic/Subtopic Classification:**

  ALGORITHMS
  - Parallel and Distributed Models and Complexity
  - Algorithmic Paradigms
  - Divide & conquer (parallel aspects)
  - Algorithmic problems

  ARCHITECTURE
  PROGRAMMING
  CROSS-CUTTING

  - **Now open** - Work in Progress
CDER Book Project

• Lack of suitable textbooks to integrate PDC topics into the core courses
  – CS1, CS2, Systems, and Data Structures and Algorithms

• Part I - For instructors: Basic Concepts and References on what and how to teach

• Part 2: For students: Supplemental teaching material for core courses

• 9 chapters
  – over 27K chapter downloads – free downloads

• 2nd Volume – Published Nov’19
  – Vol 3 – Early Adopter course and topic exemplars and accompanying resources

- **CHAPTER 1** Editors’ Introduction and Road Map

**PART 1 FOR INSTRUCTORS**

- CHAPTER 2 Hands-on Parallelism with no Prerequisites and Little Time
- CHAPTER 3 Parallelism in Python for Novices
- CHAPTER 4 Modules for Introducing Threads
- CHAPTER 5 Introducing Parallel and Distributed Computing Concepts in Digital Logic
- CHAPTER 6 Networks and MPI for Cluster Computing

**PART 2 FOR STUDENTS**

- CHAPTER 7 Fork-join Parallelism with a Data-Structures Focus
- CHAPTER 8 Shared-Memory Concurrency Control with a Data-Structures Focus
- CHAPTER 9 Parallel Computing in a Python-Based Computer Science Course
- CHAPTER 10 Parallel Programming Illustrated through Conway’s Game of Life

Free Preprint version of this CDER book: http://grid.cs.gsu.edu/~tcpp/curriculum/?q=cedr_book
# Curriculum Version II Activities

<table>
<thead>
<tr>
<th>New Aspects</th>
<th>Areas</th>
<th>Architecture</th>
<th>Algorithms</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Area Lead/Aspect Lead</strong></td>
<td>Chip Weems</td>
<td>Arnold Rosenberg</td>
<td>Alan Sussman</td>
</tr>
<tr>
<td>Exemplars</td>
<td>Sushil Prasad</td>
<td>Karen Karavanic, Eric Freudenthal</td>
<td>Erik Saule, Duane Merrill, David Bunde</td>
<td>David Brown, Eric Freudenthal</td>
</tr>
<tr>
<td>Distributed</td>
<td>Vaidyanathan Ramachandran</td>
<td>Vaidyanathan Ramachandran, Manish Parashar</td>
<td>Vaidyanathan Ramachandran, Costas Busch, Denis Trystram</td>
<td>Alan Sussman, Chi Shen</td>
</tr>
<tr>
<td>Big Data</td>
<td>Trilce Estrada</td>
<td>Craig Stunkel</td>
<td>Cynthia Phillips,</td>
<td>Debzani Deb</td>
</tr>
<tr>
<td>Energy</td>
<td>Krishna Kant, Craig Stunkel</td>
<td>Craig Stunkel, Karen Karavanic</td>
<td>Denis Trystram</td>
<td>John Dougherty</td>
</tr>
<tr>
<td>Crosscutting</td>
<td>Sheikh Ghafoor Arny Rosenberg Anshul Gupta</td>
<td>Craig Stunkel, Eric Freudenthal</td>
<td>Robert Robey, Martina Barnas</td>
<td>Sheikh Gafoor, Eric Freudenthal</td>
</tr>
</tbody>
</table>
**Big Data Aspect:**

Introduce data centric aspects that are reshaping the computing landscape and take advantage of applications in social media, health informatics, and business intelligence to motivate the need for parallel processing.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Algorithms</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehend hardware pressure (memory hierarchies, networking, I/O) because of data volume and speed</td>
<td>Apply general parallel processing principles like divide and conquer in a Map-Reduce context</td>
<td>Comprehend data parallel programming and locality issues</td>
</tr>
<tr>
<td></td>
<td>Knowledge how hardware limitations impact algorithmic design</td>
<td>Understand properties of system software for data storage, organization, and manipulation</td>
</tr>
</tbody>
</table>
**Energy aspect: areas and curriculum impacts**

- **Areas**
  - Parallel algorithm efficiency
  - Widely differing energy costs for data movement
  - Complexity of architecture
  - Power minimization for idle or lightly-loaded units
  - Technology advances

- **Curriculum impacts/focus:**
  - Energy-efficient system architectures (e.g., SMT, SIMD/vector, heterogeneous)
  - Simple architectures tend to be more energy-efficient than complex ones
  - Scalable algorithms (good speed-up) tend to be energy-efficient as well
  - Algorithms that increase “compute” to avoid data movement
    - Compute is low energy, data movement energy can be quite high
  - Techniques for reducing power for idle or lightly-loaded units
    - E.g., power management, power gating, clock gating, batch scheduling
Distributed Computing Aspect: Most Significant Changes

• Introduction of Distributed Computing Topics in early courses
  – Asynchrony and its impact on distributed systems (races, hazards, consistency)
  – Local knowledge and its impact on distributed systems (global operations, symmetry breaking)
  – Dependencies and Consistency
  – Performance metrics

• Introduction of Distributed Computing Topics in (advanced) elective course(s)
Programming Topics - Overview

• Main goal is to introduce parallel programming topics into intro programming, data structures, and systems classes

• High-level themes include:
  – **Paradigms and notations** – SIMD, shared memory, message passing, client/server, big data stack, threads, tasks, data parallel, etc.
  – **Semantics and correctness** – synchronization, concurrency defects, ...
  – **Memory models** – sequential consistency, weak consistency, ...
  – **Performance and energy** – computation and data decomposition, scheduling/mapping, data layout and locality, tools and metrics

• **Most topics at Bloom level C or K for intro courses**, but at a deeper level for upper-level courses (or deferred to upper-level completely, so at N Bloom level for intro courses)
Programming Topics - Updates

• Incorporated new programming topics related to distributed computing (e.g. client/server), big data (e.g., MapReduce), and power/energy

• Eliminated some topics from original guidelines completely, since no longer relevant

• Added small number of other topics missed in original guidelines, or newer ideas (e.g., accelerator programming)
Architecture Topics Overview

- **Classes of Parallelism** – types and forms in architectures
- **Memory Hierarchy** – aspects related to parallelism
- Floating Point – recognizing the importance of this in HPC systems
- **Performance Metrics** – benchmarks and measures
- **Power** – issues related to multicore, scaling
- **Data scaling** – volume, velocity, hardware support for big data
- **Synchronization** -- mechanisms and different time and distance scales
Changes

• Cost of data movement
• **Power/Energy**, density, static/dynamic, power efficiency
• HW support for data bound computation, large data storage
• **Delay**, atomic operations, network handshaking, system ID
• Deleted some topics that are no longer relevant, updated outcomes
  – Separated bloom levels, time, outcomes for each course, per topic
• New topics, outcomes for existing topics, related to energy
• New topics, outcomes for existing topics, related to distribution
Algorithms

- **Goal:** Encourage parallel and distributed thinking early on
- **Suggested Methodology:** Gently alter existing courses to incorporate parallel thinking
- **Introduce ideas in the tradition setting so they can be elaborated in advanced/elective courses**
  - Big Idea 1: *Parallel actors are inherently asynchronous*. Asynchrony may be useful, or may need to be tamed.
  - Big Idea 2: Recursive *divide-and-conquer* can be mapped to parallelism.
Three main topic areas:

- **Parallel and Distributed Models and Complexity**
  - basis for analyzing parallel algorithms and their performance

- **Algorithmic Techniques**
  - algorithmic kernels that can be used as building blocks for parallel algorithms

- **Algorithmic Problems**
  - parallel algorithms for solving common problems such as sorting, searching, multi-agent communication
Crosscutting Aspect

• Pervasive Topics
  • fundamental in nature,
  • appear repeatedly in different contexts
  • different sub areas of PDC (Programming, Architecture, Algorithms)
  • transcend PDC and in some cases computing in general
  • should be introduced in 1\textsuperscript{st} two years of curriculum

• Emerging/Advanced Topics
  • Should be introduced in upper division classes
Pervasive Topics

• Concurrency and sequential dependency
• Asynchrony
• Locality
• Performance Metric
Exemplar Group - Goal and Activities

● Create variety of exemplars to show some ways to adopt the curriculum
  ○ **Topic-level**: Slides and materials for a single topic for integrate into a course
  ○ **Course-level**: Examples of courses that adopt some portion of it
  ○ Give models/ideas for potential adopters and also check feasibility of the recommendations

● **Join us!**
  ○ Create exemplars from your courses and materials
  ○ Participate in meetings (Fridays 3:30-4:30 EST)
Topic-Level Exemplars

Ideally, topic-level exemplars contain the following components:

- Lecture slides
- Instructor notes
- Homework assignments
- Required and suggested readings
- Exam questions with answers and explanations
- Example problems to solve (applied-level topics only)

Topic-level exemplars should conform to the curriculum guidelines: Bloom level, approximate classroom time, course context, and learning objectives.
Topic Exemplar: Tasks and Threads

In the Programming section of the PDC curriculum, under Semantics and Correctness Issues:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Bloom</th>
<th>Hours</th>
<th>Where Covered</th>
<th>Learning Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks and threads</td>
<td>K</td>
<td>0.5</td>
<td>CS2, DS/A, Systems, Lang</td>
<td>Understand what it means to create and assign work to threads/processes in a parallel program, and know of at least one way to do that (e.g., OpenMP, Intel TBB, etc.)</td>
</tr>
</tbody>
</table>

Searching the CDER courseware repository shows the following exemplars:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Institution</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>TasksAndThreads (lecture notes).docx</td>
<td>Intel Corporation</td>
<td>21.5k</td>
</tr>
<tr>
<td>TasksAndThreads (suggested readings).docx</td>
<td>Intel Corporation</td>
<td>19.5k</td>
</tr>
<tr>
<td>TasksAndThreads (exam questions).docx</td>
<td>Intel Corporation</td>
<td>107.0k</td>
</tr>
<tr>
<td>TasksAndThreads (lecture slides).pptx</td>
<td>Intel Corporation</td>
<td>686.4k</td>
</tr>
</tbody>
</table>
Attributes of a course exemplar

- We used [ACM/IEEE-CS CS2013 Course-Exemplar Template](#) and added the following sections
  - What CDER topics are covered and in what context (traditional topics)?
  - What resources do you use to cover the CDER topics?
    - e.g., textbooks, programming languages, environments, websites
  - How are students assessed on the PDC topics?
    - What type, and number, of assignments?
      - papers, problem sets, programming projects, etc.
    - How long do you expect students to spend on completing assessed work?
## Course exemplar: Algorithms at Knox

<table>
<thead>
<tr>
<th>CDER topic</th>
<th>Topics covered</th>
<th>Bloom level</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDER_AL Parallel and distributed models and</td>
<td>Costs of computation (asymptotics, time complexity), Notions from scheduling</td>
<td>A</td>
<td>0.75</td>
</tr>
<tr>
<td>complexity</td>
<td>(dependencies, task graphs, work, makespan)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDER_AL Algorithmic techniques</td>
<td>Decomposition (recursion and divide &amp; conquer)</td>
<td>A</td>
<td>1.5</td>
</tr>
<tr>
<td>CDER_AL Algorithmic problems</td>
<td>Sorting (parallel mergesort)</td>
<td>C</td>
<td>0.66</td>
</tr>
</tbody>
</table>

**Learning goal (sorting):** Observe at least one parallel sorting algorithm together with analysis. Parallel merge sort is the simplest example, but other alternatives might be covered as well; more sophisticated algorithms might be covered in more advanced courses.
Sponsorship Acknowledgements

- NSF
  - NSF/TCPP Curriculum Initiative
  - Early adopter competitions (stipend, travel)
  - EduPar/EduHPC workshop series
  - CRI-ADDO CDER (2012-15)

- Intel (2011-)
  - International early adopter institutions (stipend, travel)
  - Curriculum Development, Courseware, Exemplars

- nVIDIA
  - GPU cards to 50+ early adopters from Spring'11, Fall'11 and Spring'12 rounds.

- IEEE TCPP, IBM
  - Keynotes in the past
Conclusion

• **Need to inculcate “parallel thinking” to all**
• **Core Curriculum Revision** is a community effort
  – Curriculum Initiative Website:
• EduHPC-19, Denver
  – PC Chair: Debdhani Deb
• EduHiPC-19, Hyderabad, India
  – PC Chair: Sheikh Ghafoor and Ashish Kuvelkar
• **CDER Resources**
  – [Book Project](#) - download free preprints
  – MPI/Spark [Cluster](#) - free access for courses
  – [Courseware](#)
Discussion

– External Reviewers of Beta 1.9 curriculum
  • Available May 30, 2019

– South American Perspectives

– What resources would be most useful for adoption in the short term?
In your opinion, what are the challenges for broader adoption of PDC topics in the undergraduate CS curriculum (Spring 2018 survey)?

1. Existing curriculum is densely packed and there is no room for PDC related core courses
2. Shortage of good textbooks that explain PDC topics appropriately for the undergraduate CS students.
3. Shortage of instructor resources
4. Lack of Faculty expertise in the concepts
5. Lack of in-house resources such as computing cluster
6. Lack of Departmental Support
7. Lack of Institutional Support
8. Other challenges

10. In your opinion, what are the challenges for broader adoption of PDC topics in the undergraduate CS curriculum?