Literacy for All in Parallel and Distributed Computing: NSF/IEEE-TCPP Guidelines for an Undergraduate Core Curriculum

Sushil K Prasad
Georgia State University
Former Chair, IEEE Technical Committee on Parallel Processing (TCPP)
SC-12 Invited talk, Nov 13, 2012, 255-D

Curriculum Initiative Website:

Linked through TCPP site: tcpp.computer.org
Outline

• Why this initiative and what are the opportunities for the audience?
• Key Activities and Milestones
  – How the community and experts from stakeholder sub-communities responded and were facilitated to participate?
  – Planning Workshops, Tele-meetings, EduPar, Early adopter competitions, Interface with 2013 ACM/IEEE CS curriculum Taskforce, CEDR center funding
• How was the curriculum formulated?
  – Blooms classification, learning outcomes, hours, which courses, how to teach
• Rationale for the resulting preliminary curriculum
• How is it getting evaluated?
  – Range of courses and early adopter institutions, feedback and resources, EduPar posters and talks
• Lessons learnt
  – How can you restructure your own core courses?
  – CS1, CS2, Data Structures, Systems,…?
• Examples of how PDC topics can be taught in core courses?
• Conclusions and Roadmap
  – Periodic curriculum update – Upcoming first version Dec 2012
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 - 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 Massachusetts
- 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.
Expected Benefits to other Stakeholders

• University and Colleges
  • New programs at colleges (nationally and internationally)
  • Existing undergraduate (and graduate) programs/courses need some periodic guidance
  • 2013 ACM/IEEE curriculum task force is now focussed on PDC as a thrust area

• Employers
  – Need to know the basic skill sets of CS/CE graduates
    – No well-defined expectations from students, but will increasingly require PDC skills
  – Retraining and certifications of existing professionals
Expected Benefits to Stakeholders

• Authors
  – Will directly benefit when revising textbooks
  – Are participating in the curriculum process
• NSF and Funding Agencies
  – Educational agenda setting
  – Help fund shared resources
• Sisters Organizations (IEEE TCs: TCPP, TCDP, TCSC, ACM SIGs, etc.)
  – Need help in setting their Educational Agenda
  – Can Employ this template elsewhere
• ACM/IEEE 2013 CS Curriculum Taskforce
How was the curriculum formulated?

Why would they come?

*Field of Dreams* (1989): "If you build it, he will come"
Why did the community and experts get on board?

- Timing and Community Need
- Transparency, inclusive
- Community outreach at all stages
- Thoroughness and quality
- Continual Feedback mechanisms
  - From experts and stakeholders
  - Early adopters
  - EduPar workshops
  - Curriculum Sessions/Panel
    - HiPC Dec 2010, India Goa
    - SiGCSE March 2011, Dallas
    - EduPar-11, Alaska
    - EduPar-12, Shanghai
    - SC-12 – this talk and panel at 3:30 pm, room
Dear TCPP member,

Like in each fall semester, I have begun to teach Introduction to Parallel and Distributed Computing course for seniors and first year graduate students. This is students’ first formal introduction to the area and they bring in varied backgrounds with lowest common denominator of some discrete mathematics, a few programming courses (for example, C/Unix), and assembly language or an architecture course.

Given the fluidity of information and rapid change in our broader area, each year I struggle to pick and choose topics, to find the right balance between theory and practice (architecture, algorithms, applications, programming - models, languages, and platforms), and to select a suitable textbook(s).

I do consult with a few colleagues, but most seem to be in the same quandary.

It has gotten harder each year, and I wish we had some curricular guidance from the community, or better, a standard core curriculum updated periodically.

Sushil
TCPP Newsletters and Timelines

• **October 2007:** Some of you sent me your syllabus for your first introductory course on Parallel and Distributed Computing. Please send yours.

• Fall 2007- Spring 2009 Followed up with community, experts, and NSF Program Directors

• Summer 2009: Secured NSF funding for Planning Workshop
Welcome to the first newsletter this Fall! I am honored to have received your enthusiastic support for my next term.

One key new initiative which has received wide support from the community is Curriculum Standards for Parallel and Distributed Computing.

TCPP is organizing an NSF-supported Planning Workshop on Curriculum Standards for Parallel and Distributed Computing to identify the needs, assess current state of practice, and propose roadmap in short and long term. TCPP invites all stakeholders for their participation in this process.

To begin with, educators worldwide are requested to provide information on the relevant courses that they teach at http://www.cs.gsu.edu/~tcpp/curriculum (undergraduate/graduate; introductory, advanced, and other related high performance and computational courses).
Curriculum Data Sampled

• 73 universities
• 91 courses – 18 courses through website upload, others manually scouted

• Separately categorized as
  – Parallel and Distributed Computing (PDC)
  – Parallel Algorithms (PA)
  – Computational Sciences/Applications
  – Also had distributed systems courses
Curriculum Data Sampled

- Statistics:
  - Topic-wise distribution for 51 PDC and Parallel Algorithms (PA) Courses
  - Topic-wise distribution for 40 PDC Courses
  - Topic-wise distribution for 11 Parallel Algorithms (PA) Courses
  - Courses uploaded through website had weekly distribution
Sample Statistics

- **PDC Topic Distribution**
- **Week-wise Av. Distribution of Uploaded Courses**
Planning Workshop Goals

• The primary goal of this planning workshop
  – setup mechanism and processes which would provide periodic curricular guidelines

• Secondary goal
  – employ the mechanism to develop sample curriculums

(My role: facilitator for the workshop; Coordinator for follow-up activities)
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
Workshop Outcomes – Curriculum
Workshop Outcomes - Minimum skill set for a CS/CE graduate

• An undergraduate student should be able to program parallel and distributed systems efficiently (productivity and performance)
  – Productivity
    • Languages, software engineering, programming methodologies, algorithms, parallel design patterns, tools, libraries
  – Performance
    • Execution time, power, memory, I/O, scalability, throughput
• To be aware of interactions among different tools, algorithms, architectures, programming models
• The knowledge should be relevant for the foreseeable future
  – e.g., multi-core, GPUs, web-services, clusters
• Common Themes
  – Concurrency, Nondeterminism, Locality
TCPP Newsletter - Feb 2010

– Last week, we held a successful TCPP/NSF Planning Workshop on Curriculum Standards for Parallel and Distributed Computing at Washington DC. The participants represented various stakeholders including NSF and sister organizations.

– Our roadmap in the short-term calls for a follow-up meeting at IPDPS, Atlanta, in April, and the release of preliminary undergraduate curricular guidance in early summer.

– Sign up to be in the loop and continue to upload your current course information at the initiatives website: http://www.cs.gsu.edu/~tcpp/curriculum/?q=workshop.
TCPP Newsletters

• April-10 (after IPDPS-10 in Atlanta)
  – We held a daylong NSF-supported curriculum initiative workshop on parallel/distributed computing meeting on Monday. A core curriculum is being proposed for CS/CE undergraduates. In addition, a few sample introductory and advanced courses are in the works. More detailed work continues, with a draft for community feedback expected early summer.

• Re: Meeting Minutes: TCPP excom/adcom meeting at IPDPS-10
  I need to form a nominations committee to review current advisory committee composition and later on coordinate TCPP chair election for 2011-13. I also need volunteers to serve in some of the positions in the executive committee and to help with new activities, such as curriculum, Body of Knowledge, seeding PhD forums at sponsored conferences, awards, newsletter, community email list, etc.
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”
TCPP announcement - Dec 21, 2010:  
NSF/TCPP Core Curriculum Initiative for Undergraduates – Call for Early Adopters

- As you know, a working group has been busy throughout this year on formulating a core curriculum on parallel and distributed computing (PDC).
- I am pleased to announce the release of the preliminary version of our proposed curriculum, now available at the Curriculum Initiative’s website: http://www.cs.gsu.edu/~tcpp/curriculum/index.php.
- I want to thank the members of this working group for their guidance, vision, and participation, and to the select group of reviewers at various stages. Special thanks go to the diehards among us ...
- We are seeking early adopters of the curriculum... Some seed level funds are available for early adopters (U.S. and internationally) with support from NSF and Intel.
- In tandem, we have also planned The 1st NSF/TCPP Workshop on Parallel and Distributed Computing Education on May16, 2011, in conjunction with IPDPS-11, Anchorage, Alaska.
How is the Curriculum being evaluated?

Early Adopter Program
EduPar Workshop series
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
Early Adopter Status Awarded - Spring 2011

1. Columbia University, USA
2. Hampton University, USA
3. Georgia Institute of Technology, USA
4. Washington and Lee University, USA
5. University of Central Florida, USA
6. Loyola University Chicago, USA
7. Wittenberg University and Clemson University, USA
8. University of Georgia, USA
9. Calvin College, USA
10. Arizona State University, USA
11. Universidad Nacional de Río Cuarto, Argentina
12. University of Pannonia, Hungary
13. Kassel University, Germany
14. Knox College, USA
15. International Institute of Information Technology, Hyderabad, India
16. Universidad Nacional del Sur, Argentina
Early Adopter Status Awarded - Fall 2011

1. St. Olaf College, USA
2. Kent State University, USA
3. Georgia State University, USA
4. NC A & T State University, USA
5. Ursinus College, USA
6. Southwest Baptist University, USA
7. University of Central Arkansas, USA
8. Florida State University, USA
9. Texas A&M University - Corpus Christi, USA
10. Texas Tech University, USA
11. Southern Polytechnic State Univ, USA
12. University of Puerto Rico, USA

13. University of Murcia, Spain
14. Universidad Nacional de La Plata, Argentina
15. Universidad Tecnológica Nacional, Facultad Regional Bahía Blanca, Argentina
16. Universidad Nacional de San Luis, Argentina
17. Universidad de Buenos Aires, Argentina
18. Universidad Tecnológica Nacional – Facultad Regional Mendoza, Argentina
Early Adopter Status Awarded - Spring 2012

1. North Carolina State University, USA
2. Moravian College, USA
3. Western Oregon University, USA
4. Ohio University, School of EECS, USA
5. University of Cincinnati, USA
6. Purdue University, USA
7. University of Illinois at Urbana Champagne, USA
8. University of Massachusetts Amherst, USA
9. Prairie View A&M University, USA
10. Radford University, USA
11. University of Utah, USA
12. University of Colorado Boulder, USA
13. University of Houston-Downtown, USA
14. Institut Teknologi Bandung, Indonesia
15. Universidade Federal de Campina Grande, Brazil
16. Middlesex College University of Western Ontario, Canada
17. Universidad Nacional de Córdoba - FaMAF, Argentina
18. University of Victoria, Canada
19. Jadavpur University, Kolkata, India
20. Departamento de Electrónica, Universidad Tecnológica Nacional, Facutad Regional, Argentina
21. Central South University, Changsha, China
Early Adopter Status Awarded - Spring 2012

1. Swarthmore College
2. CNMT Department, University of Wisconsin-Stevens Point
3. Texas State University
4. Miami University
5. Oklahoma City University
6. The University of Akron
7. University of North Dakota
8. College of the Holy Cross
9. Humboldt State University
10. Lousiana State University, CSC
11. Saint Anselm College
12. Chicago State University
13. Purdue University
14. Southern University A&M College
15. State University of New York at Stony Brook
16. Singapore University of Technology and Design
17. School of Computer Science and Technology, Huazhong University of Science and Technology
18. National University (NU), Karachi Campus
19. UFRJ + PUC-Rio
20. Missouri S&T
21. Wilberforce University
22. Lamar (Texas State) University
23. Carnegie Mellon University & Pittsburgh Supercomputing Center
24. Louisiana State University, ECE
Courses updated by Early Adopters - Spring 2011

- Loyola University Chicago

- Columbia University
  - Fundamentals of Computer Systems; Principles and Practice of Parallel Programming;

- Wittenberg University and Clemson University
  - CS1, Algorithms, Programming Languages, Computer Organization, CS2, Computational Models and Methods

- University of Georgia
  - Introduction to Computing and Programming, Software Development, Systems Programming, Data Structures

- Calvin College
  - Introduction to Data Structures, Programming Language Concepts, Operating Systems and Networking

- Arizona State University
Courses updated by Early Adopters - Fall 2012

• Swarthmore College
  – CS31 Introduction to computer systems, CS40 Computer graphics, CS41 algorithms, CS45 Operating Systems, and CS87 Parallel computing

• Oklahoma City University
  – CS1, CS2, CS3, Software Engineering

• Singapore University of Technology and Design
  – Introduction to Algorithms - departmental core; Computer System Engineering - departmental core; and Graph Theory and Algorithm - elective.

• Purdue University
  – ECE 264 - Advanced C Programming

• Huazhong University of Science and Technology
  – Parallel Programming Principle and Practice; Parallel Data Structure and Algorithm

• Wilberforce University
  – IDS L, university wide core, and a co-op course.

• Carnegie Mellon University & Pittsburgh Supercomputing Center
  – Introduction to Computational Physics and Advance Computational Physics

• Louisiana State University
  – Digital Logic I, II
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
  • Selected 20 of 50 early adopters to attend

– *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.
How to Read the Proposal
How to Read the Proposal

• Oh no! Not another class to squeeze into our curriculum!
How to Read the Proposal

• Oh yes! Not another class to squeeze into your curriculum!
How to Read the Proposal

• Oh yes! Not another class to squeeze into your curriculum!

• Teaching parallel thinking requires a pervasive but subtle shift in approach
How to Read the Proposal

• Oh yes! Not another class to squeeze into your curriculum!
• Teaching parallel thinking requires a pervasive but subtle shift in approach
• We identified topics that contribute to the shift
How to Read the Proposal

• Oh yes!  Not another class to squeeze into your curriculum!
• Teaching parallel thinking requires a pervasive but subtle shift in approach
• We identified topics that contribute to the shift
  – Descriptions are brief to give you flexibility
How to Read the Proposal

• Oh yes! Not another class to squeeze into your curriculum!
• Teaching parallel thinking requires a pervasive but subtle shift in approach
• We identified topics that contribute to the shift
  – Descriptions are brief to give you flexibility
  – ...but they’re not meant to invoke thoughts of “you can’t teach that at the sophomore level”
How to Read the Proposal

• Oh yes! **Not** another class to squeeze into your curriculum!
• Teaching parallel thinking requires a pervasive but subtle shift in approach
• We identified topics that contribute to the shift
  – Descriptions are brief to give you flexibility
  – ...but they’re not meant to invoke thoughts of “you can’t teach that at the sophomore level”
  – If that’s what you see, you’re missing the point
How to Read the Proposal

• Oh yes! **Not** another class to squeeze into your curriculum!
• Teaching parallel thinking requires a pervasive but subtle shift in approach
• We identified topics that contribute to the shift
• You choose the places they fit in your courses
How to Read the Proposal

• Oh yes! Not another class to squeeze into your curriculum!
• Teaching parallel thinking requires a pervasive but subtle shift in approach
• We identified topics that contribute to the shift
• You choose the places they fit in your courses
  – We offer some suggestions
  – Early adopters are developing examples
<table>
<thead>
<tr>
<th>Algorithms Topics</th>
<th>Bloom#Course</th>
<th>Learning Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithmic problems</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>
</tr>
<tr>
<td>broadcast</td>
<td>C/A</td>
<td>Data Structures/Algorithms represents method of exchanging information - one-to-all broadcast (by recursive doubling)</td>
</tr>
<tr>
<td>multicast</td>
<td>K/C</td>
<td>Data Structures/Algorithms 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>
</tr>
<tr>
<td>gossip</td>
<td>N</td>
<td>Not in core</td>
</tr>
<tr>
<td>Asynchrony</td>
<td>K</td>
<td>CS2, Data Structures/Algorithms asynchrony as exhibited on a distributed platform, existence of race conditions</td>
</tr>
<tr>
<td>Synchronization</td>
<td>K</td>
<td>CS2, Data Structures/Algorithms aware of methods of controlling race condition,</td>
</tr>
<tr>
<td>Sorting</td>
<td>C</td>
<td>CS2, Data Structures/Algorithms parallel merge sort,</td>
</tr>
<tr>
<td>Selection</td>
<td>K</td>
<td>CS2, Data Structures/Algorithms min/max, know that selection can be accomplished by sorting</td>
</tr>
</tbody>
</table>
Rationale for Architecture Topics
Rationale for Architecture Topics

• Multicore parallelism is everywhere
• Internet, Facebook exemplify distributed computing
  • Students are familiar users of PDC
  • They will encounter PDC architecture concepts earlier in core

• **Architecture/Organization Classes**
  • Parallelism of control vs. data
    – Pipeline (K,N), stream e.g., GPU (N/K), vector (N/K) , heterogeneous (K)
    – Multithreading (K), multicore (C), cluster and cloud (K)
  • Memory partitioning – shared vs. distributed memory
    – SMP bus (C), topologies (C), latency (K), bandwidth (K), routing (N), ...
Architecture Topics

• **Memory Hierarchy**
  – issues of atomicity, consistency, and coherence become more significant in PDC context (but easier to address in programming, rather than architecture context)
    • Cache (C), Atomicity (N), Consistency (N), ...

• **Performance Metrics**
  – unique challenges because of asynchrony
  – much harder to approach peak performance of PDC systems than for serial architectures
    • Cycles per instruction (C), Benchmarks (K), Peak performance (C), LinPack (N), ...

• **Floating-point representation**
  – Range (K), Precision (K), Rounding issues (N)
Architecture Topics Philosophy

• There are some PDC topics that are easily explained by appealing to hardware causes
  – Those belong in the context of architecture

• Many topics that could be explained through architectural examples are easier to grasp in other contexts
  – Programming, algorithms, crosscutting ideas
Architecture Topics Philosophy

• There are some PDC topics that are easily explained by appealing to hardware causes
  – Those belong in the context of architecture
• Many topics that could be explained through architectural examples are easier to grasp in other contexts
  – Programming, algorithms, crosscutting ideas
• Just because you can, doesn’t mean you should
Rationale for Programming Topics
Overall Rationale

• Assume some conventional (sequential) programming experience
• Key is to introduce parallel programming early to students
• Four overall areas
  – Paradigms – By target machine model and by control statements
  – Notations – language/library constructs
  – Correctness – concurrency control
  – Performance – for different machine classes
Parallel Programming Paradigms

• By target machine model
  – Shared memory (Bloom classification A)
  – Distributed memory (C)
  – Client/server (C)
  – SIMD (K) – Single Instruction, Multiple Data
  – Hybrid (K) – e.g., CUDA for CPU/GPU

• Program does not have to execute on a target machine with same model
Paradigms (cont.)

• By control statements
  – Task/thread spawning (A)
  – Data parallel (A)
  – SPMD (C) – Single Program Multiple Data
  – Parallel Loop (C)

• All of these can run on shared or distributed memory machines
Parallel Programming Notations

• Overall goal is to know several (at least one per group), have expertise in at least one

• Array languages
  – Vector extensions (K) – SSE
  – Fortran 95, C++ array classes (N)

• Shared memory
  – Compiler directives/pragmas (C)
  – Libraries (C)
  – Language extensions (K)
Notations (cont.)

• SPMD (C)
  – CUDA and OpenCL – for GPUs
  – MPI, Global Arrays, BSP

• Functional/Logic Languages (N)
  – Parallel Haskell
  – Erlang
  – Parlog
Correctness and semantics

• Creating parallel tasks (K)
  – Implicit vs. explicit (K)

• Synchronization (A)
  – Critical regions (A), producer/consumer (A), monitors (K)

• Concurrency defects (C)
  – Deadlocks (C), Race conditions (K)
  – Detection tools (K)

• Memory models (N)
  – Sequential, relaxed consistency (N)
Performance

• Computation
  – Decomposition strategies (C) – owner computes (C), atomic tasks (C), work stealing (N)
  – Scheduling, mapping, load balancing (C) – static, dynamic
  – Elementary program transformations (N) – loop fusion/fission/skewing

• Performance monitoring (K)
  – Tools – gprof, etc.
Performance (cont.)

• Data organization \((\mathbf{K})\)
  – Data distribution \((\mathbf{K})\) – block, cyclic
  – Data locality \((\mathbf{K})\)
  – False sharing \((\mathbf{K})\)

• Metrics \((\mathbf{C})\)
  – Speedup \((\mathbf{C})\), Efficiency \((\mathbf{C})\), Amdahl’s Law \((\mathbf{K})\)
Rationale for Algorithms in the Parallel/Distributed Computing Curriculum
Overview (Decreasing order of abstractness)

– Parallel and Distributed Models and Complexity
– Algorithmic Paradigms
– Algorithmic Problems
Overall Rationale

- The algorithmics of Parallel and Distributed computing is much more than just parallelizing sequential algorithms.
Overall Rationale

• The algorithmics of Parallel and Distributed computing is much more than just parallelizing sequential algorithms.
• Students must be taught to “think in parallel”
Overall Rationale

• The algorithmics of Parallel and Distributed computing is much more than just parallelizing sequential algorithms.
• Students must be taught to “think in parallel”—really to think “parallel-ly”
Overall Rationale

• The algorithmics of Parallel and Distributed computing is much more than just parallelizing sequential algorithms.
• Students must be taught to “think in parallel”—really to think “parallel-ly”

To this end, we must offer the students
• conceptual frameworks adequate to thinking “parallel-ly”

=> the topic, Parallel and Distributed Models and Complexity
Overall Rationale

• The algorithmics of Parallel and Distributed computing is much more than just parallelizing sequential algorithms.

• Students must be taught to “think in parallel”—really to think “parallel-ly”

To this end, we must offer the students

• conceptual frameworks adequate to thinking “parallel-ly”
  => the topic, Parallel and Distributed Models and Complexity

• conceptual tools for crafting parallel algorithms
  => the topic, Algorithmic Paradigms
Overall Rationale

• The algorithmics of Parallel and Distributed computing is much more than just parallelizing sequential algorithms.
• Students must be taught to “think in parallel”—really to think “parallel-ly”

To this end, we must offer the students
• conceptual frameworks adequate to thinking “parallel-ly”
  => the topic, Parallel and Distributed Models and Complexity
• conceptual tools for crafting parallel algorithms
  => the topic, Algorithmic Paradigms
• a range of examples to concretize the abstractions
  => the topic, Algorithmic Problems
The Bloom Classification (A reminder)

K  Know the term
C  Comprehend the term: paraphrase or illustrate
A  Apply the notion (in some appropriate way)
N  Not in the core curriculum
The Bloom Classification  
(A reminder)

K  *Know* the term  
(useful for following technology and for further enrichment)

C  *Comprehend* the term: paraphrase or illustrate  
(understanding necessary for thinking parallel-ly)

A  *Apply* the notion (in some appropriate way)  
(mastery necessary for thinking parallel-ly)

N  *Not* in the core curriculum  
(deferred to advanced courses)
Parallel and Distributed Models and Complexity

K Know the term
C Comprehend the term: paraphrase or illustrate
A Apply the notion (in some appropriate way)
N Not in the core curriculum

Sample Topics

Costs of Computation (C): Time, Space, Power, . . .
Cost reduction (K): Speedup, Space compression, . . .
Scalability (C): (in algorithms and architectures)
Model-Based Notions (K): the PRAM (P-completeness), BSP, CILK
Scheduling Notions (C): Task graphs (dependencies), Makespan
Asymptotic Analysis (C): (Possibly via an Intro to Algorithms class)
Advanced Topics (N): Cellular automata (firing squad synch),
Cost tradeoffs (time vs. space, power vs. time)
Parallel and Distributed Models and Complexity

K  Know the term
C  Comprehend the term: paraphrase or illustrate
A  Apply the notion (in some appropriate way)
N  Not in the core curriculum

Sample Topics

Theme: Benefits and Limits of parallel computing

Costs of Computation (C):  Time, Space, Power, . . .
Cost reduction (K):      Speedup, Space compression, . . .
Scalability (C):         (in algorithms and architectures)
Model-Based Notions (K): the PRAM (P-completeness), BSP, CILK
Scheduling Notions (C):  Task graphs (dependencies), Makespan
Asymptotic Analysis (C): (Possibly via an Intro to Algorithms class)
Advanced Topics (N):    Cellular automata (firing squad synch),
                        Cost tradeoffs  (time vs. space, power vs. time)
# Algorithmic Paradigms

<table>
<thead>
<tr>
<th>K</th>
<th>Know the term</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Comprehend the term: paraphrase or illustrate</td>
</tr>
<tr>
<td>A</td>
<td>Apply the notion (in some appropriate way)</td>
</tr>
<tr>
<td>N</td>
<td>Not in the core curriculum</td>
</tr>
</tbody>
</table>

## Sample Topics

- **Divide & Conquer (A)** (parallel aspects)
- **Recursion (C)** (parallel aspects)
- **Scan (K)** a/k/a parallel-prefix
  - from “low-level” (carry-lookahead adders) to “high-level”
- **Reduction (K)** a/k/a map-reduce
Algorithmic Paradigms

K  Know the term
C  Comprehend the term: paraphrase or illustrate
A  Apply the notion (in some appropriate way)
N  Not in the core curriculum

Sample Topics

Theme: Multi-purpose “tools” — you’ve seen some of these before

Divide & Conquer (A) (parallel aspects)
Recursion (C) (parallel aspects)
Scan (K) a/k/a parallel-prefix
  from “low-level” (carry-lookahead adders) to “high-level”
Reduction (K) a/k/a map-reduce

Advanced Topics (N) Series-parallel composition, Stencil-based iteration,
Dependency-based partitioning,
“Out-of-core” algorithms, Blocking, Striping
Algorithmic Problems

K  Know the term
C  Comprehend the term: paraphrase or illustrate
A  Apply the notion (in some appropriate way)
N  Not in the core curriculum

**Sample Topics**

*Collective communication:* Broadcast (A), Multicast (K), Scatter/Gather (C), Gossip (N)

*Managing ordered data:* Sorting (A), Selection (K)

*Clocking issues:* Asynchrony (K), Synchronization (K)

*Graph algorithms:* Searching (C), Path selection (N)

*Specialized computations:* Convolutions (A), Matrix computations (A)
(matrix product, linear systems, matrix arithmetic)

*Advanced topics (N):* Termination detection,
Leader election/Symmetry breaking
Algorithmic Problems

K Know the term
C Comprehend the term: paraphrase or illustrate
A Apply the notion (in some appropriate way)
N Not in the core curriculum

Sample Topics

Theme: Important specific computations, (some specialized, some familiar)

Collective communication: Broadcast (A), Multicast (K),
Scatter/Gather (C), Gossip (N)

Managing ordered data: Sorting (A), Selection (K)

Clocking issues: Asynchrony (K), Synchronization (K)

Graph algorithms: Searching (C), Path selection (N)

Specialized computations: Convolutions (A), Matrix computations (A)
(matrix product, linear systems, matrix arithmetic)

Advanced topics (N): Termination detection,
Leader election/Symmetry breaking
Rationale for Cross-Cutting Topics
Overall Rationale

• For entering students, concurrency isn’t a paradigm shift (there is no existing paradigm)
• It is a shift for educators/educated
• Concurrency early and broadly establishes it as a natural part of computer science
Rationale for Cross-Cutting Topics

• **High level themes:**
  - *Why and what is parallel/distributed computing (K)?*

• **Concurrency topics**
  - Concurrency, Non-determinism, Power (K),
  - Locality (C)
Hot Topics

• Concurrency has become visible as well as important and pervasive

• Current/Hot/Advanced Topics

  • Cluster, cloud/grid, p2p, fault tolerance (K)
  • Security in Distributed System, Distributed transactions, Web search (K)
  • Social Networking/Context, performance modeling, (N)
Lessons learnt

How can you restructure your own core courses?
## Data Structures and Algorithms (DS/A) course
- Sampled over 4 courses

### Algorithms - Topics

<table>
<thead>
<tr>
<th>Algorithms - Topics</th>
<th>DS/A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals for Topics</td>
<td>K</td>
<td>C</td>
</tr>
<tr>
<td>Asymptoptic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Time</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Space</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Speedup</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cost Reduction:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Space Compression, etc.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Time vs Space</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Power vs Time, etc.</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Scalability in Algorithms and Architectures</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Notions from Complexity-Theory:</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>P-Completeness</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>#P-Completeness</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Cellular Automata</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Dependencies</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Divide and Conquer (Parallel Aspects)

<table>
<thead>
<tr>
<th>DS/A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals for Topics</td>
<td>K</td>
</tr>
<tr>
<td>Divide and Conquer (Parallel Aspects)</td>
<td>0</td>
</tr>
<tr>
<td>Recursion (Parallel Aspects)</td>
<td>0</td>
</tr>
<tr>
<td>Scan (Parallel Prefix)</td>
<td>0</td>
</tr>
<tr>
<td>Reduction</td>
<td>0</td>
</tr>
<tr>
<td>Map-Reduce</td>
<td>0</td>
</tr>
<tr>
<td>Series-Parallel Composition</td>
<td>0</td>
</tr>
<tr>
<td>Asynchrony</td>
<td>0.5</td>
</tr>
<tr>
<td>Sorting</td>
<td>0</td>
</tr>
<tr>
<td>Selection</td>
<td>0</td>
</tr>
<tr>
<td>Graph Algorithms</td>
<td>0</td>
</tr>
<tr>
<td>Search</td>
<td>0</td>
</tr>
<tr>
<td>Path Selection</td>
<td>0</td>
</tr>
<tr>
<td>Specialized Computations</td>
<td>0</td>
</tr>
<tr>
<td>Matrix Computations</td>
<td>0</td>
</tr>
<tr>
<td>Matrix Product</td>
<td>0</td>
</tr>
</tbody>
</table>

### Hour totals per Course

<table>
<thead>
<tr>
<th>DS/A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals for Topics</td>
<td>K</td>
</tr>
<tr>
<td>Hour totals per Course</td>
<td>2</td>
</tr>
</tbody>
</table>
Data Structures and Algorithms (DS/A) course

<table>
<thead>
<tr>
<th>Crosscutting - Topics</th>
<th>DS/A</th>
<th>Totals for Topics</th>
<th>K</th>
<th>C</th>
<th>A</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------</td>
</tr>
<tr>
<td>Why and What is</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>Parallel/Dist</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
<td>0.5</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>Computiting</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>Concurrency</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0.7</td>
<td></td>
<td>0.7</td>
</tr>
<tr>
<td>Social Network/Context</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
</tbody>
</table>

Hour totals per Course: 0.25 2 0 0.7

<table>
<thead>
<tr>
<th>Programming - Topics</th>
<th>DS/A</th>
<th>Totals for Topics</th>
<th>K</th>
<th>C</th>
<th>A</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------</td>
</tr>
<tr>
<td>Shared Memory</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0.2</td>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td>Distributed Memory</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>SPMD</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>Data Parallel</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>CUDA/OpenCL</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>Computation</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1.3</td>
<td></td>
<td>1.3</td>
</tr>
<tr>
<td>Decomposition Strategies</td>
<td></td>
<td></td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------</td>
</tr>
<tr>
<td>Decomposition into</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>atomic tasks</td>
<td></td>
<td></td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------</td>
</tr>
<tr>
<td>Static</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>Dynamic</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>Data</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1.3</td>
<td></td>
<td>1.3</td>
</tr>
<tr>
<td>Performance Monitoring</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>Speedup</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>Efficiency</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td></td>
<td>0.3</td>
</tr>
</tbody>
</table>

Hour totals per Course: 0.4 2 2.5 1.6
CS2: Second Programming course  
- sampled over 3 courses

<table>
<thead>
<tr>
<th>Crosscutting - Topics</th>
<th>CS2 Totals for Topics</th>
<th>Hour totals per Course</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K</td>
<td>C</td>
</tr>
<tr>
<td>Why and What is Parallel/Dist Computing</td>
<td>1.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Concurrency</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Non-Determinism</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Locality</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Distributed Transactions</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Web Search</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Social Network/Context</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Collaborative Computing</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hour totals</td>
<td>0.8</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithms - Topics</th>
<th>CS2 Totals for Topics</th>
<th>Hour totals per Course</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K</td>
<td>C</td>
</tr>
<tr>
<td>Asymptoptic</td>
<td>0</td>
<td>1.5</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Space</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Speedup</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Time vs Space</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Divide and Conquer (Parallel Aspects)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Recursion (Parallel Aspects)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Blocking</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Striping</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Series-Parallel Composition</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Sorting</strong></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Selection</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Specialized Computations</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Matrix Computations</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Matrix Product</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Matrix Arithmetic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Matrix Transpose</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hour totals</td>
<td>1.7</td>
<td>1.3</td>
</tr>
</tbody>
</table>
## CS2: Second Programming course

### Architecture - Topics

<table>
<thead>
<tr>
<th>Topic</th>
<th>Total for Topics</th>
<th>CS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicore</td>
<td>0.5 0 0 0.2</td>
<td></td>
</tr>
<tr>
<td>Cluster</td>
<td>0 0.5 0 0.2</td>
<td></td>
</tr>
<tr>
<td>Heterogeneous (ie: cell)</td>
<td>0 0.5 0 0.2</td>
<td></td>
</tr>
<tr>
<td>Coherence</td>
<td>0 0.5 0 0.2</td>
<td></td>
</tr>
<tr>
<td>Range</td>
<td>0.5 0 0 0.2</td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.5 1 0 0.5</td>
<td></td>
</tr>
<tr>
<td>Rounding Issues</td>
<td>0 1 0 0.3</td>
<td></td>
</tr>
<tr>
<td>Error Propogation</td>
<td>0 0 1 0.3</td>
<td></td>
</tr>
<tr>
<td>754 Standard</td>
<td>0.5 0 0.5 0.3</td>
<td></td>
</tr>
<tr>
<td>Sustained Performance</td>
<td>0.5 0 0 0.2</td>
<td></td>
</tr>
</tbody>
</table>

### Hour totals per Course

- K: 0.8
- C: 1.2
- A: 0.5
- Average: 0.9

### Hour totals per Course

- K: 5
- C: 0.8
- A: 6
- Average: 4
### Algorithms - Topics

<table>
<thead>
<tr>
<th>Topic</th>
<th>CS1</th>
<th>K</th>
<th>C</th>
<th>A</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymptotic</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Time</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Map-Reduce</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Stencil-based Iteration</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Series-Parallel Composition</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Sorting</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>Selection</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td><strong>Hour totals per Course</strong></td>
<td>2.5</td>
<td>2</td>
<td>0</td>
<td>1.6</td>
<td></td>
</tr>
</tbody>
</table>

### Crosscutting - Topics

<table>
<thead>
<tr>
<th>Topic</th>
<th>CS1</th>
<th>K</th>
<th>C</th>
<th>A</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why and What is Parallel/Dist Computing</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Locality</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Web Search</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>Collaborative Computing</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td><strong>Hour totals per Course</strong></td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

### Architecture - Topics

<table>
<thead>
<tr>
<th>Topic</th>
<th>CS1</th>
<th>K</th>
<th>C</th>
<th>A</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highly Multithreaded</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Multicore</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Cluster</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Heterogeneous</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>SMP</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Range</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Precision</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>Rounding Issues</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>Error Propogation</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td><strong>Hour totals per Course</strong></td>
<td>4</td>
<td>1.5</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
A sample Data Structure and Algorithms Syllabus

(Second semester with only programming as their prior background, IIIT Hyderabad, India)

1. Introduction to data structures (Week 1)
2. Introduction to asymptotic analysis – mainly O(.) notation (Week 2)
3. Array as a data structure, sorting, parallel sorting, parallel prefix (Week 3)
4. Stacks and queues (Week 4)
5. Linked lists, ideas from list representation and ranking and its difficulty in the parallel setting. (Week 5)
6. Trees, applications to evaluation, searching, balanced search trees, scope for parallel operations (Week 6-7)
7. Graph traversal techniques, shortest paths, spanning trees, solutions in the parallel setting (Week 8-10)
8. Advanced data structures such as Union-Find, B-trees, Suffix tree, trie (Week 11-13)
Suggestions for How to Teach Selected Topics
Algorithms – Models/Complexity

• **Speedup:** Introduce the notion of speedup and give formal definition.
  – E.g., adding n numbers in \(O(\log n)\) time in parallel with \(p = n/2\).

• **PRAM:** Introduce PRAM model, highlighting unrealistic assumptions of \(O(1)\) time shared memory access (SIMD architecture); Introduce EREW, CREW, and CRCW
  – Illustrate its functioning and capability with Boolean operations over \(n\) bits (OR, AND,...)
  – \(O(1)\) time with short circuit evaluation CRCW model vs. \(O(\log n)\) using a reduction tree on an EREW; Highlight model empowers one to explore how much concurrency

• **Notions from scheduling:** Take a simple problem such as max or sum of an array of \(n\) integers
  – partitioned into smaller tasks (over subarrays), solved, and then combined
  – using a task graph of a reduction tree or of a centralized, hub-spoke tree
  – Use this to illustrate the task graph and the dependencies among parent and child tasks.

• **Dependencies:** Illustrate data dependencies
  – handshake between the producer task and consumer task.

• **Task graphs:** Show how to draw these task graphs modeling dependencies.

• **Work:** Calculate work for given task graph using big-O notation.
Algorithmic Paradigms

- **Divide & conquer (parallel aspects):** mergesort, quicksort

- **Series-parallel composition:** natural way to solve a problem needing multiple phase/sub-algorithm due to data dependency
  - Illustrate it using O(n)-time odd-even transposition sort,
  - Or, O(1)-time max-finding on a CRCW PRAM

- **Sorting:**
  - (i) Explain the parallelization of mergesort wherein each level starting from bottom to top can be merged in parallel using n/2 processors thus requiring $O(2 + 4 + \ldots + n/4 + n/2 + n) = O(n)$ time.
  - Using $p \leq n/2$ processors will lead to $O(n/p\log(n/p) + n)$ time, hence $p = \log n$ is a cost-optimal choice.
  - (ii) Highlight that a barrier (or a lock/Boolean flag per internal node of the recursion tree) on shared memory machine or messages from children processors to parent processors in a local memory machine would be needed to enforce data dependency;
  - (iii) Mention that faster merging of two n/2 size subarray is possible, e.g., in $O(\log n)$ time on a CREW PRAM using simultaneous binary search using n processor, thus yielding $O(\log^2 n)$-time algorithm.
Specialized computations

Example problem - matrix multiplication (AXB = C, nxn square matrices):

– Explain the \( n^3 \)-processor \( O(\log n) \)-time PRAM CREW algorithm
  • yields cost optimality by reducing processors \( p \) in \( O(n^3 / \log n) \) ensuring \( O(n^3 / p) \) time (exercise?).
– A practical shared-memory, statically mapped (cyclic or block) algorithm can be derived
  • for \( p \leq n^2 \) by computing \( n^2 / p \) entries of product matrix \( C \) in a data independent manner;
– For \( p \leq n \), the scheduling simplifies to mapping rows or columns of \( C \) to processors;
  • memory contention can be reduced by starting calculation at the \( i \)th column in the \( i \)th row (exercise?).
– For a local memory machine with \( n \) processors with a cyclic connection, the last approach yields a simple algorithm by distributing the \( i \)th row of \( A \) and the \( i \)th column of \( B \) to \( P_i \), and rotating \( B \)'s columns (row-column algorithm)
  • yields \( O(n^2) \) computation and communication time;
  • for \( p < n \), row and column bands of \( A \) and \( B \) can be employed - derive \( O(n^3 / p) \) time (exercise?).
– For 2-D mesh, explain cannon's algorithm (may explain as refinement of \( n^2 \)-processor shared-memory algorithm, wherein each element is a block matrix).
Cross-Cutting Topics – How to teach

• **Non-determinism:** e.g., floating point arithmetic in parallel can lead to different results;
  – example of web search on various sites can lead to better results
  – parallel branch and bound usually finds better solution because of non-determinacy.
• **Power:** Discuss importance of energy efficiency from multiple perspectives,
  – e.g., unsustainable power/thermal densities, long lifetime needs of embedded & mobile devices, energy costs, and sustainability considerations.
• **Locality:** Its manifestation in cache vs main memory, memory vs. hard drive, local memory vs. non-local memory, 1-hop neighbor vs. distant neighbor
  – dense matrix multiplication with three nested loops - Organizing the loops differently typically results in a wide fluctuation in performance.
• **Cloud/Grid:** Example: Google Doc - collaborative computing (editing), service in the cloud, and seti@home for volunteer grid computing
• **P2P:** file/music-sharing systems, multi-party games using Bluetooth devices (Nintendo DS/2). Comparison between client-server and P2P approaches to problem solving.
• **Distributed transactions:** ATM transactions (strong consistency); Social Networking sites (weak consistency): giant distributed databases;
• **Web search:** e.g. web crawling and its cluster architecture
Architecture - How to teach

• **SIMD/Vector (e.g., SSE, Cray):**
  – Any place that vector/matrix arithmetic algorithms are covered, and even in a data structures class.
  – If chip photos are shown, the control unit section of a processor could be shared among multiple identical ALUs to create an explicitly SIMD architecture.

• **Pipelines:** a chain of consumer/producer threads

• **MIMD:**
  – in an architecture course it is easy to take the step from uniprocessors to multiprocessors, since it is obvious that a CPU can be replicated and used in parallel.
  – In an early programming course, discuss the popular term multicore, and how it could impact their programming.

• **Multicore:**
  – In an architecture course, the limited ability of chips to consume power and dissipate heat motivates the shift away from the tend of using more real estate to build faster processors toward building more cores that operate at a slower rate.
  – Once a chip has multiple cores, the question of how they work together leads to coverage of communication mechanisms.
Programming – how to teach

Parallel Programming paradigms:

• **Shared memory**: Examples of thread programs with both array and control parallelism, with locks and synchronization ops,
  - programming projects w/threads (Java, pthreads, etc.)
• **Distributed memory**: Example of message passing programs, only share data via messages
• *Client Server*: Java RMI or sockets or web services example, notion of invoking a service in another (server) process, and that client and server may run concurrently
• **Hybrid**: Idea of a single parallel program, with each process maybe running on different hardware (CPU, GPU, other co-processor), and that can be client/server, or MIMD program,

Parallel Programming notations:

• **Shared memory notations**: Introduce various ways of parallel programming:
  - Parallel languages, Cilk, X10, Erlang.
  - compiler directives or pragmas, such as OpenMP.
  - Parallel libraries, such as Pthreads, Pfunc, TBB,
  - Frameworks such as CUDA, OpenCL, etc., which may incorporate elements of all three.
  - If possible, students should write simple parallel programs to implement the same algorithm using as many of the above four notations as time and resources permit.
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**
Conclusion

• Time is right for PDC curriculum standards
• Core Curriculum Revision is a community effort
  – Curriculum Initiative Website:
    – Linked through TCPP site: tcpp.computer.org

• Email sprasad@gsu.edu
• Need to inculcate “parallel thinking” to all
Acknowledgements

- NSF: Primary Sponsor
- Intel: Early Adopters
- IBM: EduPar Workshop
- NVIDIA: Early Adopters
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 in Parallel and Distributed Computing: NSF/IEEE-TCPP Guidelines for an Undergraduate Core Curriculum

Sushil K Prasad
Georgia State University
Former Chair, IEEE Technical Committee on Parallel Processing (TCPP)