Early and Often, In Community:

Bringing more parallelism into undergraduate CS curricula

Libby Shoop, Macalester College
Curt Clifton, Rose-Hulman Institute of Technology

A Collaboration with

Dick Brown	St. Olaf College
Joel Adams	Calvin College
Mark Gardner	Virginia Tech University
Michael Haupt	Hasso-Plattner-Institut, University of Potsdam
Peter Hinsbeeck	Intel Corporation

A report from ITiCSE 2010 Working Group

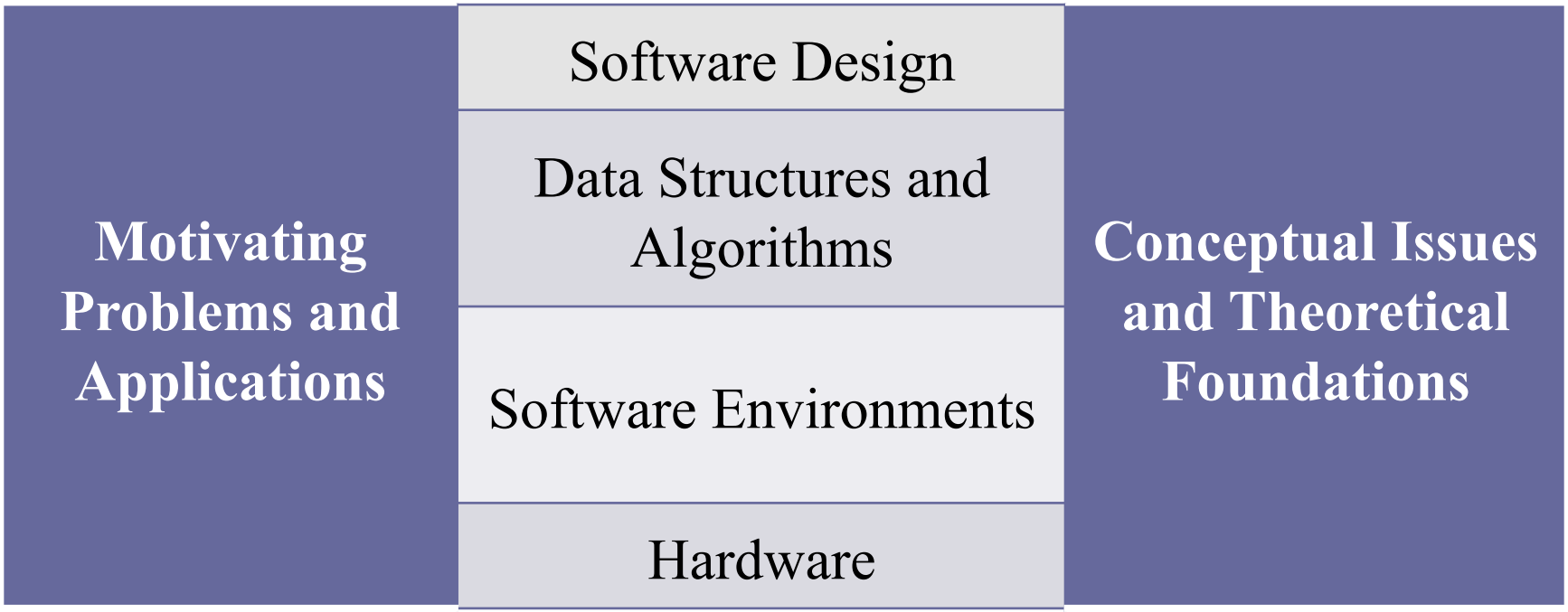
Proposition

- It is urgent and necessary to include more parallelism in undergraduate CS curricula, in many courses at various levels
  - Multicore
  - Data Intensive Scalable Computing
  - Cloud computing

How to meet the challenge

- Decide on a body of knowledge to include
  - With educational objectives
- Consider teaching and learning strategies
  - Offer suggestions on how and where to include concepts
- Help each other through online community
- Consider institutional challenges to bringing about change

A Body of Knowledge in Parallelism



A Body of Knowledge in Parallelism:
Conceptual Issues and Theoretical Foundations

- Computer science graduates will be able to:
  - Identify and discuss issues of **scalability** in parallel computational settings.
  - Define and recognize common types of parallelism and communication, namely **data parallelism, task parallelism, pipelining, message passing, and shared memory communication**.
  - Define **race condition and deadlock**; identify race conditions in code examples; identify deadlock in computational and non-computational scenarios.
  - Assess the potential impact of parallelism on **performance** using Amdahl's and Gustafson's Laws.

A Body of Knowledge in Parallelism: **Software Design**

- Given a problem to solve, CS graduates should be able to:
  - Decompose it into sequential and parallel portions,
  - Recognize possible parallel approaches that can be used to solve it, and
  - Devise and implement an efficient and scalable strategy using a chosen approach.

A Body of Knowledge in Parallelism: **Data Structures and Algorithms**

- Given a problem to solve and a chosen parallel approach to solving it, CS graduates should be able to:
  - Choose an appropriate reliable data structure,
  - Find appropriate existing parallel algorithms that solve the problem, and
  - Devise and implement an efficient and scalable strategy using the algorithm and data structure with that approach.
  - Measure the performance changes of algorithms using various numbers of cores.
  - Compute the speedup and efficiency of parallel algorithms using various numbers of cores.

A Body of Knowledge in Parallelism: **Software Environments**

- Given a problem to solve and a chosen parallel approach to solving it, CS graduates should be able to:
  - Choose an appropriate software library or programming language abstraction for the approach, and
  - Devise and implement the solution using that approach.

A Body of Knowledge in Parallelism: **Hardware**

- Given a problem to solve and a chosen parallel approach to solving it, CS graduates should be able to:
  - Choose appropriate hardware for the approach, and
  - Devise and implement the solution using that approach on that hardware.

Teaching and Learning Strategies

- Early and often
  - Adapt existing curricula
- Spiral approach
  - Students see concepts more than once, with more detail, depth added in later courses

Teaching and Learning Strategies: Adapt existing curricula

- Introductory Level
- Data Structures
- Algorithms
- Programming Languages
- OS, Computer Architecture
- Advanced Electives

Our report provides many examples of how parts of our stated objectives could be brought into various types of courses in these areas, using various languages

Implementation at Rose-Hulman

- New CS Program Outcome:
  - Identify scalable solutions to problems and analyze the scalability of existing solutions under a variety of constraints.

Implementation at Rose-Hulman

- Revised SE Program Outcome:
  - Apply software engineering theory, principles, tools and processes, as well as the theory and principles of computer science and mathematics, to the development and maintenance of complex, scalable software systems.

Parallelism in Courses at Rose-Hulman

120	132	220	221	230	232	241
304	325	332	333	335	351	371
372	373	374	375	376	377	402
403	404	413	432	433	442	451
453	461	463	473	474	479	481
494	495	496	497	498	499	

Implementation at St. Olaf, Macalester: csinparallel.org

- Preparing modules
  - 1 week or less
  - Learning objectives, reading, in-class activities, homework, assessment questions
  - Including help and information about platform and software needed
- Using in several courses throughout our degree programs
  - Introductory
  - Data Structures
  - Algorithms
  - Programming Languages
  - Hardware Design
  - Advanced OS, parallel courses

Community of Educators

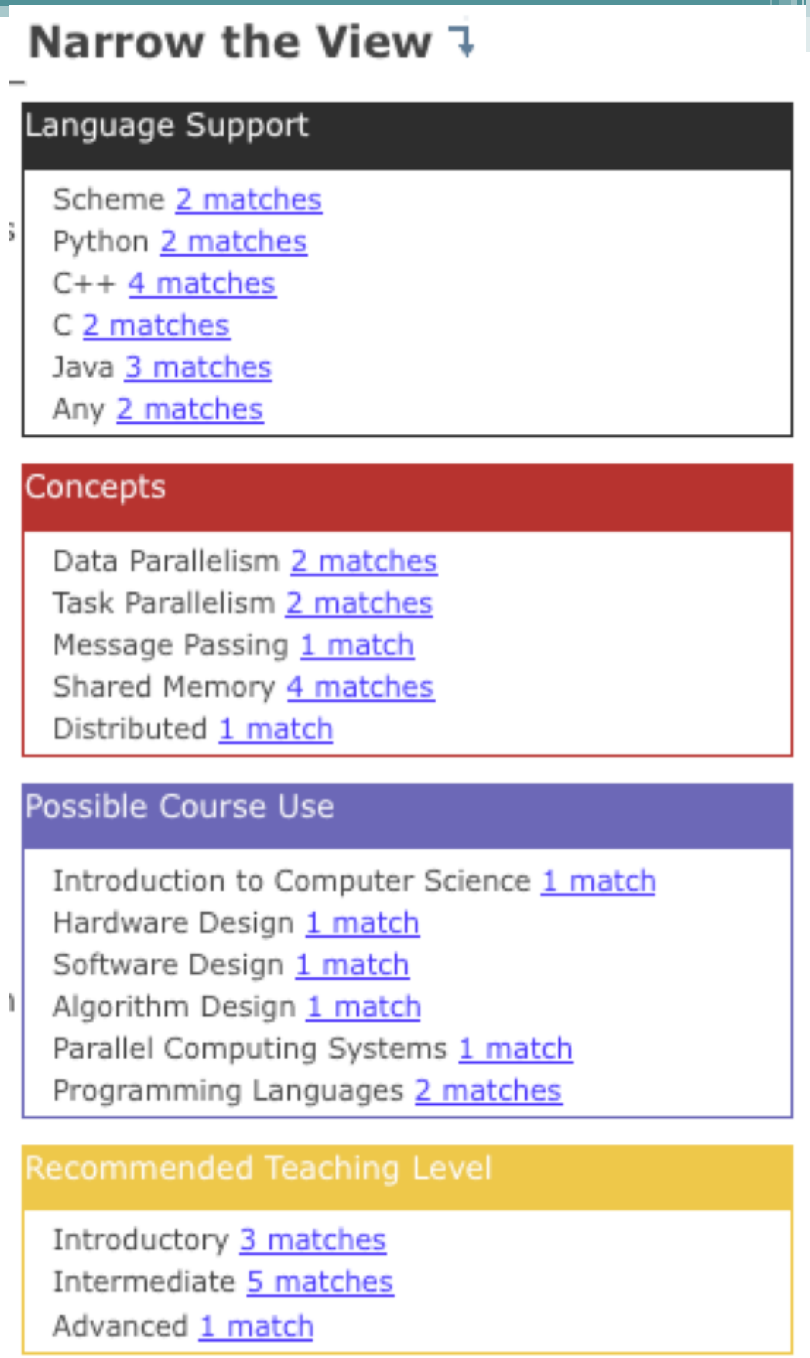
- Change can happen faster if we share ideas and materials
- Need welcoming, supportive environment
  - Contribute as little or as much as possible
    - Receive ‘credit’ for contributing
  - Obtain materials to adapt
    - Must be easy to find
  - Discuss and share information
    - What did and did not work

Community of Educators: Existing Sites

- Intel Academic Community
  - Leading the way in attempt to get academics to share content
  - Marketing flair to site
- http://wiki.opensparc.net/bin/view.pl/CourseMaterial/ConcurrentComputing
- Repositories for all CS material:
  - Ensembl
    - Not much material yet, but a community formed
  - CITADEL

Community of Educators: Existing Sites

- csinparallel.org
  - Result of NSF CCLI grant (Brown, Shoop)
  - Modules
  - Parallel platform packages
    - Software
    - Documentation
  - Organized using controlled vocabulary
  - Others can share, comment



Community of Educators: Practical Difficulties

- Sharing between sites would help
  - How to do this effectively?
- Episodic nature of course development
- Lack of incentive
  - Taking part needs to be valued as service to profession
- Small community size
- Shared content may not be easily adapted by others
- Technology changes quickly

Summary of Suggestions

- Incremental change
  - Materials with learning goals
  - Can assess their effectiveness
- Early and Often
- Spiral
- Experiential

- Proceed in and from Community of Educators

Institutional Challenges

- Curricula already full
  - Small incremental change approach
- Resistance
  - This has been tried before…
  - How can we persuade of the importance?
- Faculty have to get up to speed
  - Must try to make this easier

- ABET: we didn't consider very much

Acknowledgments
Online Advisory Panel Discussion Members

- Clay Breshears, Intel Corporation;
- Daniel Ernst, University of Wisconsin—Eau Claire;
- Gregory Gagne, Westminster College;
- Michael Heroux, Sandia National Labs and St. John's University;
- Jeanne Narum, Project Kaleidoscope; and
- Matthew Wolf, Georgia Tech and Oak Ridge National Labs.