Teaching Parallel and Distributed Computing at a Liberal Arts College

Tia Newhall

Swarthmore College

newhall@cs.swarthmore.edu

Swarthmore College CS

- Swarthmore has ~1,400 students
- ~15 CS majors each year (but 42 junior CS majors!)
- CS Dept has 4 tenure lines (6 in two years)
  - We try to cover a lot of CS with 4-6 faculty
    - I'm the lone systems person
    - Upper level courses offered once every other year
- CS curriculum not very vertical (typical for LACs)
  - CS1 and CS2 are only pre-reqs to upper level CS

  => cannot assume much/any background in systems

  (we are adding a new course to address this)

Teaching Parallel & Distributed Computing

- Wide variation in student preparedness
  - I can't assume much: need some intro to systems
    - too little for some, and too much for others
- Want some seminar-style courses in our curriculum, and this has been one
  - Research paper reading, discussion, independent projects, presentations, written work, less lecture
- Expose them to wide-range of issues in distributed and parallel computing and to a large number of different systems
  - Sometimes choose broad coverage over deep
  - Project is chance for depth

What I've tried ….

Distributed Systems (CS85, CS97)

Pure seminar-style (only a couple short intro lectures)

- Discussion of 2-3 papers read each week
  - Broad coverage of field, with some depth
  - Classic theory to current systems
  - Each presented one paper (and related)
- Assigned a couple lab assignments just to give them programming tools for projects
  - MPI, and a C client/server socket (talk, string mangler)
- Independent course project
  - Very open ended, I give them some ideas, but can do anything related to DS, must have a question
  - Propose, carry out, experiment, written and oral report
  - Like a CS research experience

Distributed Systems (CS85, CS97)

What worked well:

+ format allows for large coverage of field
+ students gain good understanding of field
+ very good at reading papers and discussion
+ good independent projects, but variable
+ particularly good for students going on to grad school

What didn't work so well:

- some papers too hard or don't have background for
- didn't always have tools to carry out projects
- DS seemed too specialized and students didn't really know what the course was about

    robotics, graphics, etc. they at least think they know

- we needed to inject some parallelism into our curriculum, and this seemed like a place to do it

Parallel & Distributed Computing (CS 87)

- Very broad coverage of two big fields
  - ~1/3 systems, ~1/3 PL, ~1/3 algorithms

  architecture, algorithms, programming interfaces and languages, systems, lots of analysis of system components to algorithms, scalability, ...
- 1/2 lecture-based, 1/2 seminar-style
  - Lecture more in 1st half, mostly on parallel
  - “Principles of Parallel Programming”, Lin & Synder
- 5 “short” labs I assign in 1st half
  - Give them more practice with parallel & distributed programming before project
- Independent project in 2nd half
- Weekly lab scheduled meetings added to class
  - teach them SW & tools, help on lab and projects

5 “Short” Lab Assignments

- Give them exposure and practice with parallel & distributed programming
- Give them practice with designing and running experiments
- They demo all labs to me
  - Think about correctness and error handling more
  - Learn to discuss how and why of their solution
- I assign different partners for each lab

Lab 1: C warm-up

- Pointers, dynamic memory allocation, scope, pass by reference, file I/O, …
- Multiple .c files, .h, extern, static
- gdb, valgrind, make

+ almost all <u>really</u> need this
- replaced an assignment I really liked:
  - Investigate a parallel system and present it to class
? Hope a new course we are adding to our intro sequence will solve the problem this addressed

Lab 2: Shared Memory

- pthreads GOL with 2 thread to board mappings
  - threads, synchronization

- Scalability analysis Part: experiments and report
  - vary problem size, #threads, # CPUs,
- Write-up: implementation, experiments, hypotheses,results, discussion of results
  - Good practice for course project
- Also more C programming practice:
  - gdb, valgrind, make
  - parsing command line options (getops, -l style)

Lab 3: TCP client server

- Multi-threaded Web Server
- They investigate HTTP 1.1 specification, figure out and implement HEAD and GET protocols
  - C TCP sockets, pthreads, signals, mutex

+ I really like this assignment

+ they learn a lot and its fun

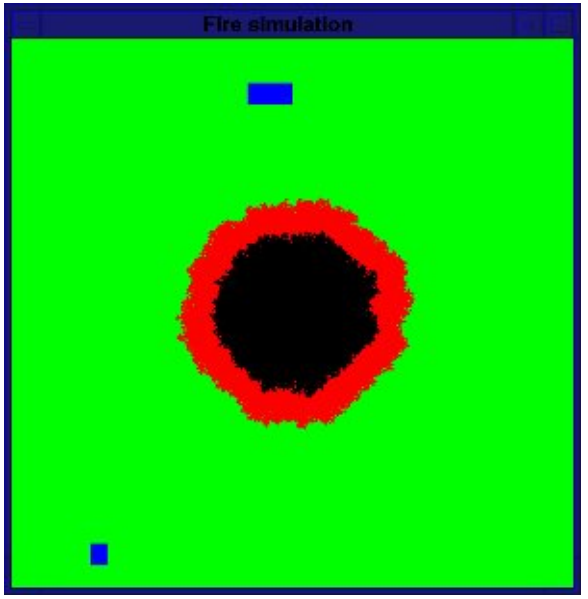
- Bryant and O'Hallaron book student site: full source to a multi-threaded web server in C

Lab 4: Cuda

Fire simulator

- replaces an OpenMP lab
- Many are interested in Cuda-related projects

- I give them a lot of starting point code including library to visualize simulation on GPU
- Gives them practice compiling and running on the GPU, timing
- Writing and calling Cuda kernels
- Copying to-from CPU-GPU
- Figuring out Cuda programming & synchronization models



Lab 5: MPI using XSEDE

- Did as weekly lab instead of assigned
- Usually fairly simple MPI program
  - Practice with message passing
  - Practice using XSEDE resources
  - I give them examples and documentation for using XSEDE
    - Simple MPI: code, makefile, job submit script
    - MPI-CUDA Hybrid: makefile (its tricky)
  - Use XSEDE as a resource for projects

Lab Projects

- Good preparation for course projects
- I'd like to do more, more parallel algorithms, different programming paradigms, etc. but, I only have 1/2 of the semester for these
- The labs and the topics covered in the first half, greatly influence student's independent project topics
  - We don't do a lot of DS until second 1/2 and there are few DS projects

Weekly scheduled labs

Goals:

1. Learning and practice with SW, Unix utilities, programming environments, etc.
2. Help on lab assignments/projects

Specific Lab Presentations/Topics/Practice:

1. C programming, multiple modules, make
2. Setting up and using git repos
3. Gdb, valgrind, man, appropos
4. Tools for running experiments: script, screen, bash scripts
5. Tools for measuring: time, gettimeofday, gprof, …
6. Obtaining system information: /proc, top, netstat, …
7. Socket, Cuda, MPI, OpenMP, …
8. Using XSEDE
9. Unix SW for documents: latex, gnuplot, …

Independent Project

Assigned near end of first 1/2 of semester

I give them some ideas, but can do anything related to parallel or distributed computing

Must have research question

Multi-part: I've added more parts over the years

1. Written Proposal and Annotated Bibliography
2. Mid-way progress report and oral presentation to class
3. Project work week: short report
4. Final oral presentation to class
5. Final written report (like conference paper) and project demo

My Thoughts

+ covers important content not covered anywhere else
+ I like teaching both parallel and distributed, and think both important
+ 1/2 lecture helps reinforce basics, better understanding
+ more assigned labs good background, broader learning
+ weekly lab meetings ensure all students getting instruction & practice
+ individual project components help keep them on task
- less good at reading, discussion, reaction notes
- most lecture in 1st half, maybe no way around this
- lecture primarily on parallel, readings primarily on distributed
- fewer papers, so one bad choice has larger effect
- Broad coverage of 2+ courses into one: lose breadth and depth
  - I always have to cut things I'd like to keep in
- Maybe need to add an exam on papers and lecture

Overall: I like this course & I like it better than DS

More Information

- Links to versions of each course off my webpage (CS87, CS85, CS97):
  - Schedule: topics and readings
  - Lab assignments, and weekly lab content
  - Project components
  - Links to resources

    www.cs.swarthmore.edu/~newhall

- Feedback, suggestions, ideas, …

  newhall@cs.swarthmore.edu

Thanks. Questions?

New Course Developing

- Intro to Computer Systems:
  - machine organization, assembly, compilers, systems, intro to parallelism, C programming
- Taken after our CS1 course in Python
  - Students can take CS2 or this in any order
- This will be a pre-req to some courses
  - ~1/2 upper level require: CS1 and CS2
  - Other 1/2: CS1, CS2, plus new course

  => We can assume students have seen this before OS, parallel and distributed, compilers, graphics, DBMS, … !