IEEE-TCPP Parallel and Distributed Computing Curriculum for Computer Science and Engineering Undergraduates - Version 2 beta

Sushil K Prasad, Sheikh Ghafoor
Alan Sussman, Charles Weems, and R. Vaidyanathan

CDER Center

Contact: sushil.prasad@utsa.edu

EduPar-21, May 17, 2021

TCPP Curriculum Initiative:
https://tcpp.cs.gsu.edu/curriculum/

Outline

- IEEE-TCPP Curriculum – 3 mins
  - History, Opportunities
  - Key Activities and Milestones
- Curriculum Version 2-beta – 12 mins
  - Revision Aspects: Big Data, Energy, Distributed Computing
  - Pervasive topics
  - Areas:
    - Programming
    - Architecture
    - Algorithms

Early Adopter and Training Programs

- Over 100 institutions worldwide
  - Spring-11: 16 institutions ; Fall'11: 18;
  - Spring-12: 21; Fall-12: 25 institutions, Fall-13: 25 institutions, Fall-14: 25, Fall-15: 13
  - Most from US (4 year to research institutions, one high school)
  - Some from South America, a few from Europe, fewer from Asia (India, China, Indonesia, Singapore), Middle East
- NSF CyberTraining Training Workshops - Summer 2018-21
  - **UMass/Maryland; Tennessee Tech**
  - NSF/Intel funded stipend up to $5K/proposal
  - *Instructor training + adoption plans*

CDER Courseware Website

Upload and Search Course Material

- **Type:**
  - Slides, Syllabus, Tutorial, Video
  - Animation, Article, Award, Blog, Book, Competition
  - Course Template, Course Module, Data
  - Hardware Access, Software/Tools
  - Proposal, Report
- **Courses:**
  - CS1, CS2, Systems, Data Structures and Algorithms, …

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

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

ARCHITECTURE

PROGRAMMING

CROSS-CUTTING

- open - Work in Progress

Additional CDER Resources

- CDER Book series:
  - Vol 1: Topics in Parallel and Distributed Computing
    - Introducing Concurrency in Undergraduate Courses, *Morgan Kaufman*
  - Vol 2: Topics in Parallel and Distributed Computing
    - Enhancing the Undergraduate Curriculum: Performance, Concurrency, and Programming on Modern Platforms, *Springer*
  - Free Pre-Print Version on CDER site (44K downloads)
  - Plan for 3rd Volume – Experience of Adopters
    - Exemplars + Resources on courses and topics
- CDER Heterogenous Cluster
  - Multi-core, GPU, Shared/Distributed Memory, Hadoop/Spark
  - Ask for class accounts
- Training workshops – NSF/Intel funded

Curriculum Version II Activities

	Areas	Architecture	Algorithms	Programming
New Aspects	**Area Lead/** Aspect Lead	Chip Weems	Anshul Gupta	Alan Sussman
Exemplars	Sushil Prasad	Karen Karavanic, Eric Freudenthal	Erik Saule, Duane Merril, David Bunde	David Brown, Eric Freudenthal
Distributed	Vaidyanathan Ramachandran	Vaidyanathan Ramachandran, Manish Parashar	Vaidyanathan Ramachandran, Costas Busch, Denis Trystram	Alan Sussman, Chi Shen
Big Data	Trilce Estrada	Craig Stunkel	Cynthia Phillips,	Debzani Deb
Energy	Krishna Kant, Craig Stunkel	Craig Stunkel, Karen Karavanic	Denis Trystram	John Dougherty
Pervasive	Sheikh Ghafoor	Craig Stunkel, Eric Freudenthal	Robert Robey, Martina Barnas	Sheikh Gafoor, Eric Freudenthal

- **Timeline:**
  - **Version-2-beta released @ EduHPC'20**
    - **Public Feedback: sushil.prasad@utsa.edu**
  - **Companion Activities:**
    - Exemplars
    - CE-oriented TCPP Curriculum
    - Competencies-based - knowledge, skills and attitudes
- New: NSF Institute Planning Grant => 5 planning workshops
  1. SC'19
  2. SIGCSE'20 - online
  3. July 27, 2020 – online
  4. Mar 26-27, 2021 - online
  5. NSF Report Workshop – Sept'21

Pervasive and Emerging Topics

Sheikh Ghafoor, Tennessee Tech

R. Vaidyanathan, LSU

Major Update

- Crosscutting view of the 2012 curriculum is modified (no crosscutting table in the 2020 curriculum)
  - Pervasive Concepts
    - Fundamental in nature
    - Likely to remain unchanged for long time
  - Emerging Topics
    - Significant current or emerging interest
    - Continually evolving as topics mature and newer topics with time
    - Often Better suited for advanced courses but may be introduced in lower level courses in a limited way.
    - Applications of many of these topics will be familiar to students

Pervasive Concepts

- Broad themes that are conceptual underpinning of PDC
- Appears at different depth throughout the curriculum
- Transcend PDC and often computing
- Should be taught at the core
  - May be through unplugged activities
  - least at “C” level in terms of Bloom classification.
- These should be further reinforced in architecture, programming, and algorithms
- Four topics have been identified
  - Asynchrony
  - Concurrency and Sequential Dependency
  - Locality
  - Performance

Pervasive Concepts

Topics	Learning Outcome and Teaching Suggestion
Asynchrony	1) Understand cause and effect of Asynchrony and how to ensure that computational correctness. 2)Understand asynchrony is the characteristics of modern system and understand asynchrony in PDC context. 3)Can utilize a standard coordination strategy to prevent incorrect operation due to uncontrolled concurrency (race conditions).
Concurrency & Dependency	1) Understand concurrency is an algorithmic property and difference between concurrency and parallelism. 2) Understand sequential dependency limit degree of concurrency and hence parallelism. 3)Identify sequential dependency in algorithms.

Emerging Topics

Topics	Core Bloom Level	Learning Outcome and Teaching Suggestion	Where Covered
Machine Learning	N	Understand that accelerators, parallel matrix decompositions, parallel matrix multiplication, and vectorized operations are fundamental aspects that enable modern machine learning	ParProg, Machine Learning(K)
Mobile Adhoc Network	N	Understand that mobility admits a dynamic topology and a different relationship between "sender" and "receiver." Explore the impact of these differences in the system.	Networking (c)
Quantum Computing	N	Know that quantum computing algorithms can solve some problems (e.g., optimization problems) exponentially faster than classical algorithms via parallelism, and that quantum processors to implement some of those algorithms are under construction, using new parallel programming models	Arch2,ParProg(k)

Programming Topics

Alan Sussman

University of Maryland and NSF

Programming Topics - Overview

- Main goal is to introduce parallel programming topics into intro programming, data structures, and systems classes
  - Secondary goal is to target upper-level classes
- High-level themes include:
  - **Paradigms and notations** – SIMD, shared memory, message passing, client/server, big data stack, threads, tasks, data parallel, etc.
  - **Semantics and correctness** – synchronization, concurrency defects, …
  - **Memory models** – sequential consistency, weak consistency, …
  - **Performance and energy** – computation and data decomposition, scheduling/mapping, data layout and locality, tools and metrics
- Most topics at a shallow level (Bloom level C or K) for intro courses, but at a deeper level for upper-level advanced courses (or deferred to upper-level completely, so at N Bloom level for intro courses)

Programming Topics - Updates

- Incorporated new programming topics related to distributed computing (e.g. client/server), big data (e.g., MapReduce), and power/energy
- Eliminated some topics from original guidelines completely, since no longer relevant
- Added small number of other topics missed in original guidelines, or newer ideas (e.g., accelerator programming)
- Revised learning outcomes, Bloom levels for some topics (mainly between C and K)
- Added learning outcomes and Bloom levels for advanced courses

Programming Topics – Input from Reviewers

- Several suggestions to do a better job on pervasive ideas
  - Across programming areas, algorithms, architecture – e.g., scalability
- Better definitions of terms and acronyms
- More limited energy/power topics – keep them high level
- Some suggestions for updating topics, new topics, eliminating topics no longer relevant
  - Informed discussion for revisions

Architecture Topics

Chip Weems
Univ of Massachusetts

Overview

- Limited coverage in core courses
  - Only K or C Bloom levels
  - Most in Systems course (a few topics in CS1, CS2, such as floating point, atomicity as basis for threaded libraries, event handlers as threads in GUIs)
  - Goal is for all students to be aware of hardware aspects of PDC affecting and providing opportunities for algorithmic problem solving
- Most topics in table are for advanced courses
  - Many already normally covered there, at or beyond suggested bloom levels

Topics Overview

- Classes of Parallelism: data (superscalar, SIMD, dataflow), pipelines (OoO, streams), control (multicore, multithread, heterogeneity), shared memory (snooping, directory), distributed memory (topology, latency)
- Underlying mechanisms (caching, atomicity, consistency, coherence, interrupts/events, handshaking, ID, virtualization)
- Floating point representation (in support of HPC)
- Performance metrics (IPC, benchmarks, network/memory bandwidth, peak performance, sustained performance)
- Power (power/energy, larger scale, embedded, density, static/dynamic, DVFS, heterogeneous cores)
- Scaling (Big data, HPC, fault tolerance, data bound computation, volume, velocity, scale out, cost of data movement)

Algorithms

A. Gupta, IBM Research
R. Vaidyanathan, LSU

Algorithms Overview

- Algorithms topics are recommended for coverage along with their sequential counterparts to minimize additional instruction time
- Organized into three sub-areas:
  - ***Parallel/Distributed Models and Complexity:*** Foundational topics aimed at equipping the students with basic knowhow for designing and analyzing parallel algorithms
  - ***Algorithmic Techniques:*** Recurrent themes or constructs that are generally useful in designing a wide variety of parallel algorithms
  - ***Algorithmic Problems:*** Basic problems for which learning both the sequential and parallel algorithms would be considered valuable for almost all CS/CE students

Algorithms: Major update 1

- Model(s) of Choice:

- Understand concurrency basics without the trappings of real systems (routing, data alignment etc.).

- Recognize the PRAM as embodying the simplest forms of parallel computation: Embarrassingly parallel problems can be sped up easily just by employing many processors.

- Recognize how a completely connected network abstracts away from routing details. Recognize the difference between the model(s) and real systems.

- Such a Model of Choice (MoC) is assumed to be chosen and adopted by the instructor on which PDC concepts would be discussed.

Algorithms: Major update 2

- Deeper Algorithmic Experience:

- Experience through class instruction and assignment/project the (1) design, (2) analysis, and (3) implementation aspects of at least one parallel or distributed algorithm of choice in detail.

- Master PDC algorithmic concepts through a detailed exploration, including recognizing how algorithm design reflects the structure of the computational problem(s) and the PDC model/environment.

- Possible computational problems (to be chosen by instructor(s)) include, but are not limited to, matrix product, map reduce, sorting, search, convolution, a graph algorithm of your choice, etc.

- We recommend that all CS/CE undergrads have this experience at some point before graduating.

Q&A

TCPP Curriculum Initiative:
https://tcpp.cs.gsu.edu/curriculum/

Feedback & Participation: sushil.prasad@utsa.edu