Slide 1

Administrivia

- Homework 1 will be on Web later today. Due next Tuesday. Simple MPI program.

Slide 2

Minute Essay From Last Lecture

- Question: Look again at the simple send/receive program. What do you think would happen (i.e., what would print) if right after the call to `MPI_Send` we changed one or both elements of `buff`?
- Answer?

Slide 3

Timing MPI Programs

- "How long did it take?" often of interest. Can use system tools (e.g., `time` command) to check total elapsed time. Or can time "interesting" parts of program:

  `MPI_Wtime` returns elapsed time; call twice and subtract to find out how long something takes (`time_msg.c` on "sample programs" page).
- How meaningful output is depends — e.g., on whether the system is otherwise idle. Probably best to repeat observations a few times, and do some sort of averaging.

Slide 4

Simple (Blocking) Point-to-Point Communication in MPI

- Send with `MPI_Send` — returns as soon as data has been copied to system buffer, buffer in program can be reused.
- Receive with `MPI_Recv` — waits until message has been received.
- Can use "tags" to distinguish between kinds of messages. Can receive selectively or not (`MPI_ANY_TAG`). Received tag is in returned `MPI_Status` variable (e.g., `status.MPI_TAG`).
- Can receive from specific sender or from any sender. (`MPI_ANY_SOURCE`). Sender is in returned `MPI_Status` variable (e.g., `status.MPI_SOURCE`).
- For `MPI_Recv`, "length" parameter specifies buffer length. Use `MPI_Get_count` to get actual count.

Slide 5

Not-So-Simple Point-to-Point Communication in MPI

- For not-too-long messages and when readability is more important than performance, `MPI_Send` and `MPI_Recv` are probably fine.
- If messages are long, however, buffering can be a problem, and can even lead to deadlock. Also, sometimes it's nice to be able to overlap computation and communication.
- Therefore, MPI offers several other kinds of send/receive functions — "synchronous" (blocks both sender and receiver until communication can take place), "non-blocking" (doesn't block at all, program must later test/wait for communication to take place).

Slide 6

Collective Communication in MPI

- "Collective communication" operation — one that involves many processes (typically all, or all in MPI "communicator").
- Could implement using point-to-point message passing, but some operations are common enough to be library functions — broadcast (`MPI_Bcast`), "reduction" (`MPI_Reduce`), etc.

Slide 7

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 8

Minute Essay

- Any questions so far about MPI?