Early Adopter - PDC Education Early and Often

At a Four-Year Liberal Arts College

Steven Bogaerts, Kyle Burke, Brian Shelburne
Department of Mathematics & Computer Science
Wittenberg University
Springfield, OH

Melissa Smith
Department of Electrical & Computer Engineering
Clemson University
Clemson, SC

Eric Stahlberg
CCR Bioinformatics Core
National Cancer Institute
Bethesda, MD

Introduction

Our theme in introducing parallelism and distributed computing (PDC) concepts into our curriculum is to focus on *integration*, as opposed to *insertion*. Historically, any PDC content in an undergraduate curriculum appeared in the context of operating systems, and perhaps algorithms or an upper-level elective. It is undeniable that PDC is of such significance today that it cannot be relegated solely to such courses.

Here, we outline our methodology and experiences of integrating PDC into undergraduate courses at various levels, in connection with the proposed core curriculum on PDC. We show that each course carries ample opportunities to consider PDC in a manner that is both level-appropriate and in harmony with the traditional topics of the course.

CS0 or Computer Literacy
General education elective, 12 students

Syllabus

- Programming in Scratch - variables, loops, if statements, objects
    - Parallelism - terminology, history, physical activities [1] and demonstrations
- Excel - practical applications for business and finances
- Access - introduction to relational databases and queries

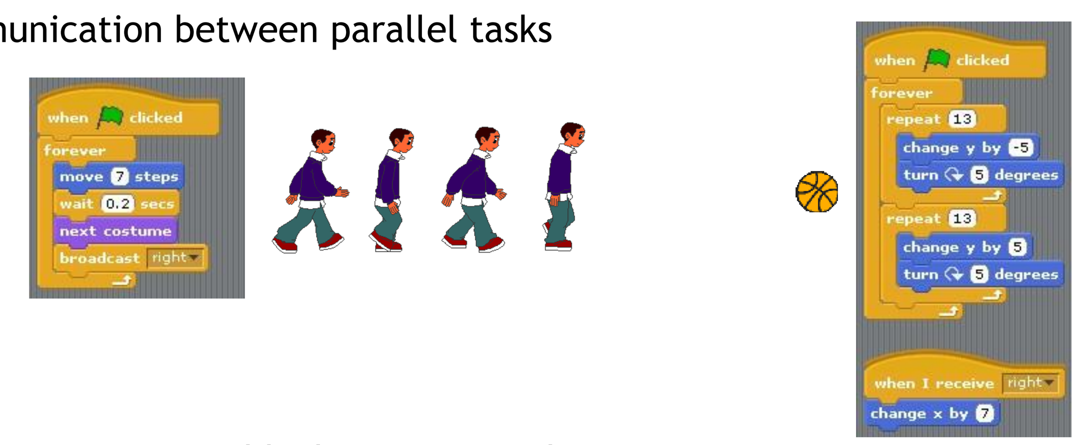
Curriculum Initiative Topics

- Architecture: Multicore (K)
- Programming: Fixed number of processors (K)
  Critical regions (K)
- Algorithms: Scalability (K)
  Broadcast (C)
  Asynchrony (K)
- Cross-Cutting: Why and what is PDC? (K)
  Non-determinism (K)
  Power (K)
  Cluster (K)
  Cloud/Grid (K)

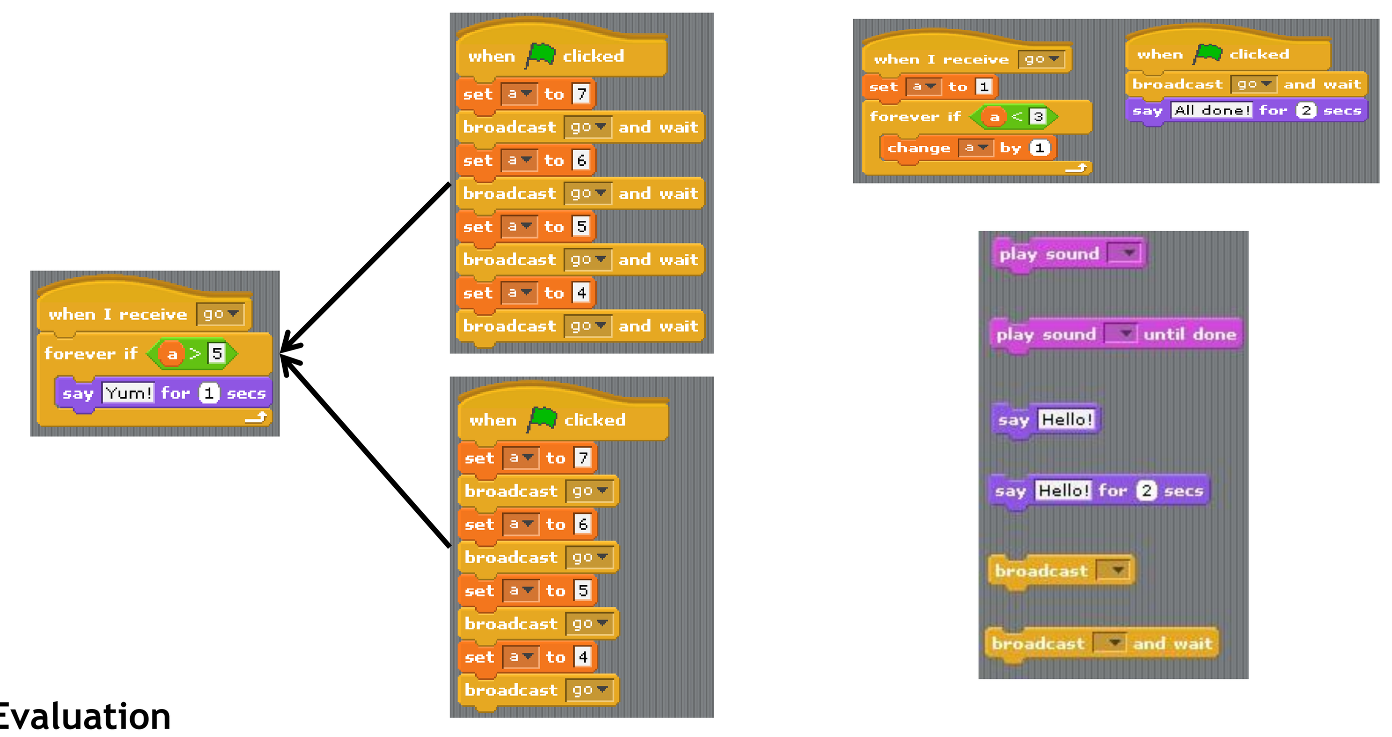
Integration

Scratch provides ample opportunities for easy exploration of concepts in parallelism.

- Communication between parallel tasks



- Blocking versus non-blocking commands



Evaluation

- Strong performance (A's and B's) on parallelism assignments and exam questions.
- Parallelism in Scratch - students report high understanding (4.7/5.0) and enjoyment (4.4/5.0).
- General parallelism discussions - students report very good understanding (4.0/5.0) and enjoyment (4.0/5.0).
- Material was very easy to integrate into the class in a level-appropriate manner.

CS1
1st year, core, 20 students

Syllabus

- Programming in Python - variables, input/output, loops, strings, if statements, objects, classes, parallelism, basic data structures and algorithms, abstraction

Curriculum Initiative Topics

- Architecture: Multicore (K)
  Message passing (K)
- Programming: Client/Server (C)
  Fixed number of processors (K)
  Critical regions (C)
- Algorithms: Scalability (K)
- Cross Cutting: Why/what is PDC? (K)
  Power (K)
  Cluster (K)
  Cloud/grid (K)

Integration

- Use multiprocessing as a *medium* for core
- Python multiprocessing - minimal syntax!
- Fork, join, communication, locks
- Medium for: classes and objects, parameter passing, modularity and abstraction, searching and sorting, pattern matching, clustering, simulations

```
def greet(q):
    print "(child process) Waiting for name..."
    name = q.get()
    print "(child process) Well, hi", name

def sendName():
    q = Queue()

    p1 = Process(target=greet, args=(q,))
    p1.start()

    time.sleep(5) # wait for 5 seconds
    print "(main process) Ok, I'll send the name"
    q.put("Jimmy")
```

Do what you usually do, just do (some of) it in parallel.

Evaluation

- Students successfully completed projects on producer/consumer, concurrent encryption, and a simple pipeline
- First time, in Fall 2010, multiprocessing was introduced a bit too early (4-5 weeks)
    - Students struggled with some basic semantic issues
- Second time, Spring 2011, multiprocessing introduction was delayed to about week 11 - much more successful

Advanced Data Structures Algorithms
3rd/4th year, core, 6 students

Syllabus

- Algorithm analysis, brute force, divide-and-conquer, transformations, space and time tradeoffs, dynamic programming, greedy algorithms, iterative improvement

Curriculum Initiative Topics

- Algorithms: Costs of computation (A)
  Asymptotics (A)
  Time (A)
  Space (A)
  Speedup (A)
  PRAM (A)
  Divide & Conquer (A)
  Recursion (A)
  Scan (A)
  Reduction (A)
  Synchronization (A)
  Sorting (A)
  Selection (A)
  Search (C)

Integration

- Use of Chapel as an easy language to pick up for this work
- Pattern throughout course of sequential algorithm followed by parallelization
- Example applications: reductions, parallel merge sort

Evaluation

- Students report enjoying the parallel algorithms very much - insisting to see a parallelization after each sequential algorithm!
- Students wanted to see even more parallel programming projects.

CS2 + Core Data Structures
1st/2nd year, core, 10 students

Syllabus

- Programming in Java - CS1 concepts, OOP, Swing, basic algorithm analysis, generics, recursion
- Data structures - lists, stacks, queues, deques, trees, maps, heaps, priority queues, graphs
- Parallelism - Java ForkJoin Framework, as described in [2].

Curriculum Initiative Topics

- Programming: Shared memory (A)
  Client server (K)
  SPMD (C)
  Data parallel (A)
  Parallel loop (C)
  Language extensions (A)
  Libraries (C)
  Tasks and threads (C)
  Computation (C)
- Cross Cutting: Concurrency (K)
  Power(K)
- Algorithms: Asymptotics (K)
  Time (K)
  Space (K)
  Speedup (K)
  Scalability (K)
  Dependencies (A)
  Divide & conquer (A)
  Recursion (A)
  Reduction (C)
  Asynchrony (K)
  Synchronization (K)
  Selection (C)

Integration

- Parallelism is a *medium* for core CS2 topics
- Java ForkJoin Framework (Java 7)
    - Recursion (divide and conquer), trees and stacks (understanding recursion and forking processes), Java generics, time and space considerations, inheritance, parameter passing and shared memory, exception handling

Evaluation

- Students found the fork/join material to be a fun culmination of a number of Java and data structures topics.

Computational Modeling (Under Development)
1st/2nd year, elective, 15 students

Syllabus

- Computational modeling as an integration of natural science, computer science, and applied mathematics
- Experimental methods, simulation, dimensional analysis, visualization, optimization
- Computer implementations primarily in Mathematica

Integration

- Mathematica Parallel Computing Toolkit gives built-in parallel programming primitives
- Exercises in reductions, matrix operations, parallel plotting of data
- To be tried for the first time spring 2012.

References

[1] Maxim, B. D.; Bachelis, G.; James, D.; and Stout, Q. 1990. Introducing Parallel Algorithms in Undergraduate Computer Science Courses. In *Proceedings of the Twenty-First SIGCSE Technical Symposium on Computer Science Education, p. 255. New York, NY, USA: ACM Press.*

[2] Grossman, D. 2011. A Sophomoric Introduction to Shared-Memory Parallelism and Concurrency. http://www.cs.washington.edu/homes/djg/teachingMaterials/

Acknowledgement

We gratefully acknowledge the support of the National Science Foundation through grant CCF-0915805, SHF:Small:RUI:Collaborative Research: Accelerators to Applications - Supercharging the Undergraduate Computer Science Curriculum.