Slide 1

Administrivia

- Homework 3 to be on Web today or tomorrow. Due next week. Basic idea is to improve your programs from Homework 2.
- Information about projects coming soon.

Slide 2

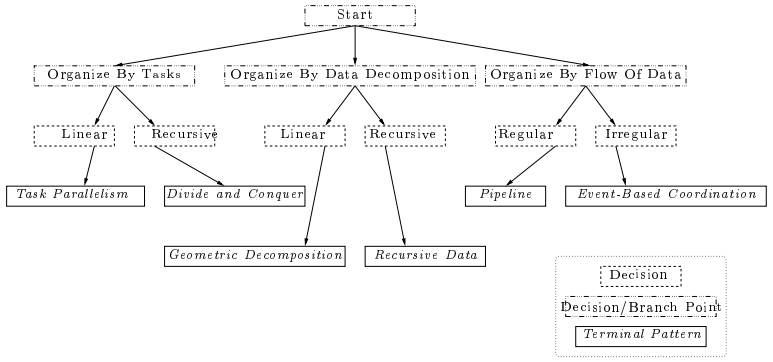
Algorithm Structure Design Space

- Historical note: These are the patterns with the longest history. We started out trying to identify commonly-used overall structures for parallel programs (these patterns), and then at some point added the other "design spaces".
- After much thought, writing, revision, and arguing, we came up with . . .

Slide 3

Algorithm Structure Decision Tree

(Figure 4.2):



Slide 4

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.

- Many, many examples, including:
  - Molecular dynamics example (next slide).
  - 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.

Slide 5

Molecular Dynamics and *Task Parallelism*

- How to define tasks so we get "enough but not too many"?

  One task per atom pair is too many; one task per atom is probably right.

- How to manage data dependencies (if any)?

  Dependency involving `forces` array — potentially any UE can write to any element, if we exploit symmetry resulting from Newton's third law. But computation is accumulation/reduction, so just give each UE a local copy and combine all copies at end.

- How to assign tasks to UEs? statically (at compile time) or dynamically (at runtime)?

  Work per task can vary, since how many atoms are "close" varies. Decide at next level.

Slide 6

Geometric Decomposition

- Problem statement:

  How can an algorithm be organized around a data structure that has been decomposed into concurrently updatable "chunks"?

- Key ideas in solution — distributing data, arranging for needed communication.

- Probably second most common pattern. Examples include:
  - Heat-diffusion problem previously discussed (next slide).
  - Matrix multiplication using blocks.

Slide 7

Heat Diffusion and *Geometric Decomposition*

- How to distribute data?

  One chunk per UE will probably work well. (Note that for other problems it might not.) Might be nice to include in data structure a place to store values from neighboring chunks. More in *Distributed Array*, next chapter.

- How to synchronize/communicate?

  With shared memory, just need barrier synchronization.

  With distributed memory, need to exchange values with neighbor UEs, also perform reduction.

Slide 8

Minute Essay

- None — sign in.