Slide 1

Administrivia

- One purpose of the syllabus is to spell out policies, especially about:
  - Course requirements and grading.
  - Exam dates.
  - Late work.
  - Academic integrity.
- Most other information will be on the Web, either on my home page (here, office hours) or the course Web page (here).

  A request: If you spot something wrong with course material on the Web, please let me know!

Slide 2

More Administrivia

- Part of my job is to answer your questions outside class, so if you need help, please ask! in person or by e-mail or phone.
- Some of my office hours are designated as "open lab". At those times I will be in one of the labs (probably HAS 228) ready to answer questions.

Slide 3

More Administrivia

- Probably the easiest (though not the only) option for doing the assignments is to use the Linux lab machines.
- You should have physical access (via your TigerCard) to four rooms containing such machines any time the building is open. You should have remote access to any that are booted into Linux.
- Returning students should already have accounts set up. (If you've forgotten your password, go to the ITS help desk and ask for it to be reset.)

Slide 4

More Administrivia

- Why are we using "my" book when there are books that are more textbook-like? because (1) I think it emphasizes the right things, which many textbooks don't, and (2) learning from a not-really-a-textbook and other resources should be good practice for whatever you do after you graduate.

  (I don't actually think I'm going to be able to retire on the extra royalty income — but it might be enough to finance a trip to Java City for the class?)

  Also — if you spot errors, even typos, please let me know. The first person to report any legitimate error I don't know about is eligible for extra-credit points.

What is Parallel/Distributed Computing?

- Some computational jobs are just too much for one processor — no way to get them done in reasonable time.
- For jobs done by people, what do you do when the job is too much for one person?

Slide 5

What is Parallel/Distributed Computing?

- For jobs done by people, if too much for one person you assign a team — but you have to figure out
  - How to divide up work among team members.
  - How to coordinate activities of team members.
- Same idea applies to computing — if too much for one processor, use multiple processors. Issues are similar — how to divide up work, how to coordinate.

Slide 6

Slide 7

Simple Examples

- "People job" examples:
  - Digging a hole.
  - Building a house.
  - Baby.

  What do you notice about the last one in particular?
- Computer examples:
  - Adding up a lot of numbers.
  - Computing Fibonacci numbers.

  But these don't seem too "big" . . .

Slide 8

How Much Calculating is "A Lot"?

- Examples from computational biology — how many operations per second are needed to get things done fast enough to be useful?:
  - Sequence the genome — 10^{12} ops/second (500 2-Gigahertz processors).
  - Protein/protein interactions — 10^{14} ops/second (25,000 4-Gigahertz processors).
  - Simulating whole-body response to a drug — 10^{16} operations/second (1,250,000 8-Gigahertz processors).
- (Source — Intel's former life sciences industry manager.)

Slide 9

How Much Calculating is "A Lot"?

- Simplified example — weather simulation:

  Divide earth's surface into 1-square-km cells (about 5×10^8 of them); examine from surface to 14 km out. Gives 7.5×10^9 3D cells.

  Typically need to update least five variables per cell (temperature, humidity, wind (3D), etc.). So, 37.5×10^9 updates.

  To model 24 hours in 1-minute chunks: 86400 minutes. Total of 3.24×10^{15} updates.

  Optimistically assuming 10^9 updates per second, 3.24×10^6 seconds — 900 hours.

- (Adapted from example by Dr. Eggen.)

Slide 10

What Are Some Other Hard Problems?

- Crash simulation / structural analysis.
- Oil exploration.
- Explosion simulations (why Los Alamos is interested).
- Astrophysics simulations (e.g,, Dr. Lewis's work on Saturn's rings).
- Fluid dynamics.
- "Rendering" for computer-generated animation.
- And many others ...

The Need for Speed

- Solving the same problems faster — reducing wall-clock time.
- Solving bigger problems.
- Solving problems more exactly — to get better answers, need more detail, hence more processing.

Slide 11

Can't You Just Get a Faster Computer?

- Up to a point — yes. Moore's law predicts that number of transistors on a die roughly doubles every 1.5 years. Until recently, that meant doubling processor speed and memory. (Over 30 years, that's a factor of about a million!)
- But . . .

Slide 12

Slide 13

Can't You Just Get a Faster Computer? continued

- As you know — however fast processors are, it's never fast enough, and faster is more expensive.
- Eventually we'll run into physical limitations on hardware — speed of light limits how fast we can move data along wires (in copper, light moves 9 cm in a nanosecond — one "cycle" for a 1GHz processor), other factors limit how small/fast we can make chips.
- Maybe we can switch to biological computers or quantum computers, but those are pretty big paradigm shifts ...
- In the past few years, chip makers are still able to put more transistors on a chip, but they seem to have run out of ways to exploit that to get more speed, and are instead making chips with multiple "cores".

Slide 14

"The Answer" — Parallel Computing

- Analogous to "team of people" idea — if one processor isn't fast enough, use more than one.
- Also useful when there's something "inherently parallel" about the problem — e.g., operating systems, GUI-based applications, etc.
- `http://www.top500.org` tracks fastest computers; for many years now all have been "massively parallel".

Slide 15

But I Don't Want To Solve Problems Like Those!

- What if you aren't interested in solving problems like these "grand challenge" problems, Is there still a reason to be interested in parallel computing?
- The hardware is there, and it's becoming mainstream — multicore chips, hyperthreading, etc. (The Intel person says "the chip makers can put more and more transistors on a chip, and this is the best way to use that.")

  To get best use of it for single applications, will probably need parallelism.
- Also, for some applications, thinking of them as parallel/multithreaded can lead to a solution that lets you do something useful while waiting for I/O, etc.

Slide 16

About the Course

- Can think of this course as the equivalent of PAD I for parallel (and to some extent concurrent and distributed) programming. As with PAD I, many things to learn all at once:
  - A new "box of tools" — or several boxes of tools (different languages/libraries/paradigms). Must learn syntax/functions, plus tools such as compilers and runtime systems.
  - How to use the stuff in the box of tools to solve interesting problems — from low-level "what is this syntax good for?" to algorithm design.
  - How to think about "does it work?"
  - How to think about "how fast is it?"
- Also as with PAD I, the idea will be to teach a mix of technical skills and basic concepts, with emphasis on learning by doing.

Slide 17

Minute Essay

- What are your goals for this course?
- Are you reasonably comfortable with Java and C? How about C++? (There will be assignments using both C and Java.)
- Do you have any experience already with parallel or multithreaded programming? If so, tell me about it, briefly.
- Will it be a problem for you if I assign homework that will be hard to do without access to our Linux lab machines?
- Anything else you want to tell me? about the course, about what you did over the break . . .