Lecture Topics

- dimensions for parallel models (review + wrap-up)
- a few basic concepts

Administrivia

- …

Dimensions for Parallel Models

- static/dynamic number of execution contexts
- implicit vs. explicit parallelism (a range of possibilities)
- asynchronous/synchronous sharing
  - synchronous models
    - each shared datum has an owning thread
    - owner thread must allow access with API/language
    - segments of code in owner thread
      - are implicitly atomic
      - with respect to all accesses to thread's shared data by other threads
  - asynchronous models
    - do not have such a property
    - no owner thread for data
    - or other threads can access asynchronously with respect to owner thread
  - software runtime can be used to move between these two
- (last dimension…hardware)
- and a hardware dimension that interacts with sharing control:
  - shared memory: processors share a common pool of memory
  - distributed memory: processors have private pools of memory
  - many variations/combinations, but let's start simple
  - software can be used to masquerade here, too

A Few Basic Concepts

- high-performance computing definitions…

- parallel speedup, or just "speedup"
  - When I run my program in parallel
    - it finishes X times faster than when I run it sequentially
    - X = T(sequential) / T(parallel)
  - X is the speedup
  - finding T(sequential)
    - best algorithm for sequential machine
    - optimized for sequential machine
    - no parallelism support remnants
- note that speedup assumes a fixed problem size

- parallel efficiency, or just "efficiency"
  - How well am I using my parallel resources?
  - efficiency on P processors = speedup on P processors / P

- scalability
  - For how many processors is speedup linear, or is efficiency flat?
  - at some value of P, with fixed problem size
    - speedup will flatten out
    - later it will drop (unless you leave processors idle)
  - "good" scalability means
    - no falloff on your machine
    - maximum measurable value of P

- other variants of speedup
  - see J. P. Singh, J. L. Hennessy, A. Gupta, “Scaling Parallel Programs for Multiprocessors: Methodology and Examples,” *IEEE Computer*, 26(7):42-50, July 1993
  - fixed size per node (scaled speedup)
    - using parallel system to run bigger problems, so why use measure fixed size?
    - patently false for most applications at some scale
  - memory constrained speedup
    - biggest problem that fits in memory
    - looks really bad unless the problem runs in O(N)
  - time constrained speedup
    - biggest problem that finishes by the time I return from lunch
    - sometimes reasonable…
    - …but we could wait overnight for a grand challenge application?

- parallel grain size (work per parallel task)
  - each possible source of parallelism has “natural” grain size
    - loop body
    - objects in a container
    - rows/columns/blocks in a matrix
    - elements in a matrix
    - graph nodes/connected components
  - some may exhibit higher variance than others
    - conditionals/inner loops in loop body
    - complex per-object methods
    - rows in upper/lower diagonal matrix
    - matrix elements usually roughly constant
    - degree of nodes, size of connected components