Slide 1

Administrivia

- About homeworks: Feedback on your work is coming. In the meantime, I will try to post sample solutions this week.
- About projects: If you haven't turned in a proposal yet, please do at your earliest convenience!

Slide 2

Minute Essays From Previous Lectures

- What people found interesting/difficult about Homework 3 (game of life):
  - Message-passing!
  - Lesson in basic program design / something that's easy in one paradigm is difficult in another.
- What people found interesting/difficult about Homework 4 (quicksort):
  - Knowing when the sort was finished.
  - Getting data to the new thread (?).
  - How quicksort works.
  - The importance of `join()`.

Slide 3

Example Application — N-Body Problem

- Many (?) problems involve computing all interactions between pairs of N bodies — the "N-body problem". (Part of our molecular dynamics example fits this model.)
- Straightforward parallelization uses *Task Parallelism*. An alternate approach, though, is based on the idea that a cluster of bodies far away can be treated as a single body (with mass the sum of the masses of the individual bodies, and position at the center of mass of the cluster). This leads to a divide-and-conquer approach . . .

Slide 4

N-Body Problem — Barnes-Hut Algorithm

- Idea of algorithm is to build a tree ("oct-tree") by repeatedly subdividing the whole space (splitting in half first in x dimension, then y, then z, then x again), discarding subdivisions with no bodies, until you get to one body per subdivision.
- Pseudocode for algorithm then looks like this:

```
loop over time steps
  build_octtree();
  compute_mass_and_center_of_gravity();
  compute_forces();
  update_pos_and_velocity();
end loop
```

with computation of mass and center of gravity, and then forces, performed using the tree and a divide-and-conquer strategy.

Slide 5

N-Body Problem — Barnes-Hut Algorithm, Continued

- All but the last step and could be parallelized using *Divide and Conquer*. Load balance might be poor, but that can be corrected by splitting in a way that gives roughly equal numbers of particles in subdivisions.
- Last step could be parallelized with *Task Parallelism*.

Slide 6

Example Application — Prefix Sum

- Problem here is to compute, for each element i of a list, the sum of that element and all elements to the left. Sounds purely sequential, right? but there is a clever *Recursive Data* solution . . . (See book, pp. 101–102.)

Slide 7

Minute Essay

- None — sign in.