Slide 1

Administrivia

- (None?)

Slide 2

Minute Essay From Last Lecture

- More people were using digital versions than print; a few said they were renting from the bookstore (“hm!”?).
- Most are at least doing *some* reading and finding it at least a bit helpful (some range there).

Slide 3

Example Application — Generic Master/Worker Program

- As an illustration of the *Master/Worker* program-structure pattern, try writing a sort of mock-up of such a program, in which tasks are represented by "sleeps" of varying lengths.
- Sequential code just generates some number of fake tasks with varying times generated using `rand()`. (Look at code.)

Slide 4

Generic Master/Worker Program — OpenMP

- Parallelizing sequential code with OpenMP is fairly straightforward:
- We don't need an explicit master thread because all it would do is assign tasks to threads, and we can get that with `omp parallel for`. Here we might want to try both static and dynamic scheduling.
- (Look at code, and notice additions to also show how tasks were distributed among threads. Also notice use of `#omp critical` to avoid potential race conditions with calls to `rand()`. This would not be a good strategy in an application where those calls were a big contributor to overall program runtime, but here they're probably not.)

Slide 5

Generic Master/Worker Program — MPI

- Parallelizing sequential code with MPI is less straightforward:
- For static scheduling, we don't need an explicit master; we can easily have each process pick out "its" tasks.
- For dynamic scheduling, it does seem like we need an explicit master, so have one process serve in that role, with a defined protocol for master/worker interaction:
  - Each worker process repeated requests a task from the master, receives one, and executes it, continuing until it gets a task meaning "no more".
  - The master process repeatedly receives requests for a task from workers, responds to it, and records results, until all tasks are complete. It then sends each worker a "no more" task.

Slide 6

Generic Master/Worker Program — MPI, Continued

- (Look at code, and notice additions to also show how tasks were distributed among processes. Also notice that the static-distribution version just generates the whole sequence of tasks in each process and then only executes some of them. This would not be a good strategy in an application where generating the tasks was a big contributor to overall program runtime, but here it's probably not.)

Minute Essay

- None really — sign in.