Microsoft

Microsoft Research Faculty Summit 2012

ADVANCING THE STATE OF THE ART

Parallel programming for undergraduates

David Padua
University of Illinois

1. THE ISSUES

An open question

- Widespread interest in **expanding the coverage** of parallel programming in the CS core.
- We have not reached a steady state
- There is no clear consensus on
  – whether this should be done,
  – what topics to cover, and
  – when and how to cover them.

2. WHETHER

Should we ?

- Parallelism is ubiquitous in computing and has always been.
- In recognition of this, is common place to teach about processes, synchronization, deadlock.
- However, that was ok when parallelism came mostly from overlapping I/O and CPU
- **More is needed** in our times. Three examples next.

3. WHAT
TOPIC 1: PERFORMANCE

illinois.edu

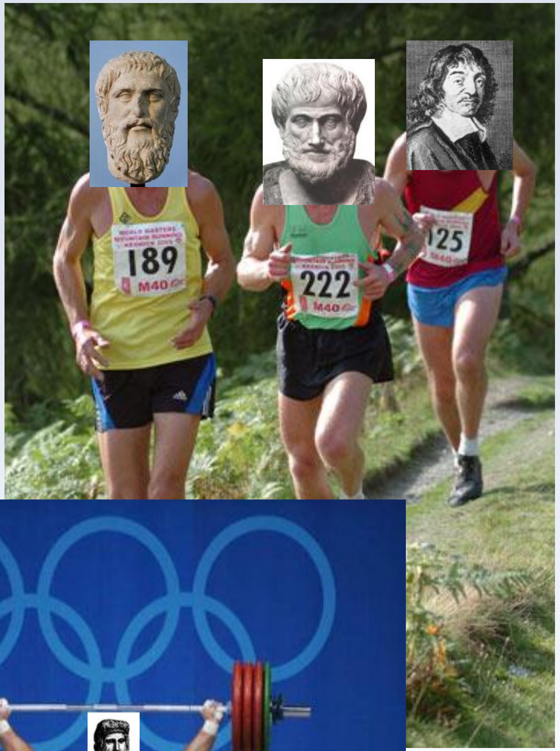
Today's emphasis is in expressiveness and correctness

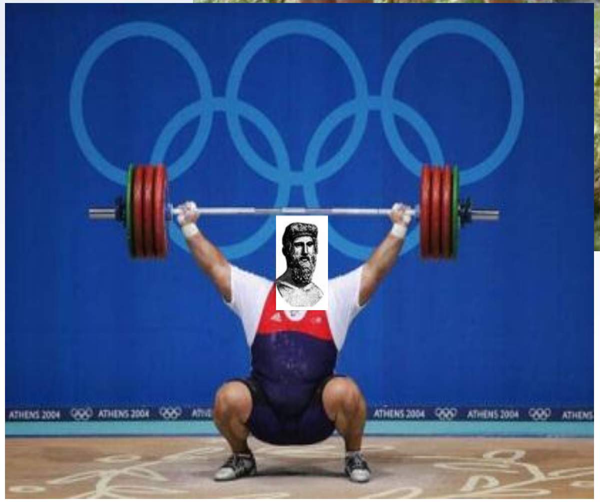
- Parallel programming for expressiveness:
  - Simulations (real word is parallel)
  - Reactive codes – "dining philosophers"
- These can be represented in sequential form, but less clearly.

Figure by Benjamin D. Esham / Wikimedia Commons

Performance

- However, another equally important dimension is the not-so-much-in-fashion **performance**.
  - Physical limitations slowed performance improvements and led to the advent of multicores
  - Parallelism needed for continued gains in execution **speed**.
  - Fixing speed, parallelism can reduce **power** (energy) consumption





Education in performance

- Concepts in parallel programming are not that difficult.
- Understanding and attaining performance improvements (speedup) and high efficiency can be challenging.
- Need understanding of machine organization, compilers, runtime systems, algorithms and the interactions between these.
- Need to understand and develop skills to measure program behavior

3. WHAT

TOPIC 2: RACES AND NON DETERMINACY

Easy concepts difficult praxis

- The notion of race condition and non-determinacy are relatively easy.
- Finding sources of these errors can be difficult.
- Need tools.
- Need much experience.

3. WHAT
TOPIC 3: ABSTRACTIONS

Programming at the high level

- Using thread spawning and synchronization is low level programming.
- Use of abstractions is the way of the future:
  - Array/collective operations
    - e.g. Map reduce/MPI reduce
  - Parallel loops
- Numerous languages/notations widely available for teaching.

4. IN WHAT COURSES TO TEACH PARALLEL PROGRAMMING

Courses

- Specialized courses
  - Parallel programming
  - Parallel algorithms
  - Program optimization techniques
  - Compiling for parallelism
  - Heterogeneous parallel programming (now a coursera MOOC)
- An effective strategy
  - Spread parallelism throughout the CS core curriculum.
  - Machine organization, algorithms, data structures.

Experience indicates that it is feasible and effective

- Spring 2012. CS 225 Data structures and Programming Principles.
- Three lab sessions (out of 14) devoted to parallel programming
  - Replaced sessions devoted to exams questions review.
  - Session 1: First encounter with parallel programming – fully parallel OpenMP loops.
  - Session 2: Races and non-determinacy
  - Session 3: Reductions
  - Session 4 (planned for future semesters) tasking, recursive parallelism (e.g. quicksort)
- TAs did all of the teaching.
  - Intel colleagues trained TAs in the use of tools.
  - Instructors prepared material for the TAs.

Useful material on topics that can be covered for undergraduates and in which courses

- **Developed by NSF/TCPP Curriculum Standards Initiative in Parallel and Distributed Computing – Core Topics for Undergraduates.**

http://www.cs.gsu.edu/~tcpp/curriculum/index.php

5. CONCLUSION

- For most computer scientists programming is at the center of their profession.
- Parallelism will be an increasingly important part of programming.
- CS core curricula must evolve accordingly

Microsoft