Slide 1

Administrivia

- Reminder: Homework 1 due 11:59pm today. If you have questions, I should be around this afternoon.

Slide 2

Sketch of Parallel Algorithm Development (Again)

- Start with understanding of problem to be solved / application.
- Decompose computation into "tasks" — snippets of sequential code that you might be able to execute concurrently.
- Analyze tasks and data — how do tasks depend on each other? what data do they access (local to task and shared)?

  (Or start with decomposition of data and infer tasks from that.)
- Plan how to map tasks onto "units of execution" (threads/processes) and coordinate their execution. Also plan how to map these onto "processing elements".
- Translate this design into code.
- Our book organizes all of this into four "design spaces". For this course, we'll start at the bottom and work up, so we can start writing code now!

Slide 3

Example — Numerical Integration

- Compute π by integrating $\int_0^1 \frac{4}{1+x^2}\,dx$.
- Do this numerically by approximating area under curve by many small rectangles, computing their area, adding results.
- Sequential program fairly straightforward. (`num-int-seq.c` on "sample programs" page).
- "Parallelize" how? (`num-int-par.c` on "sample programs" page).

Slide 4

Numerical Integration, Continued

- In general, we hope that "parallelizing" a sequential program doesn't change its output, except maybe for execution time. Here, output might vary depending on number of processes and details of what `MPI_Reduce` does, since floating-point addition is not associative.
- Execution time seems to scale with number of processes (speedup approximately equal to number of processes), up to a point — good result.

Slide 5

Minute Essay

- What do you think you learned from Homework 1, or as much of it as you've done so far?
- Any more questions about MPI?