CDER Booth Special Sessions

1. Modernizing Early Computing Courses with PDC - Sushil K Prasad, UTSA
2. PDC Concepts into CS1 & CS2 Courses – April Crocket, TTU
3. Peachy Assignments – David Bunde, Knox
4. Fastcode Community for SPE – Bruce Hoppe, MIT
5. Dive Into Systems Book – Tia Newhall, Swarthmore

SIGCSE-25, Feb 27-29, Pittsburgh

https://tcpp.cs.gsu.edu/curriculum/

Modernizing Early Computing Courses with Parallel and Distributed Computing

Sushil K Prasad, UT San Antonio
Alan Sussman, U. Maryland
Chip Weems & Neena Thota, UMass
R. Vaidyanathan, LSU
David Bunde & Jaime Spacco, Knox
Sheikh Ghafoor, April Crockett, & Jerry Gannod, TTU

SIGCSE-25, Feb 27-29, Pittsburgh
https://tcpp.cs.gsu.edu/curriculum/

Public Feedback on TCPP Curriculum & Contact
sushil.prasad@gmail.com

Sponsors

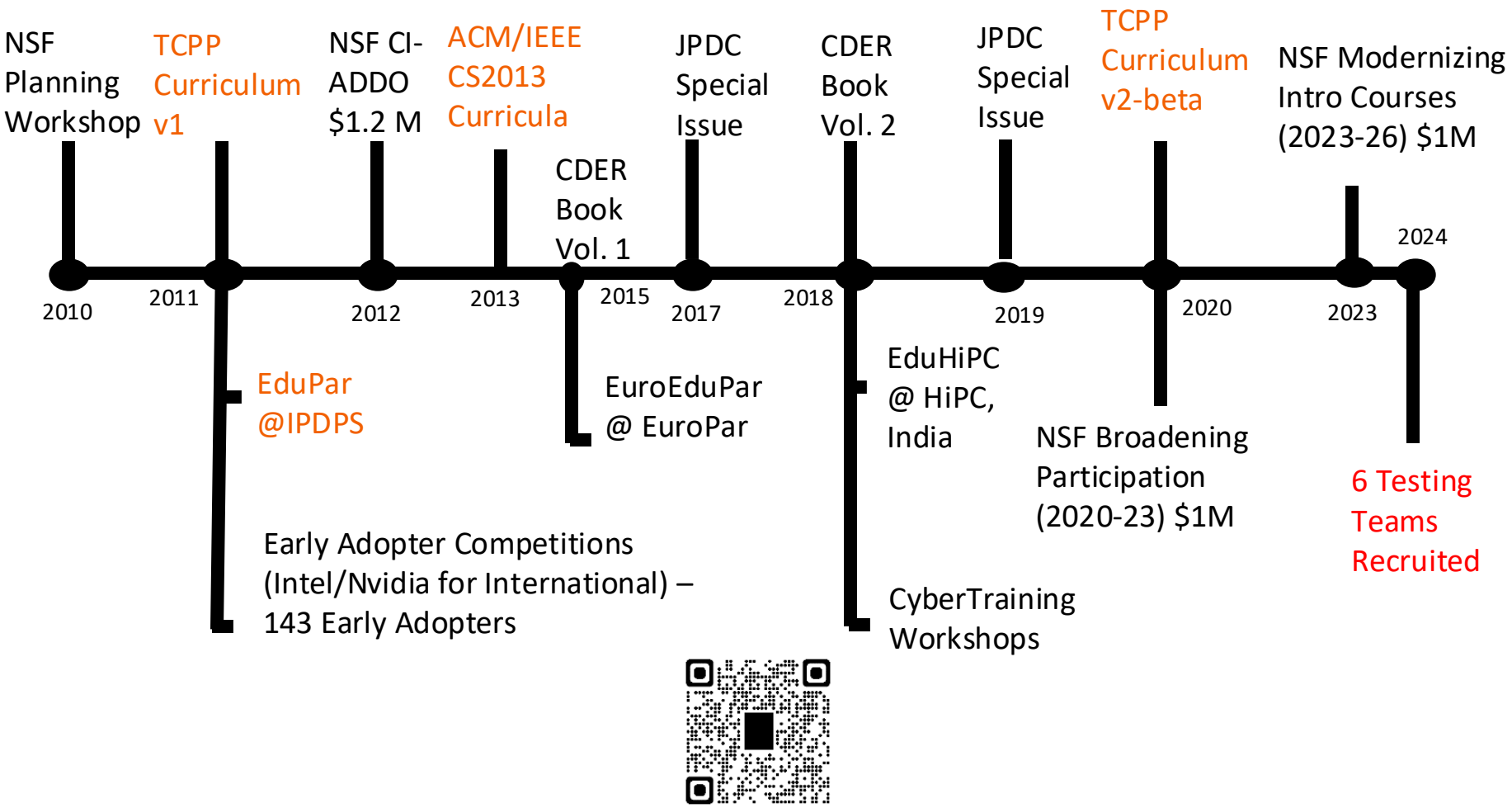
TCPP Curriculum Initiative

What should every Computer Science and Engineering Student know about Parallel and Distributed Computing (PDC)?
https://tcpp.cs.gsu.edu/curriculum/

- **Areas:** Programming, Architecture, & Algorithms
  - Version 1 – 2012
- **New Aspects:** Big Data, Energy, Distributed Computing, Pervasive topics
  - Version-2-beta released 2020

- **Companion Activities:**
  - CE-oriented TCPP Curriculum
  - Modernizing CS1/CS2 Exemplars
    - 2 Development Teams
    - Recruited 6 Testing teams
  - CDER Book Vol 3
    - Experience of Adopters
      - Exemplars + Resources
  - JPDC Special Issues

CDER - Center for Parallel and Distributed Computing Curriculum Development and Education Resources - Timeline



3 Curriculum Areas + Cross-Cutting
Architecture, Programming, Algorithms

TCPP Curriculum Example

Algorithms Topics	Bloom#	Course	Learning outcome and teaching notes
Algorithmic problems			Algorithmic problems section contains parallel algorithms for certain problems. The important thing here is to emphasize the parallel/distributed aspects of the topic
Communication and Synchronization			Understand (at the pseudo-code level) how certain patterns of communication can be implemented in a parallel/distributed model. Also appreciate the cost of communication in PDC.
Reduction and Broadcast for communication and synchronization	**C**	Data Struc/Algo	Understand, for example, how recursive doubling can be used to for all-to-one reduction, and its dual, one-to-all reduction, in log(p) steps. The same applies to all-to-all broadcast and all-to-all reduction. Recognize that all-to-all broadcast/reduction are synchronizing operations in a distributed (event-driven) environment.
Parallel Prefix (Scan)	**C**	Data Struc/Algo	Understand the structure of at least one simple parallel prefix algorithm. One could consider recursive or iterative approaches (such as those of Ladner-Fischer, Kogge-Stone, Brent-Kung)
Multicast	**N**		
Permutation	**N**		

Early Adopter and Training Programs

Over 200 early adopter and trainee institutions worldwide

- Spring-11 - Fall-15
- US, South America, Europe, Asia, and Middle East

NSF CyberTraining PDC Workshops - Summer 2018-25

- UMass; Tennessee Tech; LSU

Additional Training workshops

- **SIGCSE** 2023, 2024, **2025**
- **HiPC** 2022, 2023, **2024**

SIGCSE Tutorial

Modernizing the CS Introductory Sequence with Parallel and Distributed Computing (and some AI)

NSF Stipend: $400/trainee

Fri Feb 28, 2025
7:00 pm - 10:00 pm
at
Meeting Room 310

Some slots available
Register now!

Active, hands-on & unplugged activities

Integrating PDC topics in our courses CS1 & CS2

GenAI in CS1 & CS2

Call for Applications

NSF/CDER Instructor Training Workshop

MODERNIZING THE EARLY CS COURSES WITH PARALLEL AND DISTRIBUTED COMPUTING

July 14 - 18, 2025 hosted by UMass, Amherst

Application Deadline
March 31, 2025

$3000
Stipend/trainee

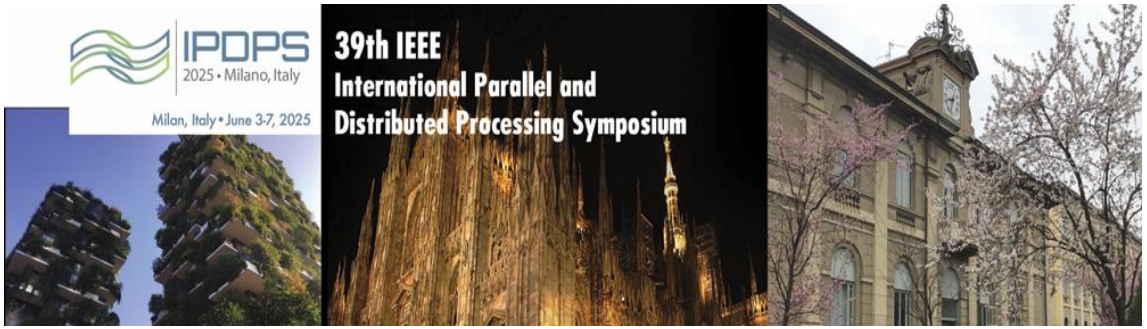
Apply now!

Workshop Agenda:

- PDC Unplugged & Plugged Activities
- Educational Evaluation Methodology
- Previews of modern exemplar first year courses
- Integrating PDC Topics in CS1, CS2, Data Structures and Algorithms, Systems/Comp Org

Edu* Workshop Series

- **EduPar-11** at IPDPS-2011
  - EduPar25 – June, Milan, Italy

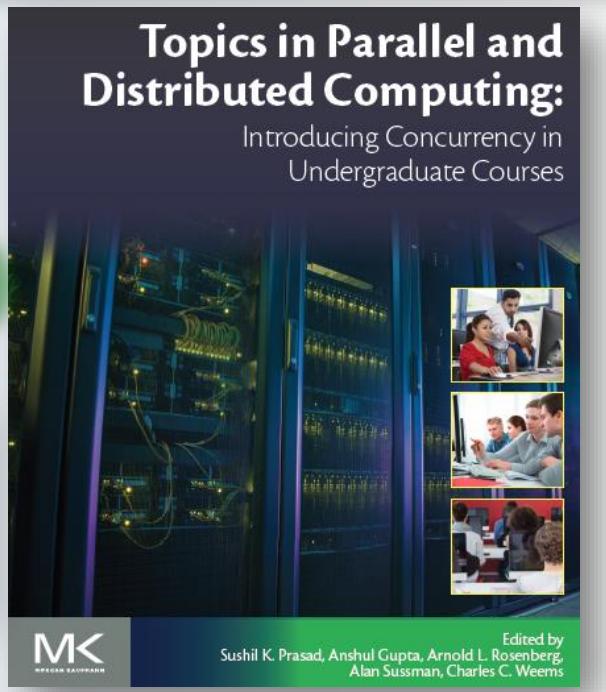


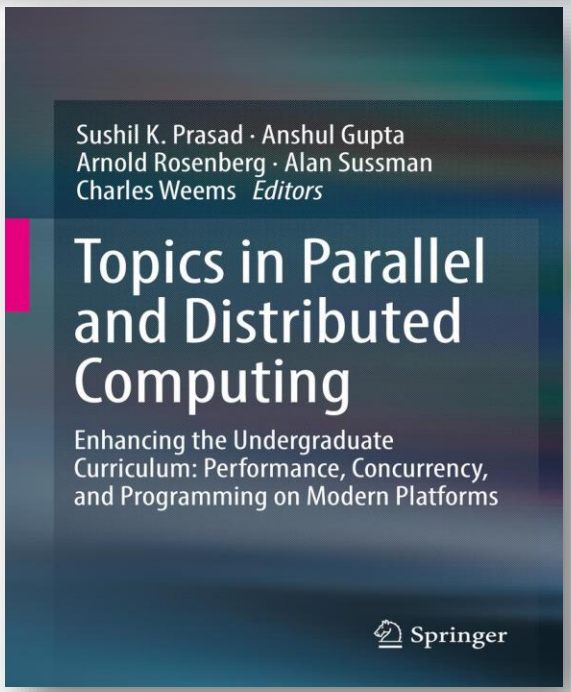
- **EduHPC** at SC-13
  - EduHPC-25 @ SC, St Louis, MO
- **EduHiPC 2018 @ HiPC in** India
  - EduHiPC'25 in Hyderabad, India

CDER Books - Topics in Parallel and Distributed Computing for Undergraduates

50K+ Chapter Downloads

Free Download





Free Download

PART 1 FOR INSTRUCTORS
5 Chapters

- ✓ Hands-on Parallelism with no Prerequisites and Little Time Using Scratch
- ✓ Parallelism in Python for Novices
- ✓ Modules for Introducing Threads
- ✓ Introducing Parallel and Distributed Computing Concepts in Digital Logic
- ✓ Networks and MPI for Cluster Computing

PART 2 FOR STUDENTS
4 Chapters

- ✓ Fork-join Parallelism with a Data-Structures Focus
- ✓ Shared-Memory Concurrency Control with a Data-Structures Focus
- ✓ Parallel Computing in a Python-Based Computer Science Course
- ✓ Parallel Programming Illustrated through Conway's Game of Life

PART 1 FOR INSTRUCTORS
5 Chapters

- ✓ What do we need to know about parallel algorithms and their efficient implementation?
- ✓ Models for Teaching Parallel Performance Concepts
- ✓ Scalability in Parallel Processing
- ✓ Energy Efficiency Issues in Computing Systems
- ✓ Scheduling for fault-tolerance: an introduction

PART 2 FOR INSTRUCTORS
5 Chapters

- ✓ MapReduce - The Scalable Distributed Data Processing Solution
- ✓ The Realm of Graphics Processing Unit (GPU) Computation
- ✓ Managing Concurrency in Mobile User Interfaces with Examples in Android
- ✓ Parallel Programming for Integrative GUI Applications

EduPar'24

- CDER Book Vol 3 - upcoming

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

TCPP Computer Engineering Curriculum

- PDC Concepts
  - Concurrency, Asynchrony and Locality
  - Decomposition and Coordination
  - Performance and Pitfalls
- CE Courses identified
- Broad areas
  - Hardware and Architecture
  - Programming and Algorithms
  - Communications and Systems

Identifying topics in lower-level courses
in which PDC concepts can be incorporated

NSF Modernizing CS1/CS2 Exemplar Project

Vision: Create modern course exemplars for CS1 and CS2 courses to serve as national models	• Systems are more networked, parallel, and graphical than traditional courses • Software uses more libraries and APIs
2 Development teams	• Knox College (Java version) and Tennessee Tech (C/C++ version) • Piloting CS1 this term, CS2 in the Spring
6 Testing Teams	• Casper, Hawaii Pacific U., Montclair, U. Nebraska Lincoln, U. Southern Indiana, and Webster • Will teach the new courses in 2025-26

Introducing HPC Concepts into a CS1 & CS2 Course

Developer Team

- **C++ version** of CS1 & CS2
  Tennessee Tech University - April Crockett
  (acrockett@tntech.edu)

- **Java version** of CS1 & CS2
  Knox College - Jaime Spacco & David Bunde
  (jspacco@knox.edu) & (dbunde@knox.edu)

PDC Focus Areas to Introduce in CS1/CS2

- Introduction to Parallel Programming, Multi-Core Architecture
- Data Parallelism / Parallel Programming
- Distributed Computing / Remote Data Access
- Event Handling

How We are Introducing PDC Topics in CS1 & CS2

- Added Course Objectives
- Lecture slides and interactive questions (using AhaSlides) to introduce topics
- Unplugged activities in lab (showing FlagMaker Unplugged activity sneak peak in a minute)
- Two PDC-related programming assignments in both CS1 & CS2 - one focusing on parallel data and the other on distributed computing

Testing New Content & Assessments

As part of this project, we are testing our course materials & activities as well as having several “testing teams” from different institutions test them in their courses.

TNTECH Testing

- CS1 - Fall 2024
- CS2 - Spring 2025

Testing Teams Testing

- CS1 - Fall 2025
- CS2 - Spring 2026

Unplugged Activity Sneak Peak

The next few slides show the Unplugged Flagmaker Activity, which we use in CS1 during a lab.

The first part of the lab class is this activity and then the second part of the class they work on a parallel programming assignment processing data stored in a vector in parallel.

Unplugged FlagMaker Activity

Note: this is the first slide we show students to introduce this activity. Then the next slide tells them how they will be creating this flag like a computer.

- You'll be pretending to be the computer drawing the flag of **Mauritius** (on the right).
- You'll use **paper** and **markers**, but otherwise follow how a computer might draw it.

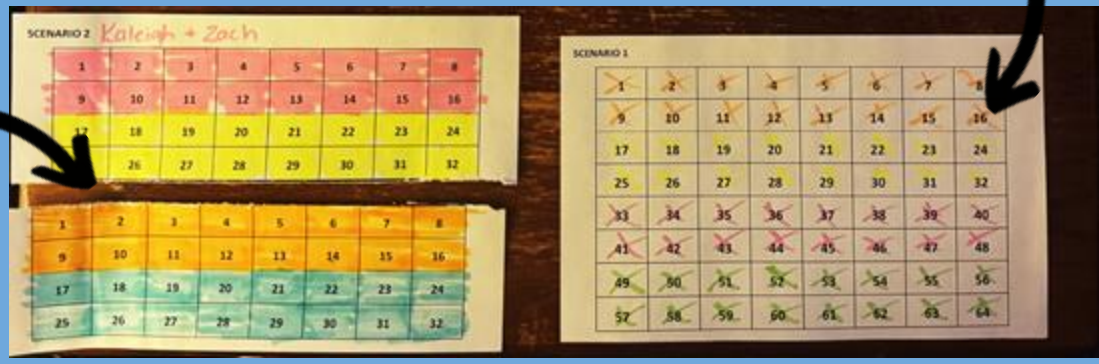
Img from: https://en.wikipedia.org/wiki/Mauritius

HOW TO FILL IN THE FLAG

You must **color one square at a time**. Do not color in multiple squares at one time. You fill in each box in the numerical order shown, like a computer would do with a loop.

CORRECT!

WRONG!
Fill in the squares



WRONG!
Don't tear the paper!

Scenario 1: A Single Processor

Note: after each group finishes each scenario, the instructor writes down each group's time on a whiteboard in the room. Group with the lowest time gets a small prize (makes it more fun and allows students to compare times between the scenarios)

Two people:

1. **The Processor**: colors first stripe (cell by cell), second stripe (cell by cell), third stripe (cell by cell), and then fourth stripe (cell by cell)
2. **Timer**: Times the Processor

P1	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
	17	18	19	20	21	22	23	24
	25	26	27	28	29	30	31	32
	33	34	35	36	37	38	39	40
	41	42	43	44	45	46	47	48
	49	50	51	52	53	54	55	56
	57	58	59	60	61	62	63	64

After the flag is filled out and timer stopped, tell the instructor the time for your group.

Scenario 1: A Single Processor
Let's Talk About It

Note: after each scenario the instructor hands out the prize to the group with the lowest time, and then talks about what the scenario represents in a slide like this.

With only **one processor**, every action is **sequential**, meaning each cell must be processed one at a time without any parallel work.

This highlights the limit of a single-threaded system where tasks cannot be divided to reduce time, demonstrating the concept of processing bottlenecks.

In complex tasks requiring a lot of data handling, using a single processor would result in slow performance as it processes each item individually.

Scenario 2: Two Processors Collaborating

Three people:

1. **Processor 1**: Colors red and blue stripes (cell by cell)
2. **Processor 2**: Colors yellow and green stripes (cell by cell)
3. **Timer**: Times them; stop when **all** cells are colored.

P1	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
	17	18	19	20	21	22	23	24
	25	26	27	28	29	30	31	32
P2	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
	17	18	19	20	21	22	23	24
	25	26	27	28	29	30	31	32

After the timer is stopped, tell the instructor the time for your group.

Scenario 2: Two Processors Collaborating
Let's Talk About It

Adding a second processor allows for **parallel processing**, so two stripes (red and blue, yellow and green) can be worked on **simultaneously**, cutting down completion time.

This demonstrates how **distributing tasks between processors** can increase efficiency. **(We can do things faster!)**

Scenario 3: Four Processors Collaborating

Five people:

1. **Processors 1-4**: Each assigned to one color, which they color (cell by cell)
2. **Timer**: Times them; stop when **all** cells are colored.

P1	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
P2	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
P3	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
P4	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16

After the timer is stopped, tell the instructor the time for your group.

Scenario 3: Four Processors Collaborating
Let's Talk About It

Four processors allow each to work on one color stripe independently, **speeds up the work even more**!

However, with more processors, **synchronization** becomes crucial to ensure they don't interfere with each other's work.

In real systems, **load balancing** is important to make sure all processors have tasks of roughly equal duration.

If one processor finishes early, it sits idle (example of **underutilized resources**).

Scenario 4: Four Processors Collaborating with Column Assignments

Five people:

1. **Processors 1-4**: Each assigned a strip of two columns (first takes left 2 columns, second takes next two columns, etc.) Processors colors these the correct color cell by cell.
2. **Timer**: Times them; stop when **all** cells are colored.

P1		P2		P3		P4	
1	9	1	9	1	9	1	9
2	10	2	10	2	10	2	10
3	11	3	11	3	11	3	11
4	12	4	12	4	12	4	12
5	13	5	13	5	13	5	13
6	14	6	14	6	14	6	14
7	15	7	15	7	15	7	15
8	16	8	16	8	16	8	16

After the timer is stopped, tell the instructor the time for your group.

Scenario 4: Four Processors Collaborating a Different Way (Column Assignments)

Let's Talk About It

Note: we experienced that every group always had a slower time for scenario 4 then scenario 3 because of having to share markers.
Usually, however, scenario 4 was faster than or similar in time to scenario 2.

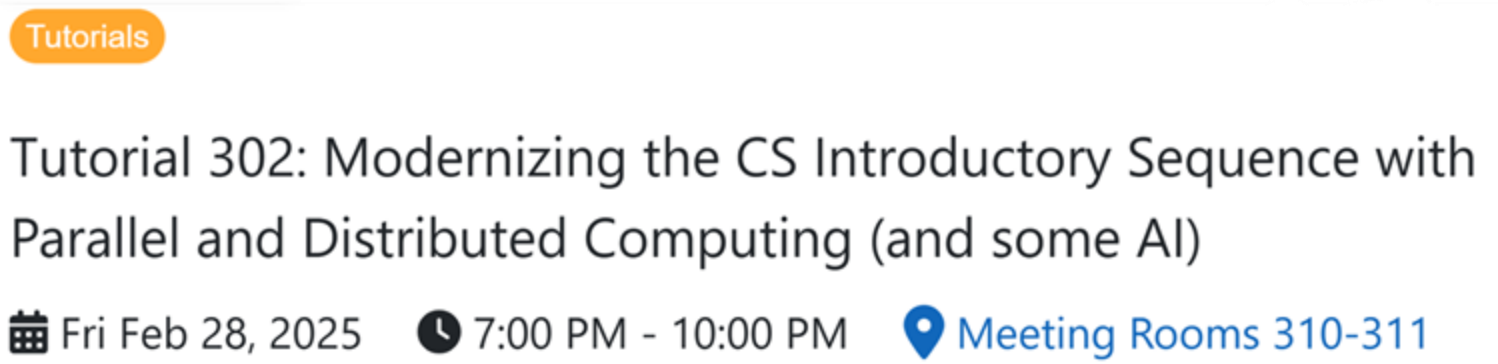
Was it slower or faster?

Processors might again risk stepping into each other's workspaces at the boundaries, which can lead to **race conditions** unless clearly separated.

Want to learn more?

Join our tutorial Friday evening!

We will show how you can fit these topics into your courses, and give you access to course materials that we have developed!



Peachy Assignments: Cool, Adoptable Parallel and Distributed Computing Assignments

David Bunde, Knox College

dbunde@knox.edu

Peachy Assignments: Motivation

Good course assignments

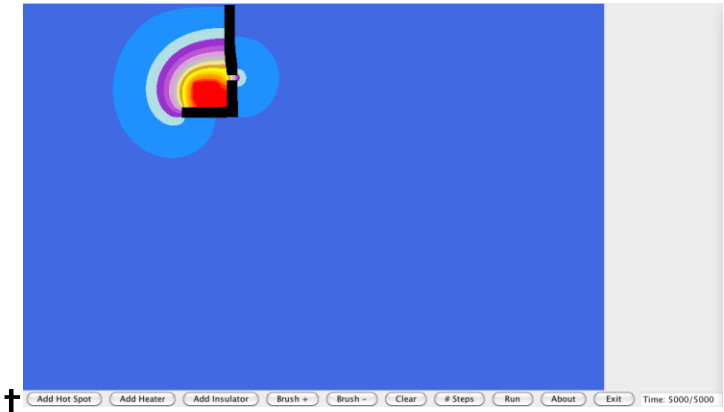
+Inspire students

+Improve student learning

+Are fun to give

- Are lots of work to create

- Don’t always work as well as we thought



Successful assignments should be shared like

"Nifty Assignments" (http://nifty.stanford.edu)

https://tcpp.cs.gsu.edu/curriculum/?q=peachy
(or Google Peachy assignment)

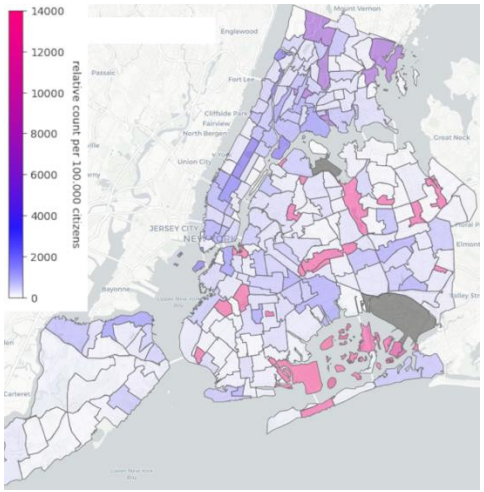
Peachy Assignments

- Selected competitively and presented at
  - EduPar (co-located with IPDPS in May)
  - EduHPC (co-located with SC in November)

- Available on the Peachy website

  https://tcpp.cs.gsu.edu/curriculum/?q=peachy

  (or Google Peachy assignment)



https://tcpp.cs.gsu.edu/curriculum/?q=peachy
(or Google Peachy assignment)

Peachy Assignment Criteria

- Tested
  - Used in the classroom and comes with instructor reflections
- Adoptable
  - Includes handout and given code for students, plus optional solution
  - Description discusses prerequisites and options for adaptation
  - Widely-taught concepts using readily available hardware
- Cool and inspirational
  - Fun and exciting for students
  - Ideally, something they show their roommates

https://tcpp.cs.gsu.edu/curriculum/?q=peachy
(or Google Peachy assignment)

Recent Peachy Assignments

EduHPC 2024

- DNA sequence alignment: An assignment for OpenMP, MPI, and CUDA/OpenCL
- An Introduction to Parallel Quantum Computation and Circuit Cutting through QAOA for Max-Cut

EduPar 2024

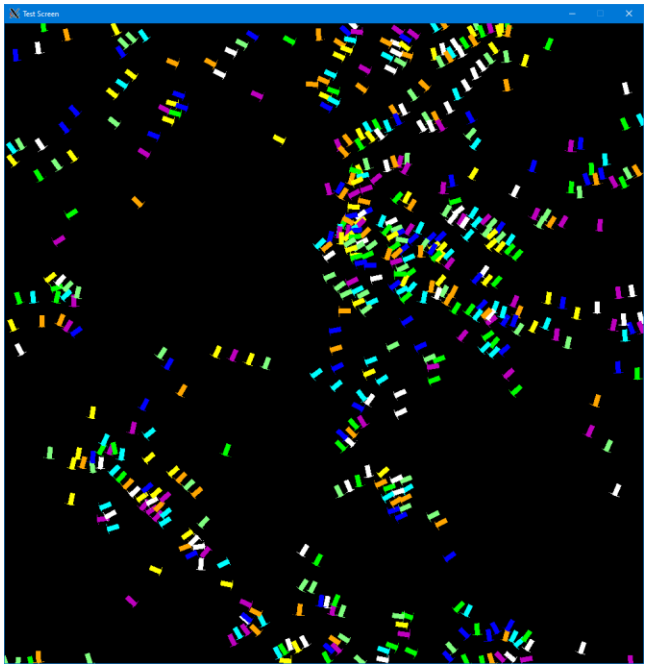
- Fixed-Radius Nearest Neighbor Search using RAPIDS AI
- Visualizing parallelism with a flocking algorithm

https://tcpp.cs.gsu.edu/curriculum/?q=peachy
(or Google Peachy assignment)

Sample Peachy Assignment: Visualizing parallelism with a flocking algorithm

(Elizabeth Shoop and Ethan Scheelk, EduPar 2024)

- "boids" flocking model by Reynolds and Flake
  - Each has random initial position and velocity
  - Updated each time step based on behavior of the others
- Students are given serial code and need to analyze where to apply parallelism using OpenMP
  - Result is visibly faster

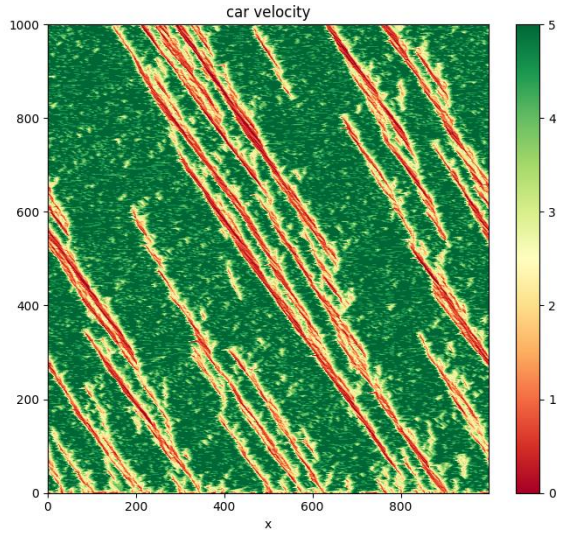


https://tcpp.cs.gsu.edu/curriculum/?q=peachy
(or Google Peachy assignment)

Sample Peachy Assignment: Traffic Jam Modeling

(Ramses von Zon and Marcelo Ponce, EduHPC23)

- Nagel-Schreckenberg traffic model
  - Chain of cars in one lane, each following simple update rules for their speed
- Focus on parallel pseudorandom number generation
- Parallelize C++ using OpenMP (MPI or CUDA are suggested variants)



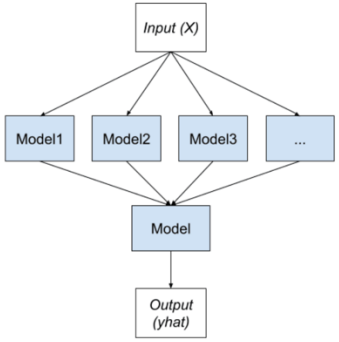
Peachy Assignments

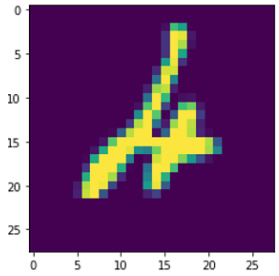
https://tcpp.cs.gsu.edu/curriculum/?q=peachy
(or Google Peachy assignment)

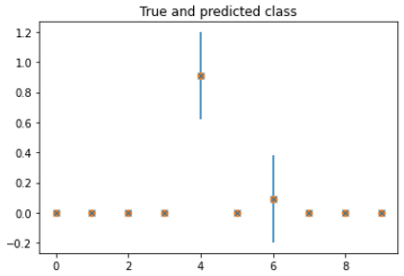
Sample Peachy Assignment: Uncertainty Estimation for Deep Learning

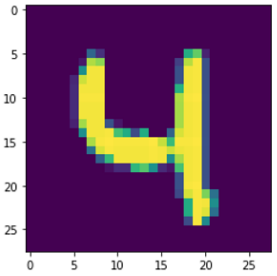
(Pautsch, Li, Rizzi, Thiruvathukal, and Pantoja; EduHPC 23)

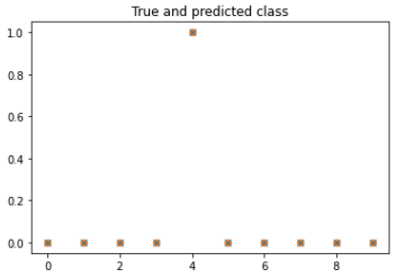
- Using ensemble of neural networks to estimate confidence in output
- Train models independently in parallel using Python











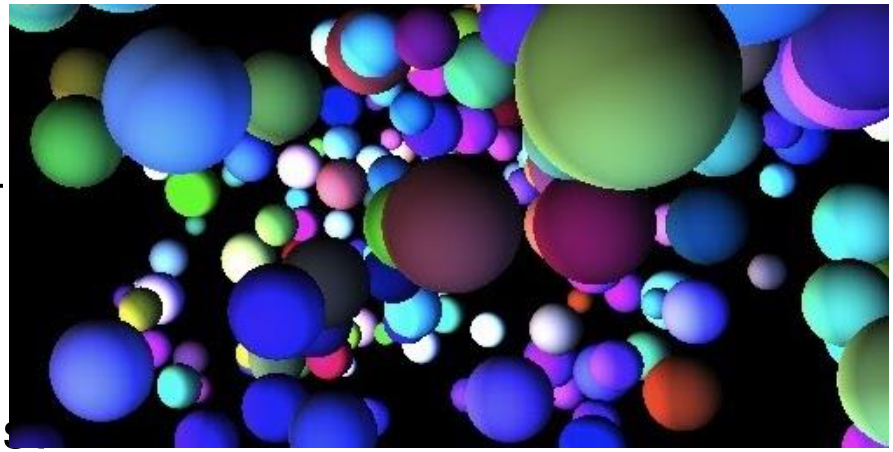
Peachy Assignments

https://tcpp.cs.gsu.edu/curriculum/?q=peachy
(or Google Peachy assignment)

Sample Peachy Assignment: Colliding Spheres

(Iliopoulos, Leiserson, Park, Rosa, and Schadl; EduHPC 22)

- N-body simulation with gravity and elastic collisions
- Parallelize given C++ using Cilk+ (OpenMP is suggested variant)
- Contest to complete as many tes inputs as possible in given time



https://tcpp.cs.gsu.edu/curriculum/?q=peachy
(or Google Peachy assignment)

Fastcode

An open-source community dedicated to advancing software performance engineering

Bruce Hoppe
Fastcode Community Manager
behoppe@mit.edu

https://fastcode.org

What is Software Performance Engineering (SPE)?

SPE is about making software consume few resources such as

- Time
- Storage
- Energy
- Network bandwidth

Properties more important than performance

- Correctness
- Usability
- Security
- Maintainability
- Reliability
- Compatibility
- Etc.

Performance is like currency:

or

We “spend” performance to get more important things.

Once upon a time SPE was important

Apollo 11
Guidance Computer

Launched: 1966

Clock rate: 2 MHz

Data path: 16 bits

Memory: 4KB

Then came Moore's Law

1965-2005: Free performance!

The Demise of Moore's Law

Science
June 2020

REVIEW

COMPUTER SCIENCE

There's plenty of room at the Top: What will drive computer performance after Moore's law?

Charles E. Leiserson[1], Neil C. Thompson[1,2]*, Joel S. Emer[1,3], Bradley C. Kuszmaul[1]†, Butler W. Lampson[1,4], Daniel Sanchez[1], Tao B. Schardl[1]

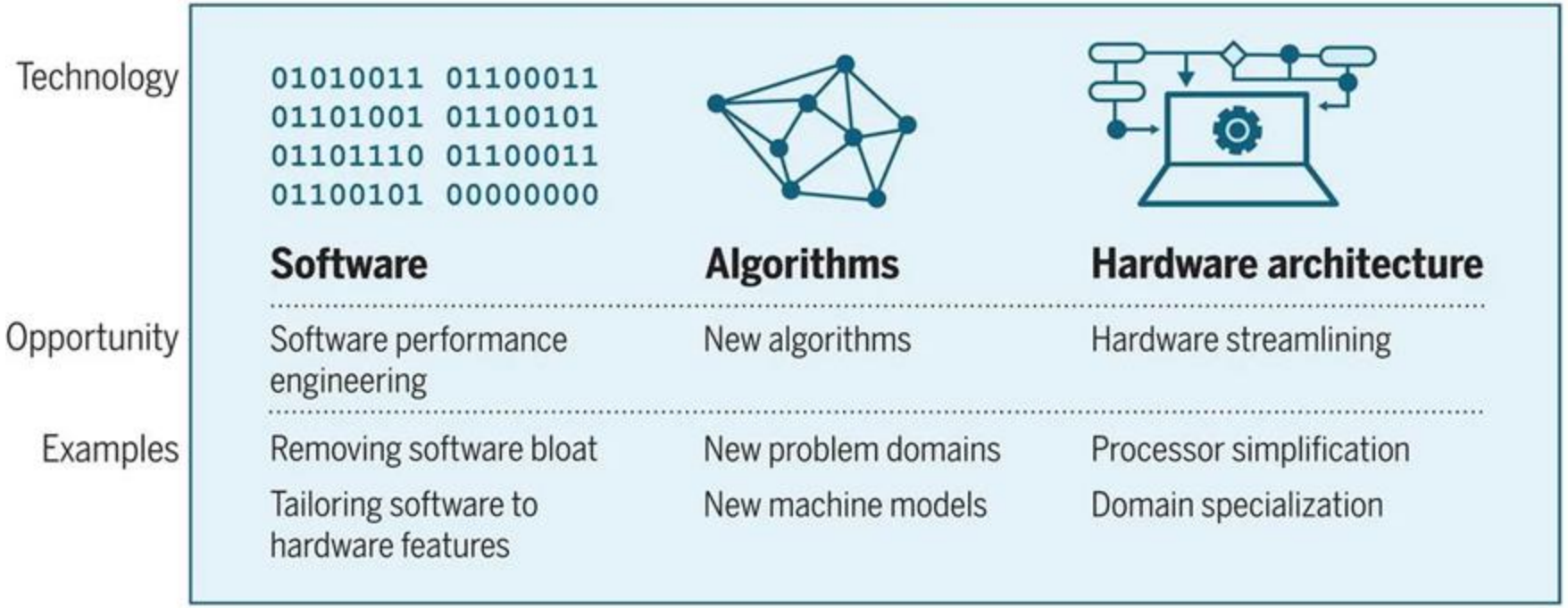
The miniaturization of semiconductor transistors has driven the growth in computer performance for more than 50 years. As miniaturization approaches its limits, bringing an end to Moore's law, performance gains will need to come from software, algorithms, and hardware. We refer to these technologies as the "Top" of the computing stack to distinguish them from the traditional technologies at the "Bottom": semiconductor physics and silicon-fabrication technology. In the post-Moore era, the Top will provide substantial performance gains, but these gains will be opportunistic, uneven, and sporadic, and they will suffer from the law of diminishing returns. Big system components offer a promising context for tackling the challenges of working at the Top.

Over the past 50 years, the miniaturization of semiconductor devices has been at the heart of improvements in computer performance, as was foreseen by physicist Richard Feynman in his 1959 address (*1*) to the American Physical Society, "There's Plenty of Room at the Bottom." Intel founder Gordon Moore (*2*) observed a steady rate of miniaturization and predicted (*3*) that the number of transistors per computer chip would double every 2 years—a cadence, called

physics of materials changes at atomic levels—and because of the economics of chip manufacturing. Although semiconductor technology may be able to produce transistors as small as 2 nm (20 Å), as a practical matter, miniaturization may end around 5 nm because of diminishing returns (*10*). And even if semiconductor technologists can push things a little further, the cost of doing so rises precipitously as we approach atomic scales (*11*, *12*).

In this review, we discuss alternative ave-

Three ways to continue improving performance



SPE is hard to teach

- Curricula is unfamiliar and interdisciplinary.
- Teaching resources are scarce.
- Computing infrastructure is non-standard.

Fastcode Instructors Community for SPE

- Curricula
- Teaching resources
- Computing infrastructure
- Peer support

Charles E. Leiserson

Bruce Hoppe
Fastcode Community Manager
behoppe@mit.edu

https://fastcode.org/get-involved

Modernizing Early Computing Courses with Parallel and Distributed Computing

Sushil K Prasad, UT San Antonio
Alan Sussman, U. Maryland
Chip Weems & Neena Thota, UMass
R. Vaidyanathan, LSU
David Bunde & Jaime Spacco, Knox
Sheikh Ghafoor, April Crockett, & Jerry Gannod, TTU

SIGCSE-25, Feb 27-29, Pittsburgh

https://tcpp.cs.gsu.edu/curriculum/

Public Feedback on TCPP Curriculum & Contact

sushil.prasad@gmail.com

Sponsors