Slide 1

Administrivia

- Reminder: Homework 3 due today. If you can't finish by midnight, *send me what you have*, and send a final version later.
- Homework 4 to be on the Web soon (tomorrow?). I will send mail.

Slide 2

Minute Essay From Last Lecture

- People who reported progress generally seemed to be still having trouble with the MPI program. It's definitely the hard part of this assignment!

Slide 3

Example Application — Mergesort

- Mergesort should be familiar from other courses. Sequential algorithm is divide-and-conquer, and solution of subproblems is independent, so:
- We could pretty much skip the whole *Finding Concurrency* step and go directly to . . .
- *Algorithm Structure* pattern *Divide and Conquer* seems to fit.

Slide 4

Mergesort, Continued

- One important consideration is whether to every call to the recursive function should be a task. Probably not — way too much overhead.
- For this problem, at each level both subproblems are roughly the same size, so what probably makes sense is to have at most one task per UE. (If the subproblems were of different sizes, we'd want to consider having more tasks and mapping them to UEs in a way that would produce good load balance.)
- Look at code . . .

Slide 5

Example Application: Matrix Multiplication

- Basic problem is straightforward: For two N by N matrices A and B, compute the matrix product C with elements defined thus (assuming 0-based indexing):

$$c_{i,j} = \sum_{k=0}^{N-1} a_{i,k} \cdot b_{k,j}$$

  (Actually A and B don't have to be square and the same size, but for the moment let's assume they are.)

- Simple approach to calculating this is obvious — just do the above calculation for all i and j between 0 and $N-1$.

- Less obvious approach: Decompose A, B, and C into blocks and think of the calculation in terms of these blocks (equation similar to the above, but for blocks rather than individual elements).

  Why? often makes better use of cache and therefore is faster.

Slide 6

Parallelization — Understanding the Problem

- In the simple approach, the code is just nested loops over the elements of C. A block-based approach is slightly more complicated, but not a great deal.

- Consider parallelizing for first shared-memory and then distributed-memory environments.

- (To be continued next time.)

Minute Essay

- Anything noteworthy to report about Homework 3?