NSF/IEEE-TCPP Parallel and Distributed Computing Curriculum

Revision Planning Meeting
April 27, 2017, Arlington, Virginia
Who We Are

Karen Karavanic, Portland State U.
Erik Saule, U. of North Carolina, Charlotte
Krishna Kant, Temple U.
Debzani Deb, Winston-Salem State U.
David Bunde, Knox College
Randy Bryant, Carnegie Mellon U.
John Dougherty, Haverford College
Chi Shen, Kentucky State U.
Eric Freudenthal, U. of Texas, El Paso
Richard LeBlanc, Seattle U.
Andrew Lumsdaine, PNNL
Cynthia Phillips, Sandia National Lab
Bob Robey, Los Alamos National Lab

Duane Merrill, NVIDIA
Allen Parish, USNA and ABET
Craig Stunkel, IBM
Sushil Prasad, NSF
Almadena Chtchelkanova, NSF
Jim Kurose, NSF
Vipin Chaudhary, NSF
R. Vaidyanathan, Louisiana State U.
Martina Barnas, Indiana U.
Sheikh Ghafoor, Tennessee Tech.
Charles Weems, U. of Mass, Amherst
Arnold Rosenberg, Northeastern U.
Alan Sussman, U. Maryland, College Park
And now a word from our sponsor...
Morning Agenda

- Overview of current guideline
- Constraints and guidance for the update
  - Our experience
  - Early adopter survey
- August 2015 NSF workshop recommendations
- How to organize, develop afternoon agenda
  - Groups, formation, overlap
Afternoon Agenda (TBD)

- Group Discussions
  - Distributed, Algorithms, Architecture, Programming, Crosscutting Topics, Emerging Topics, Upper Level Courses, Exemplars
- Exemplars of implementation
  - Within core courses (CS1, CS2, DS/Alg, Systems)
  - Unplugged and facilitated activities
  - Within local contexts (resources, preparedness)
- Review of group discussions
- Roadmap, timeline, action items, IPDPS meeting
Not on the Agenda

- Concepts
- Technologies
- Teaching techniques
- Starting the actual revision
- We all love to work on these things, but have to stay focused on planning
Motivation

- Multicore, manycore, GPU, cluster
- Standard libraries, language features
- Web programming, GUI, IoT, data science
- Everything has gone parallel and distributed
Motivation

- Multicore, manycore, GPU, cluster
- Standard libraries, language features
- Web programming, GUI, IoT, data science
- Everything has gone parallel and distributed

EXCEPT THE CURRICULUM
Typical C++ Textbook

- Sequence, branch, loop, function
- Arrays, vectors, strings, pointers
- Classes, inheritance, polymorphism, overloading
- Stream and file I/O
- Standard Template Library, Templates
- Exceptions
- Classical searching and sorting
- C subset of C++
Typical Java Textbook

- Sequence, branch, loop, method
- Arrays
- Objects, classes, inheritance, polymorphism
- Exception handling
- Abstract classes and interfaces
- JavaFX GUI
- Event-driven programming
- Recursion
- One book does include “Concurrency”
Typical Java Textbook

- Sequence, branch, loop, method
- Arrays
- Objects, classes, inheritance, polymorphism
- Exception handling
- Abstract classes and interfaces
- JavaFX GUI
- Event-driven programming
- Recursion
- One book does include “Concurrency”

In Chapter 23
At least it’s a start...

- After being in the ACM Curriculum for 4 years
- Not just a separate “Concurrency” course
- Rise of web programming and IoT courses
- Mostly 3rd semester or above
- Sophomore level systems classes
But more to do

• Need to move toward ideas such as
  • Parallelism being as basic as control structures
  • I/O being as much over a network as from a UI or files
  • Work being related to locality and energy
• From the very beginning
Existing Guideline

- Released 12/12
- Draft released 12/10 for comment
- Four areas:
  - Architecture
  - Programming
  - Algorithms
  - Cross Cutting Topics
- Designed with 5 to 7-year expected lifetime
Rationale Sections

• Architecture
  • Becoming more non-core
  • Needs to be more interwoven across core
  • Introduced to explain behaviors
  • Identify concepts with most direct impact
  • Abstract specifics into concepts that can stand the test of time
Rationale Sections

• Programming
  • Focus on major paradigms (vector, shared memory, distributed, hybrid)
  • Intersect with models (data parallel, task parallel, SPMD, etc.)
  • Notations (languages and libraries)
  • Correctness, semantics (synchronization, consistency, concurrency defects)
• Performance
Rationale Sections

- Algorithms
- Parallel/distributed models and complexity
- Algorithmic paradigms (e.g., divide and conquer, series-parallel, parallel-prefix)
- Algorithmic problems (e.g., leader election, termination detection, searching, sorting, graphs, matrices)
Rationale Sections

- Cross-cutting Topics
  - Aspects appear across all areas
  - Concurrency, nondeterminism, locality, fault tolerance, energy

- Advanced Topics
  - Things that are hot at the time
  - Expected to “evolve”
  - May or may not produce concepts to migrate into main areas
Bloom Level

- Different concepts can be learned to different levels of understanding in the core
- Shorthand guidance to implementors
- K = know the term
- C = comprehend so as to paraphrase/illustrate
- A = able to apply in some way
- N = not in core, but may be elective
- Sometimes multiple are indicated to provide options for coverage
Missing Bloom Levels

- Analyze (compare, contrast, infer, diagram)
- Evaluate (conclude, critique, appraise, defend)
- Create (devise, design, plan, compose, write)

By their nature, these levels demand a considerable investment of teaching time, study, and practice. For some topics a solid skill foundation would be necessary. That’s too much change to expect in the core for the first attempt.
Core Courses

- CS0 - Computer science principles
- CS1 - Intro to programming
- CS2 - Second programming course
- Systems - Intro to systems/organization
- Discrete math
- DS/A - Data structures and algorithms

Although this is a common structure for the first two years, it’s not universal
Advanced/Electives

- Elective architecture
- Elective algorithm design/analysis
- Programming languages
- Software engineering
- Parallel algorithms
- Parallel programming
- Compiler design
- Communication networks
- Distributed systems

These are not part of the guideline, but allow us to indicate where important topics would be covered that are too advanced for the core curriculum.
Hours

- For the core, which has a maximum Bloom level of Apply, we tried to indicate a low expectation

- A K-level topic might be a discussion of a few minutes in class, and appear on a glossary

- The A-level topics, which involved 1 to 2 hours, were minimized

Monday, May 1, 17
**Example**

<table>
<thead>
<tr>
<th>Algorithmic problems</th>
<th>8.5</th>
<th>The important thing here is to emphasize the parallel/distributed aspects of the topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>C/A</td>
<td>2</td>
</tr>
<tr>
<td>Broadcast</td>
<td>C/A</td>
<td>1</td>
</tr>
<tr>
<td>Multicast</td>
<td>K/C</td>
<td>0.5</td>
</tr>
<tr>
<td>Scatter/gather</td>
<td>C/A</td>
<td>0.5</td>
</tr>
<tr>
<td>Gossip</td>
<td>N</td>
<td>0.5</td>
</tr>
<tr>
<td>Asynchrony</td>
<td>K</td>
<td>0.5</td>
</tr>
<tr>
<td>Synchronization</td>
<td>K</td>
<td>1</td>
</tr>
<tr>
<td>Sorting</td>
<td>C</td>
<td>1.5</td>
</tr>
<tr>
<td>Selection</td>
<td>K</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Cross-Reference Table

- Topics are listed with a course matrix
- Enables users to look at a particular course and see which topics are suggested as appropriate
- Note that we don’t say required, or even recommended
Teaching Suggestions

• For each topic, a very brief set of examples of how it might be covered in different places is given

• Meant to stimulate creative thinking, open minds to possibilities, reduce anxiety

• Tried to use “could be,” “can,” and neutral descriptions versus “should” and “must,” although not entirely successful
Sample Elective Course

- Although we prefer topics to be introduced in the core, we recognize that for some institutions an elective is an easier approach (foot in the door philosophy)
- Outlines a single semester/quarter class
- Allocates time to add up to 45 hours
Questions?

- About the existing curriculum itself
- We will address the reasons for the approach and the directions for the revision in upcoming sections
How Not to Develop a Curriculum Guideline

- Make it intimidatingly large and thorough
- Imply that every concept is necessary
- Ignore diversity of institutions and students
- Assume availability of resources
- Assume that if we build it, they will come
- Omit popular topics from discussion
Why do we do it?

- For the students - to better prepare them
Why do we do it?

- For the students - to better prepare them
- What does that imply?
Why do we do it?

• For the students - to better prepare them
• What does that imply?
• Adoption is paramount
Why do we do it?

• For the students - to better prepare them
• What does that imply?
  • Adoption is paramount
    • If it doesn’t get adopted, even partially, then we fail to affect student’s lives
Why do we do it?

• For the students - to better prepare them
• What does that imply?
  • Adoption is paramount
    • If it doesn’t get adopted, even partially, then we fail to affect student’s lives
• Curriculum has to be adoptable
Why do we do it?

• For the students - to better prepare them
• What does that imply?
• Adoption is paramount
• If it doesn't get adopted, even partially, then we fail to affect student's lives
• Curriculum has to be adoptable
• Think warm and fuzzy puppy/kitten
Adoptability

- Consider diversity of adopters
- University, large 4-year college, small 4-year college, community college transfer courses, international contexts
- Semester and quarter systems, departments in science, engineering, math
- Instructor preparation and course load
- Student preparation
Adoptability

- Consider resource availability
  - Educational GPU cluster, a few multicore servers, standard laptops, nothing reliable
  - Varying levels of network access
  - Software licenses to only freeware
  - Time and space in existing curriculum
Adoptability

- Minimize impediments to adoption
- One size doesn’t fit all -- maximize flexibility
- Think in terms of instructor effort
- Look for ways that existing concepts can be enhanced, rather than replaced
- Focus on small changes that are easy
Warm and Fuzzy

• Not enough to just minimize impediments
• Changes have to be attractive, exciting, engaging, inspiring
• Not just abstractly how they will someday help students to get better jobs
• How will they make the class more interesting, fun, popular, and rewarding?
• Goal is to maximize effect on students via engagement
Fuzzy

- Concepts that have broad applicability
- Able to be taught at different Bloom levels
- Can be introduced early in simple form
- Revisited in multiple places to deepen
- Entry points into other concepts
Warm

- Clearly relevant
- Multiple ways to apply
- Easy to illustrate through engaging examples
- Clear value to enable powerful solutions
- Relatable to real-world technologies
- Incentivize adoption
Warm and Fuzzy

• What are some of your warmest and fuzziest experiences with changing a curriculum or course syllabus?

• What made it a positive experience?
Prior Experience

- Easy to list many concepts, technologies
- Hard to winnow down (it’s all important)
- Easy to be very specific
- Hard to generalize, make flexible, envision alternatives
- Easy to end up with duplication across areas
- Hard to reduce duplication while preserving breadth of applicability
Prior Experience

- Easy to think something is so fundamental that it should be required
- Hard to accept that different contexts have different priorities and shouldn’t be judged
- Easy to assume that because you can do it easily, it will be easy for others
- Hard to recall how hard it can be
Prior Experience

- Easy to assume innovation starts at the top
- Hard to recognize that necessity may be the best mother of invention
- Easy to pontificate
- Hard to listen
Incentives for Adoption

- Work with development of standards
- Provide resources (teaching materials, access to systems, software, texts)
- Provide grants for development
- Provide training
- Create community
- Publicize successful efforts
- Industry and government encouragement
Early Adopters Program

- Small grants, some with hardware donations from NVIDIA, international support from Intel
- Easy application process, twice per year
- About 140 awarded
- Opportunities to publish at EduPar, EduHPC, and on CDER web site, JPDC special issue
- Recently surveyed
Survey Review
August 2015 Workshop
Attendees

Organizers
Charles Weems, U. of Massachusetts, Amherst (CDER)
Alan Sussman, U. of Maryland, College Park (CDER)
Arnold Rosenberg, Northeastern U. (CDER)
Anshul Gupta, IBM, (CDER)
Sushil Prasad, NSF
Almadena Chuchelkanova, NSF
Amy Apon, NSF
Mimi McClure, NSF

Participants
Anyndia Banerjee, NSF
Randy Bryant, White House OSTP
Barbara Chapman, U. of Houston
Debzani Deb, Winston-Salem State U.
Akshaye Dhawan, Ursinus College
John Dougherty, Haverford College
Trilce Estrada, U. of New Mexico
Diana Franklin, U.C. Santa Barbara
Eric Freudenthal, U. of Texas, El Paso
Ajay Gupta, Western Michigan U.
Karen Karavanic, Portland State U.
George Karypis, U. of Minnesota
Dan Katz, NSF
Andrew Lumsdaine, Indiana U.
Brandeis Marshall, Spelman College
Duane Merrill, NVIDIA
Edusmildo Orozco Salcedo, U. of Puerto Rico
Cynthia Phillips, Sandia National Laboratory
Erik Saule, U. of North Carolina, Charlotte
Chi Shen, Kentucky State U.
Libby Shoop, Macalaster College
Michelle Strout, University of Arizona
Violet Syroitiuk, Arizona State U.
Michela Taufer, U. of Delaware
Dominique Theibaut, Smith College
R. Vaidyanathan, Louisiana State U.
Susan Wang, Mills College
Michael Wrinn, Intel

Monday, May 1, 17
Broadening Participation

- Prioritize topics for core courses
- Integrate with existing material
- Open student’s thinking to concepts that grow
- Enable solving problems that generate interest
- Keep 4-year path in mind
- Higher level perspective in earlier classes
- Consider different goals of non-majors and majors
- Get buy-in of students using exciting examples
- High cognitive load in early classes - don’t overload
- Create model courses
- Educate the educators
How to Educate Educators?

- Summer support for PD would generate interest
- Identify training models
- Ongoing workshops at conferences
- Incentives to Dept. Chairs, Deans (travel grants to attend workshops, hardware/software donations, certifications and awards)
- Curriculum experimentation grants tailored for smaller institutions
- Industrial help (complex relationship)
- Partnerships between institutions
How to Educate Educators?

• More specificity in the curriculum guideline
• Example courses
  • Implies need to consider different populations of educators and types of institutions
• Simplify organization of the curriculum
• Align more with a model
• Consider reducing references to HPC and emphasizing data science, machine learning, social networks, online gaming, IoT

Monday, May 1, 17
Upper Levels

• Opens opportunities in Data Analytics, ML

• Databases, Info. Retrieval, Bioinformatics, Algorithms, Networks, Scientific Computing, Distributed Services, Game Development

• Interdisciplinary Capstones, Service Learning, Entrepreneurship, REUs

• Changes approaches in OS, Algorithms, PL
Upper Levels

• Courses tend to use problems that are intellectually interesting for the concepts they illustrate

• Access to PDC resources enables students to solve problems that engage the senses while illustrating concepts

• OS without the pain
Upper Levels

- Specific projects:
  - Flash mob detection from Twitter feed
  - Multi-agent mobile/server interactive game
  - Identify global trends in geoscience data
  - Seam carving (graphics)
  - Game of life on cell phones
Guideline Recommendations

- Clarify descriptions of courses
- Mixing in upper level, zero-time, topics is confusing
- Show progression of Bloom levels through multiple courses
- Example curriculum frameworks, courses
- Make OS core
- More emphasis on distributed
- Delete Flynn’s taxonomy, benchmark averaging
Guideline
Recommendations

• scaling and efficiency
• power and energy
• hardware versus software threads
• atomicity
• ordering and consistency, weak consistency
• fences
• lock-free algorithms
• accelerators as a more general model
• more out-of-order processing
Guideline Recommendations

- Algorithms section is challenging
- Granularity is finer than other sections
- Possibly reorganize
- Possibly start from learning goals and rewrite
- Complexity model needs to shift from computation to locality and energy
- Move some topics to programming
- Shift emphasis from shared memory to message passing and distributed
Summary

• Reorganize and prioritize within areas
• Extend to four-year progression with suggested upper level topics/reinforcement
• Increase emphasis on distributed
• Specific changes to Arch and Prog
• Overhaul Algorithms
• Develop exemplars (curricula, courses, projects)
• Propose means for professional development
Further Thoughts?

- Ideas and experiences since the workshop to be considered as part of recommendations?
- New emerging trends?
- Other issues noticed in the guideline?
Break
Getting Organized

• What teams are needed?
• Keep areas but subdivide into core and upper?
• How to infuse distributed?
• How to build exemplars?
• How to structure communication between groups?
World Cafe

- Tables select a host at start of each round
- Host encourages participation, note making, remains for next round to summarize
- At end of round, redistribute to different tables to spread ideas
- At end of third round, identify common themes and gather
Getting Organized

• What teams are needed?
• Keep areas but subdivide into core and upper?
• How to infuse distributed?
• How to build exemplars?
Next Steps

- Volunteering for team roles
- IPDPS meeting and action items
- Outline schedule of work
## Volunteers

<table>
<thead>
<tr>
<th>Area Lead (Co-lead TBD)</th>
<th>Aspect Lead</th>
<th>Architecture</th>
<th>Algorithms</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chip Weems</td>
<td>Arnold Rosenberg</td>
<td></td>
<td>Alan Sussman</td>
</tr>
<tr>
<td>Exemplars</td>
<td>Karen Karavanic</td>
<td>Karen Karavanic, Eric Freudenthal</td>
<td>Erik Saule, Duane Merril, David Bunde</td>
<td>David Brown, Eric Freudenthal</td>
</tr>
<tr>
<td>Distributed</td>
<td>Krishna Kant?</td>
<td></td>
<td>Vaidyanathan Ramachandran, Denis Trystram</td>
<td>Chi Shen</td>
</tr>
<tr>
<td>Big Data</td>
<td></td>
<td>Craig Stunkel</td>
<td>Cynthia Phillips, Debzani Deb</td>
<td></td>
</tr>
<tr>
<td>Energy</td>
<td>Craig Stunkel</td>
<td>Craig Stunkel, Karen Karavanic</td>
<td>Denis Trystram</td>
<td>John Dougherty</td>
</tr>
<tr>
<td>Crosscutting</td>
<td>Randy Bryant</td>
<td>Craig Stunkel, Eric Freudenthal</td>
<td>Robert Robey, Martina Barnas</td>
<td>Sheikh Gafoor, Eric Freudenthal</td>
</tr>
</tbody>
</table>