NSF/TCPP Curriculum Standards Initiative in Parallel and Distributed Computing – Core Topics for Undergraduates

Sushil K. Prasad, Georgia State University
Alan Sussman, University of Maryland
Viktor Prasanna, USC
Manish Parashar, Rutgers
Jie Wu, Temple University

Curriculum Initiative Website:
http://www.cs.gsu.edu/~tcpp/curriculum/index.php

Session Outline

- **Introduction & Overview – S. Prasad** (5 mins)
  - **Why this initiative?**
  - Curriculum Released: Preliminary: Dec-10, Version I: May 2012
  - **Process and Bloom's Classification**
- **Rationale for various topics**
  - **Architectures – V. Prasanna** (5 mins)
  - **Programming – A. Sussman** (5 mins)
  - **Algorithms - S. Prasad** (5 mins)
  - **Cross-Cutting Topics – J. Wu** (4 mins)
- **Call for Early Adopters – Fall 2012** (1 min)
  - **Seed funds from NSF**
- **Q&A - 20 minutes**

Who are we?

- Chtchelkanova, Almadena - NSF
- Dehne, Frank - University of Carleton, Canada
- Gouda, Mohamed - University of Texas, Austin, NSF
- Gupta, Anshul - lBM T.J. Watson Research Center
- JaJa, Joseph - University of Maryland
- Kant, Krishna - NSF, Intel
- La Salle, Anita - NSF
- LeBlanc, Richard, University of Seattle
- Lumsdaine, Andrew - Indiana University
- Padua, David- University of Illinois at Urbana-Champaign
- Parashar, Manish- Rutgers
- Prasad, Sushil- Georgia State University
- Prasanna, Viktor- University of Southern California
- Robert, Yves- INRIA, France
- Rosenberg, Arnold- Northeastern and Colorado State University
- Sahni, Sartaj- University of Florida
- Shirazi, Behrooz- Washington State University
- Sussman, Alan - University of Maryland
- Weems, Chip, University of Massachussets
- Wu, Jie - Temple University

Why now?

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

Stakeholders

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

Current State of Practice

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

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

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

Weekly Meetings on Core Curriculum (May-Dec'10; Aug'11-Feb'12)

Goal: Propose core curriculum for CS/CS graduates

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

Process: For each topic and subtopic

1. Assign **Bloom's classification**

   K= Know the term (basic literacy)
   C = Comprehend so as to paraphrase/illustrate
   A = Apply it in some way (requires operational command)

2. Write **learning outcomes**
3. Identify core CS/CE courses impacted
4. Assign number of hours
5. Write suggestions for "how to teach"

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! <u>Not</u> 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! <u>Not</u> 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

K: know term
C: paraphrase/illustrate
A: apply

Example

Algorithms Topics	Bloom#	Course	Learning Outcome
Algorithmic problems			*The important thing here is to emphasize the parallel/distributed aspects of the topic*
Communication			
broadcast	**C/A**	Data Struc/Algo	*represents method of exchanging information - one-to-all broadcast (by recursive doubling)*
multicast	**K/C**	Data Struc/Algo	*Illustrate macro-communications on rings, 2D-grids and trees*
scatter/gather	**C/A**	*Data Structures/Algorithms*	
gossip	**N**	*Not in core*	
Asynchrony	**K**	CS2	*asynchrony as exhibited on a distributed platform, existence of race conditions*
Synchronization	**K**	CS2, Data Struc/Algo	*aware of methods of controlling race condition,*
Sorting	**C**	CS2, Data Struc/Algo	*parallel merge sort,*
Selection	**K**	CS2, Data Struc/Algo	*min/max, know that selection can be accomplished by sorting*

Rationale for Architecture Topics

Viktor Prasanna
University of Southern California

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

Parallel Programming Topics

Alan Sussman
University of Maryland

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**) – *S*ingle *I*nstruction, *M*ultiple *D*ata
  - 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**) – *S*ingle *P*rogram *M*ultiple *D*ata
  - 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 (**K**)
  - Data distribution (**K**) – block, cyclic
  - Data locality (**K**)
  - False sharing (**K**)
- Metrics (**C**)
  - Speedup (**C**), Efficiency (**C**), Amdahl’s Law (**K**)

Algorithms in the Parallel/ Distributed Computing Curriculum

Sushil Prasad

Georgia State University

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

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

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 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

Cross-Cutting Topics

Jie Wu
Temple University

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)

Early Adopter Program

Sushil Prasad

How to obtain Early Adopter Status?

- Spring-11: 16 institutions ; Fall'11: 18; Spring-12: 21
- **Fall-12 round of competition:** Deadline June 30, 2012
  - 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** such as Algorithms, Architecture, Programming Languages, Software Engg., etc.
  - introductory/advanced **PDC course**
  - dept chairs, dept curriculum committee members responsible

Conclusion

- Time is right for PDC curriculum standards
- Core Curriculum Revision is a community effort
  - **Curriculum Initiative Website:**
  - http://www.cs.gsu.edu/~tcpp/curriculum/index.php
  - Linked through TCPP site: tcpp.computer.org

- Feedback: *Email sprasad@gsu.edu*
- *Need to inculcate "parallel thinking" to all*

–

Acknowledgements

- US NSF: Primary Sponsor (CNS/CISE/OCI)

- Intel: Early Adopters

- IBM: EduPar Workshop

- NVIDIA: Early Adopters

Q&A

NSF/TCPP Curriculum Standards Initiative in Parallel and Distributed Computing – Core Topics for Undergraduates

Sushil K. Prasad, Georgia State University
Alan Sussman, University of Maryland
Viktor Prasanna, USC
Manish Parashar, Rutgers
Jie Wu, Temple University

Contact: sprasad@gsu.edu
Website: http://tcpp.computer.org