Teaching Multicore Programming: When, What, How?

Håkan Grahn | MCC-11 | 2011-11-24

www.bth.se

BLEKINGE INSTITUTE OF TECHNOLOGY

What should all students in computer science know?

- When leaving with a *bachelor degree*?
  - C/C++ (Java, …)
  - Concurrency concepts, e.g., threads, synchronization, libraries, directives, etc.
  - Correctness, correctness, correctness, …
- When leaving with a *master degree*?
  - Heterogeneous systems
  - Performance, performance, performance, …
- When leaving with a *PhD degree*?
  - Develop future multicore technologies ☺

What are the strengths and deficiencies in current university education?

Strengths:

- High competence among some teachers
- Many courses exist today, but mainly at the master level
- Strong research in the area

Deficiencies

- General awareness/knowledge too low, i.e., far too few with the right competence
- Multicores are not considered a major problem

Parallel platforms, languages and tools?

- Platforms
  - Knowledge about both homogeneous and heterogeneous platforms
- Languages
  - C/C++ (Java,…)
  - Pick your favorite parallel language ☺
- Tools
  - For testing, debugging, and verification
  - For performance evaluation and optimization

Important parallel computing concepts and programming techniques in 5 and 10 years ?

- C/C++ programming skills (Java, Fortran, …), with appropriate parallel extensions and libraries
- Domain specific languages will probably be more important
- More advanced compilers and runtime systems, e.g., StarPU, auto-tuning, …
- Software engineering for multicores

Suggestions for universities?

- Introduce parallel programming already in year 1
- Teach methods and techniques for verification & validation, testing, and debugging
- Develop new education programs with main focus on parallel programming
- Agree upon a basic curricula for multicore / parallel systems