Introduction

- Class is an introduction to parallel computing
  - topics include: hardware, applications, compilers, system software, and tools
- Will count for Masters/PhD Comp Credit
- Work required
  - small programming assignment
  - midterm
  - classroom participation
  - project
- Reading for the lecture: Chapter 1

What is Parallel Computing?

- Does it include:
  - super-scalar processing (more than one insn at once)?
  - client/server computing?
    - what if RPC calls are non-blocking?
  - vector processing (same instruction to several values)?
  - collection of PC's **not** connected to a network?
- For this class, parallel computing is:
  - a collection of processing elements (more than one).
  - connected to a communication network.
  - working together to solve a single problem.

Why Parallelism

- Speed
  - need to get results faster than possible with sequential
    - a weather forecast that is late is useless
  - could come from
    - more processing elements (P.E.)
    - more memory size
    - more disks
- Cost: cheaper to buy many smaller machines
  - this is only recently true due to
    - VLSI
    - commodity parts

What Does a Parallel Computer Look Like?

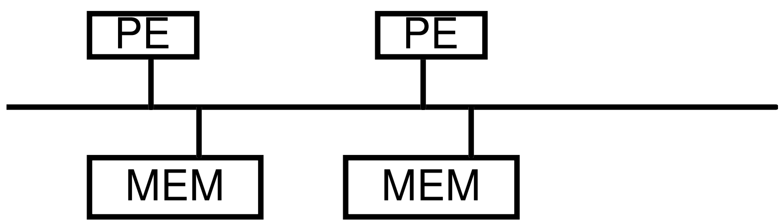
- Hardware
  - processors
  - communication
  - memory
  - coordination
- Software
  - languages
  - operating systems
  - programming models

Processing Elements (PE)

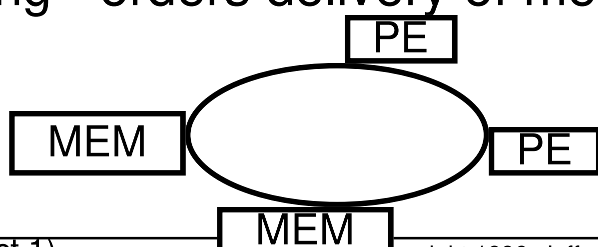
- Key Processor Choices
  - How many?
  - How powerful?
  - Custom or off-the-shelf?
- Major Styles of Parallel Computing
  - SIMD - Single Instruction Multiple Data
    - one master program counter
  - MIMD - Multiple Instruction Multiple Data
    - separate code for each processor
  - SPMD - Single Program Multiple Data
    - same code on each processor, separate PC's on each
  - Dataflow - instruction waits for operands
    - "automatically" finds parallelism

Communication Networks

- Connect
  - PE's, memory, I/O
- Key Performance Issues
  - latency: time for first byte
  - throughput: average bytes/second
- Possible Topologies
  - bus - simple, but doesn't scale

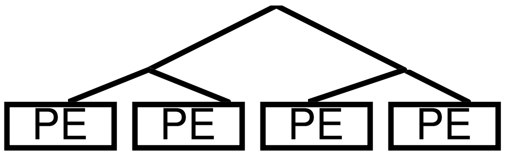


  - ring - orders delivery of messages

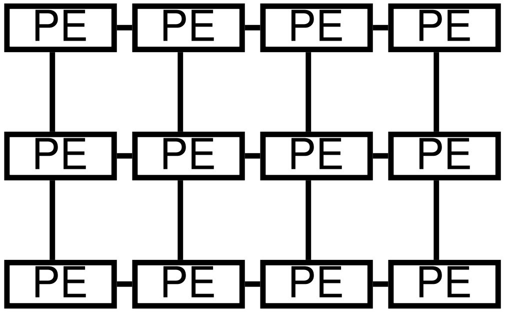


Topologies (cont)

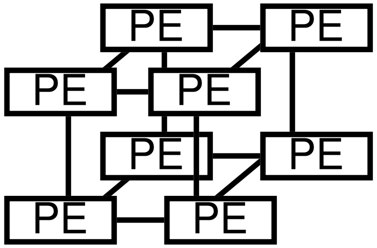
– tree - needs to increase bandwidth near the top



– mesh - two or three dimensions



– hypercube - needs a power of number of nodes



Memory Systems

- Key Performance Issues
  - latency: time for first byte
  - throughput: average bytes/second
- Design Issues
  - Where is the memory
    - divided among each node
    - centrally located (on communication network)
  - Access by processors
    - can all processors get to all memory?
    - is the access time uniform?

Coordination

- Synchronization
  - protection of a single object (locks)
  - coordination of processors (barriers)
- Size of a unit of work by a processor
  - need to manage two issues
    - load balance - processors have equal work
    - coordination overhead - communication and sync.
  - often called “grain” size - large grain vs. fine grain

Sources of Parallelism

- Statements
  - called “control parallel”
  - can perform a series of steps in parallel
  - basis of dataflow computers
- Loops
  - called “data parallel”
  - most common source of parallelism
  - each processor gets one (or more) iterations to perform