Teaching by example: using analogies and live coding demonstrations to teach parallel computing concepts to undergraduate students

Nasser Giacaman
Parallel and Reconfigurable Computing group,
Department of Electrical and Computer Engineering,
The University of Auckland, New Zealand

21st May 2012
Workshop on Parallel and Distributed Computing Education
IPDPS 2012, Shanghai, China
From down under...

New Zealand
Zooming in...
Auckland
The University of Auckland

- NZ's largest university
- 40,000 students
- Department of Electrical and Computer Engineering
Parallelisation of desktop environments
Overview

- Course context
- “First impressions count”
- Analogies and live coding demonstrations
- Multi-threading contents
- Supplementary learning material
- Student feedback
Course context

- 2nd year Computer Science
- 220 students enrolled
- Two 1st year prerequisite courses:
  - Introductory programming
  - Advanced data structures
Course context

- Course aims: to introduce students to developing more complex software applications
- 12 weeks split as follows:
  - First exposure to OOP (4 weeks)
  - Frameworks, GUI in particular (3 weeks)
  - Testing techniques (1 week)
  - Application-level multi-threading (4 weeks)
First impressions count

- The difference between:
  - **Sequential**
  - **Concurrent**
  - **Parallel**
First impressions count

- **Sequential** programs have 2 problems:
  - GUI freezes until computation is completed
  - Only 1 CPU core is utilised
First impressions count

- **Sequential** programs have 2 problems:
  - GUI freezes until computation is completed
  - Only 1 CPU core is utilised
First impressions count

- **Concurrent** programs have 1 problem:
  - Only 1 CPU core is utilised
First impressions count

- **Parallel** programs solve both problems

100% 100% 100% 100%
First impressions count

- “Performance” means many things:
  - Decreasing wall-clock time
  - Improving user experience (interactivity)
- We want both:
  - Concurrency for interactivity
  - Parallelism for performance
Motivation: “The free lunch is over”
Increasing processor speed
Increasing processor speed
Increasing processor speed
Increasing processor speed
Increasing processor speed

Dual-core processor 2005

Quad-core processor 2006
Multi-core processors
So what's the problem?
“The free lunch is over”
So what's the problem?
“The free lunch is over”
Overview

• Course context
• “First impressions count”
• Analogies and live coding demonstrations
• Multi-threading contents
• Supplementary learning material
• Student feedback
Theory delivered as analogies

- The “Quad-core” processor
Theory delivered as analogies

- “Threads”
Theory delivered as analogies

- Too few threads
Theory delivered as analogies

- Too many threads

Advertise job and hire new employee (thread creation), sit on desk/get off desk (context switching), release the employee, ...
Theory delivered as analogies

● The “Runnable” task

Permanently hire fixed number of employees, each sits on own desk. Work written on paper (the tasks) and placed in a pile shared amongst the workers.
Theory delivered as analogies

- Sharing resources

Access to shared resources must be managed to avoid conflict. For example, have a lock per item, only 1 key
Live demonstrations to reinforce concepts

- Coding in-class, almost half the lecture
- "Slow-motion", e.g. classical race condition to cause bug
- Code relates to analogy, e.g. `Thread bob = new Thread();`
- Builds on from previous demo/lecture
- "What if you change XYZ?" → let's see!
Multi-threading contents

4 weeks total, 3 lessons per week

• Motivation and introduction to threads (3 lessons)
  – Threads, Runnables, critical sections, thread safety mechanisms, etc

• Concurrency in GUI applications (3 lessons)
  – Event loop/handlers, GUI thread, GUI components, SwingUtilities, SwingWorker

• Parallel computing on multi-cores (6 lessons)
  – Threading vs Tasking, multiple SWs, ExecutorService, Scheduling policies
Supplementary learning material

- Practical focus on the course:
  - Live-coding (about half the lesson)
  - Examples from class were made available online after each lesson
  - Lectures recorded (captures lecturer voice and records projector). Students can “replay” the lesson
  - Practical assignment. Provided with a sequential GUI app, and asked to multi-thread for concurrency and parallelism
Student feedback

- 105 students responded to the anonymous online evaluation
- “The way material was presented assisted my understanding of the subject”
  - 47% SA, 46% A, 5% N, 2% D, 0% SD
- “The lecturer stimulated my interest in the subject”
  - 54% SA, 39% A, 6% N, 1% D, 0% SD
- “The lecturer stimulated my engagement in the learning process”
  - 45% SA, 47% A, 8% N, 1% D, 0% SD
Student feedback

- "What was most helpful for your learning?"
  - 53x live coding & practical examples in class
  - 16x use of analogies to explain
  - 13x video recordings
  - 8x code available online
Conclusions

• Providing a practical approach to teaching parallel computing provided for higher engagement from students in the learning process:
  – *During* the lesson (use of analogies, live coding and demonstrations)
  – *After* the lesson (code online, practical assignment)