

Slide 1

Recap — Algorithm Structure Patterns

- If decomposition/analysis reveals organization in terms of tasks *Task Parallelism* (probably most common strategy) or *Divide and Conquer*.
- If decomposition/analysis reveals organization in terms of data *Geometric Decomposition* (second most common strategy) or *Recursive Data*.
- Slide 2
- If organization is in terms of flow of data (*Pipeline* and *Event-Based Coordination*).
- Last time we talked briefly about *Task Parallelism* and *Geometric Decomposition*. Review, and look at the other four. (Notice the common "pattern format" problem statement, solution, key ideas.)

Task Parallelism

• Problem statement:

When the problem is best decomposed into a collection of tasks that can execute concurrently, how can this concurrency be exploited efficiently?

• Key ideas in solution — managing tasks (getting them all scheduled), detecting termination, managing any data dependencies.

Slide 3

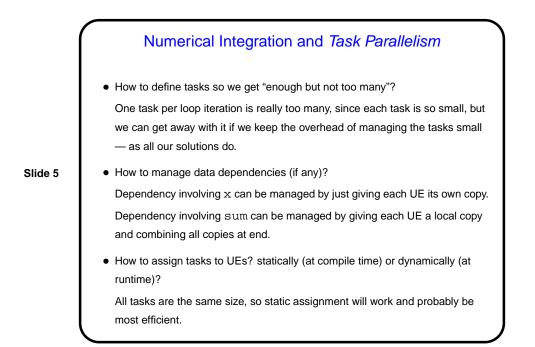
- Many, many examples, including:
 - Numerical integration example (next slide).
 - Molecular dynamics example (after that).
 - Mandelbrot set computation.
 - Branch-and-bound computations: Maintain list of "solution spaces". At each step, pick item from list, examine it, and either declare it a solution, discard it, or divide it into smaller spaces and put them back on list. Tasks consist of processing items from list.

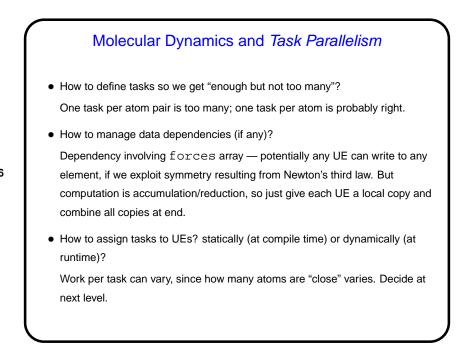


- A task decomposition probably makes sense here, with the tasks being the iterations of the main loop.
- There's only one group of tasks, and the tasks in the group can execute concurrently.

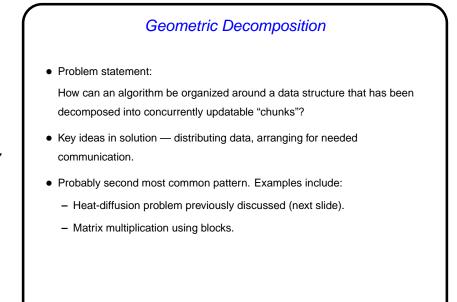
Slide 4

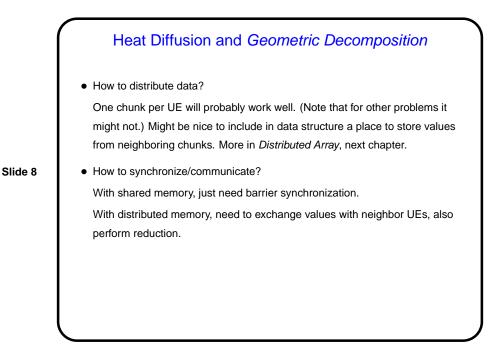
• Data shared among tasks includes a read-only variable (step), a variable that could be made task-local (x), and an "accumulate data" variable (sum).



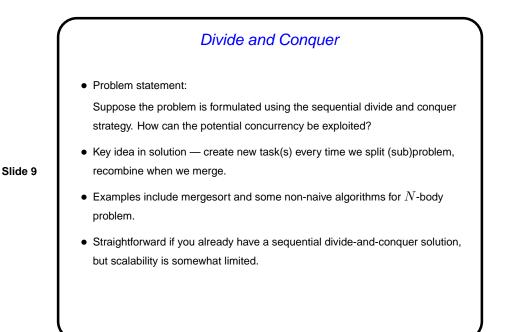


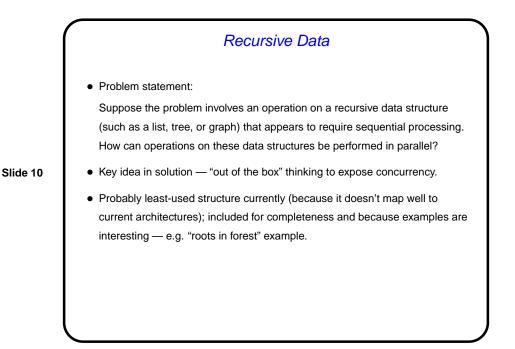
Slide 6

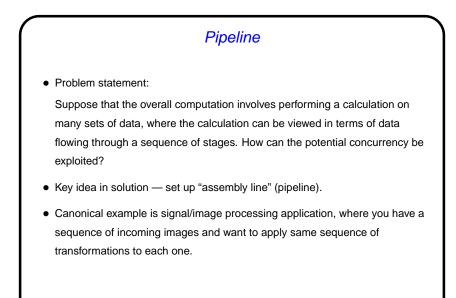




Slide 7







Event-Based Coordination

• Problem statement:

Suppose the application can be decomposed into groups of semi-independent tasks interacting in an irregular fashion. The interaction is determined by the flow of data between them which implies ordering constraints between the tasks. How can these tasks and their interaction be implemented so they can execute concurrently?

- Key idea in solution structure computation in terms of semi-independent entities, interacting via "events".
- Canonical example is discrete event simulation simulating many semi-independent entities that interact in irregular/unpredictable ways.

Slide 11

Slide 12

