



• The bookstore should have two books for the course, the one that I'm a co-author of and one by Quinn. "My" book is required; the other is optional — more a traditional textbook, for those who want such a thing. I'll probably use it as a source of examples.

Slide 3

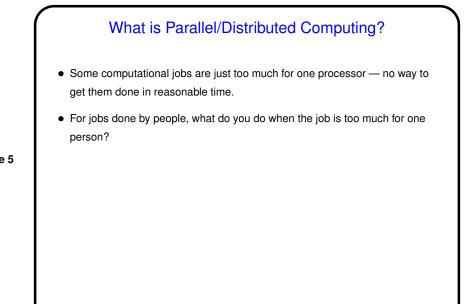
Why are we using "my" book when there are books that are more textbook-like? because (1) I think it emphasizes the right things, which many textbooks don't, and (2) learning from a not-really-a-textbook and other resources should be good practice for whatever you do after you graduate. (I don't actually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

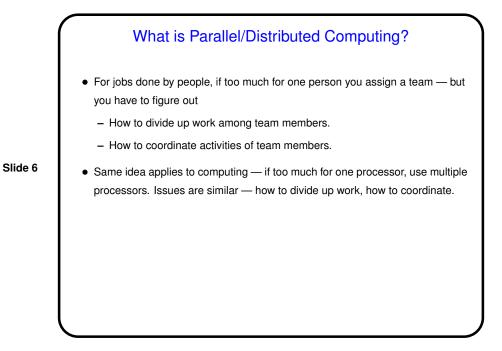
A Few Words About Computer Use in Class

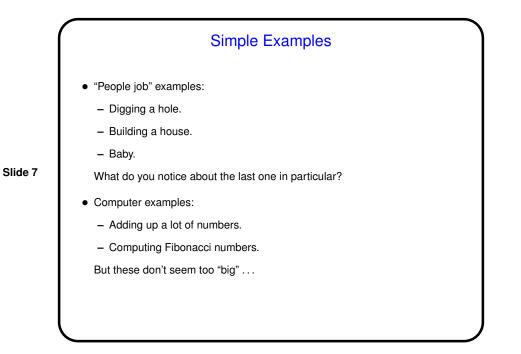
- Checking your e-mail when you first get here is okay.
- Taking notes online is okay.
- Surfing the Web or playing games during lecture is not okay.

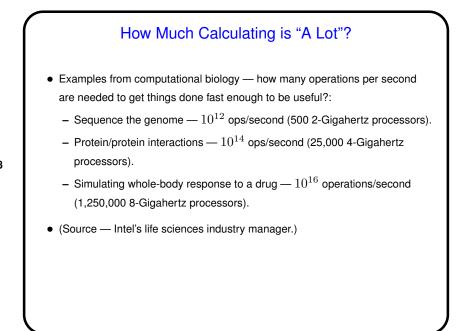
Slide 4

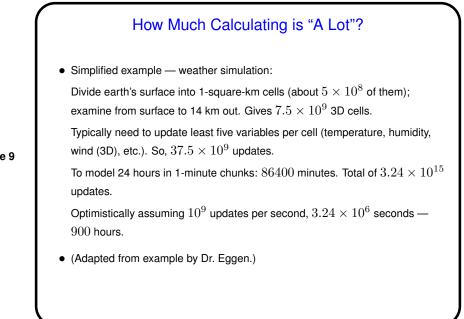
• Remember that I can lock all screens, project what's on one student's screen, etc. — and I will if need be. But I'd rather you'd all be responsible enough to resist this distraction!



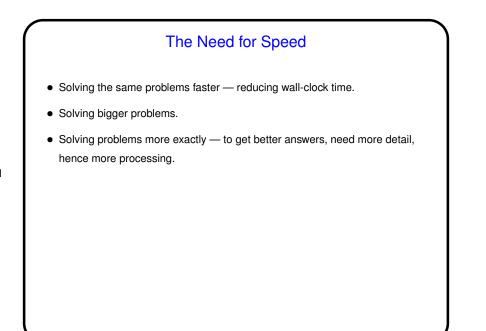






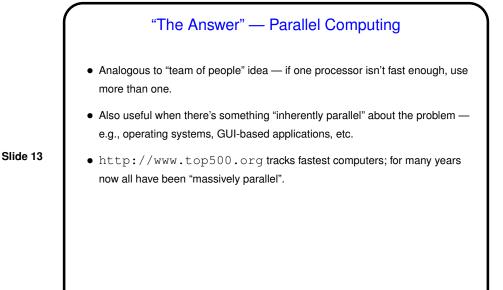


What Are Some Other Hard Problems? Crash simulation / structural analysis. Oil exploration. Explosion simulations (why Los Alamos is interested). Astrophysics simulations (e.g., Dr. Lewis's work on Saturn's rings). Fluid dynamics. "Rendering" for computer-generated animation. And many others ...



Slide 12

Can't You Just Get a Faster Computer? Up to a point — yes. Moore's law predicts that processor speed and memory both double about every 1.5 years. Over 30 years, that's a factor of about a million! But … As you know — however fast processors are, it's never fast enough. Faster is more expensive, and price/performance is not constant. 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 we can make chips. Maybe we can switch to biological computers or quantum computers, but those are pretty big paradigm shifts …



Parallel Versus Distributed Versus Concurrent • Key idea in common — more than one thing happening "at the same time". Distinctions among terms (in my opinion) not as important, but: • "Parallel" connotes processors working more or less in synch. Examples include high-end multiple-processor systems. Analogous to team of people all in the same room/building, working same hours. • "Distributed" connotes processors in different locations, not necessarily working in synch. Example is SETI@home project. Analogous to geographically distributed team of people. "Concurrent" includes apparent concurrency. Example is multitasking operating systems. Analogous to one person "multitasking". Can be useful for "hiding latency".

Hardware

- "Multiprocessors" multiple processors sharing memory. Category includes multi-CPU mainframes/servers, dual-processor PCs, "multi-core" chips. (Hyperthreading? not exactly the same, but similar.)
- "Multicomputers" multiple processor/memory units interacting by message-passing. Category includes "massively parallel" supercomputers, "Beowulf clusters", clusters/networks of workstations/PCs.
- In both categories early representatives (1980s or before) were "supercomputers" (in both performance and price), but ideas have trickled down to mainstream hardware.

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 makes 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.



- Key idea processes executing concurrently, each has its own memory, all interaction is via messages.
- Maps well onto most-common hardware platforms for large-scale parallel computing.
- Challenge for programmers is to break up the work, figure out how to get separate processes to interact by message-passing no shared memory.
- (How would the "add up a lot of numbers" example work here?)



- Key idea threads executing concurrently, all sharing one memory.
- Maps well onto hardware platforms for smaller-scale parallel computing, can be implemented on other platforms too.
- Slide 18
- Challenge for programmers is to break up the work, figure out how to get threads to interact *safely* sharing variables has its pitfalls.
- (How would the "add up a lot of numbers" example work here?)

