SIGCSE 2010 BOF ON CURRICULA IN CONCURRENCY AND PARALLELISM

ABSTRACT

The idea for this BoF spawned from the First Workshop on Curricula in Concurrency and Parallelism, held in conjunction with OOPSLA 2009 in October.

The issues of parallelism and concurrency in the CS curriculum have begun generating a lot of interest at SIGCSE (and elsewhere) in the last two years. Industry sponsors, such as Intel and Google, have promoted these issues actively, and educators have similarly engaged in the discussions, with education-oriented workshops at ASPLOS 2009 and OOPSLA 2009, plus a panel and a BOF at SIGCSE 2009. This year at SIGCSE 2010, the focus continues with another panel, multiple tutorials/workshops, two related paper sessions, a keynote, and further industrial participation.

A FEW RESOURCES...

- Ning site for OOPSLA workshop: http://concurrencypedagogy.ning.com/
  Includes list of outcomes from the workshop, along with position papers from attendees.

- Intel’s Academic Community: http://software.intel.com/en-us/academic/
  - (Courseware: http://software.intel.com/en-us/courseware)
  A community for educators interested in teaching parallelism. Besides significant amounts of course material, this site hosts the “Teach Parallel” video interview series, along with blogs and forums which give access to Intel expertise.

- Sun’s Concurrent Computing Education Wiki:
  http://wiki.opensparc.net/bin/view.pl/CourseMaterial/ConcurrentComputing
  Shared teaching material for concurrent computing

- Data-Intensive Scalable Computing (DISC)
  - Google’s Code University: http://code.google.com/edu/parallel/
  - DISC in Education Google Group (with links): http://groups.google.com/group/disc-in-edu

- HPC University: http://www.hpcuniv.org/
- CSERD: http://www.shodor.org/cserd/
  HPC (Parallel Computing) and Computational Science portals for NSDL.
Notes on Outcomes from the First Workshop on Curricula in Concurrency and Parallelism

The Context

- **Software trends**: Interacting programmable systems are everywhere: internet, cell phone networks, social websites, cloud computing, interactive multiplayer games, millions of GUIs generating "events", client/server, etc.
- **Architectural trends**: clusters, multi/many-cores, GPUs, heterogeneous hardware

*Existing software and software methodologies were/are ill-prepared for these trends*

Important Concurrency Ideas for a CS Curriculum

- Identifying potential parallelism
- Using "safe" parallelism (where the artifacts of underlying implementation do not alter program behavior)
- Seeing the distinction between "total work" and "critical path length" (necessary delay)
- Exploiting parallelism – basic types:
  - Recursive (divide-and-conquer) parallelism
  - Stream parallelism
  - Data parallelism
- Understanding the difference between "threads" that share a memory address space and "processes" that do not

Experiences that Students Should Have

- building a multiplayer game
- building a shared database
- building a fast numerical code
- building a discrete event simulator

Issues on Which We Were Divided as to Pedagogical Approach

- Languages vs. libraries; if a new language, which?
- Shared vs. distributed memory
- Should course CS1 use concurrency at all?
- How much concurrency should be taught to non-majors?
- "How low should you go?" Should monitors, locks, semaphores, Dekker's algorithm be emphasized?
  - For all? Or for a few?
- How important is performance to pedagogy?