Per Holmberg

Computer Architect

Ericsson

- This is from the viewpoint programming large scale applications on multicore processors.

- Views are my own.

- Getting parallelism in application
  - We know the application and are generally Ok in this.
    - Problems are local, to specific parts of SW, not large scale
      - but of course, that does not help that team....
- Expressing parallelism in SW
  - There is little new, communication systems are message based, distributed, in nature
    - starting-point of >100 processors → 1000 cores → 10K cores/threads
    - algorithms tends to be better handled in specialized HW accelerators when possible
      - interpreting algorithms in SW is neither very power nor cost efficient

- Most important things
  - Parallelism V.S. SW Robustness
    - Avoiding the worst error-prone constructs
    - Avoiding the hard to test ones
  - Parallelism V.S. Verification!
    - How to stress race conditions
    - Within reasonable test time
  - Parallelism V.S. Real Time
    - Guarantees by priority or placement?
    - Guarantees when bottleneck is somewhere else than in the core?
  - Parallelism V.S. Optimization
    - Parallelizing for multicore seems to be first proposal to any performance issue
    - Great risk of parallelizing inefficiency

- * What should all students in computer science and engineering
- know about parallel computing and multicore programming
- when they leave with a bachelor, master or PhD degree?

  - No difference as before: A, D, P and the different tasks during development (systemization, writing code, verification)

- * What are the strengths and deficiencies in current
- university education regarding parallel and multicore programming?

  - numerical computing /algorithms – this is where parallel programming comes from, but now it is everywhere

- * If any, what concrete parallel platforms, languages and tools
- should students learn about / work with in their university
- education?

  - No.... this type of education is readily available in industry anyhow, university should focus on problem understanding.

- * Which parallel computing concepts and programming techniques
- do you foresee as important in 5 and 10 years from now?

  - Same as today – revolutionary steps will be local, for example for DSP algorithms.

- * Do you have concrete suggestions for universities how
- they could improve or extend their teaching of parallel and
- multicore programming?

  - It should be an integrated part of SW development, not a add-on at the end...