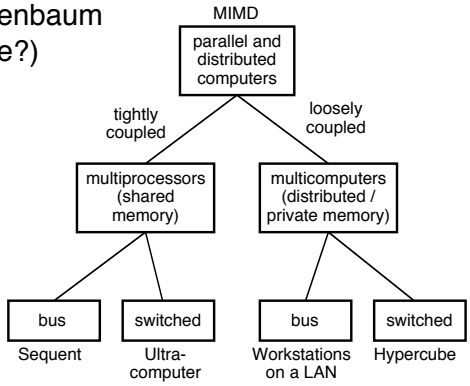
SIMDs & MIMDs — Parallel or Distributed Architecture? (Review)

- Tanenbaum (date?)



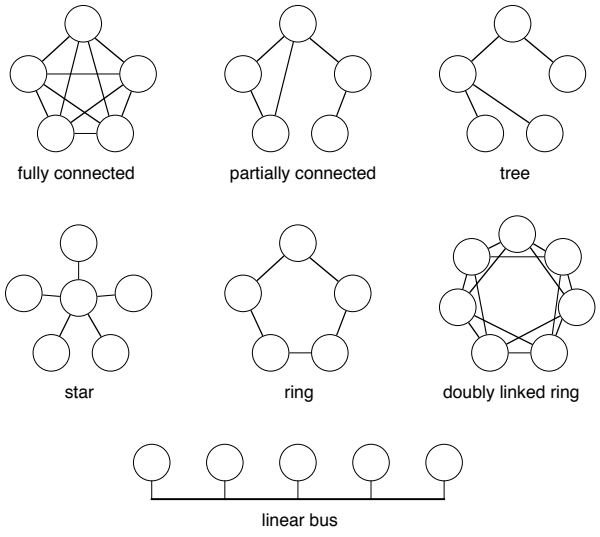
- Few (tens) of powerful commercial RISC processors, connected in some way
  - One cabinet vs. multiple cabinets
  - Distributed memory vs. shared memory
  - Many interconnection alternatives

Processor Interconnection

- MIMDs (as well as SIMDs) may interconnect processors in several ways:
  - Switching network
  - LAN (local area network)
- Switching network
  - Grid
    - n^2 nodes arranged as an n x n grid
    - ✘ Maximum route proportional to $2n$
    - ✘ Most messages take multiple hops
  - Hypercube
    - A n-degree hypercube (n-cube) consists of 2^n processors arranged in an n-dimensional cube, where each processor is connected to n other processors
    - ✓ Maximum route proportional to n
    - ✘ Most messages take multiple hops
  - Usually considered a parallel architecture

Processor Interconnection (cont.)

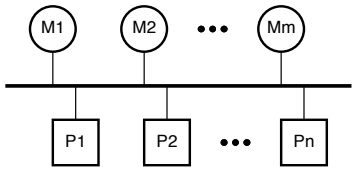
- LAN-based interconnection



- Parallel or distributed architecture?
  - Depends on the OS!

Multiprocessors — Connecting Processors to Memories

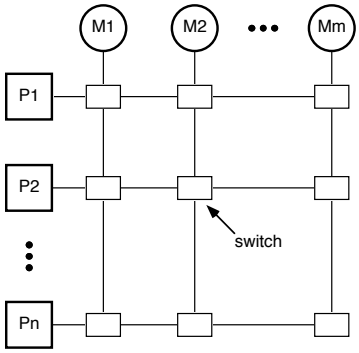
- Multiprocessors use a variety of methods to efficiently connect the processors to the memories:
  - Bus
  - Switch (crossbar, multistage switch)
- Bus-based interconnection



  - ✔ Simple
  - ✔ No need for routing — just broadcast
  - ✘ Contention for access to bus (does not scale well)
  - ✘ Limited to 32, maybe 64, processors

Multiprocessors — Connecting Processors to Memories (cont.)

- Crossbar switch



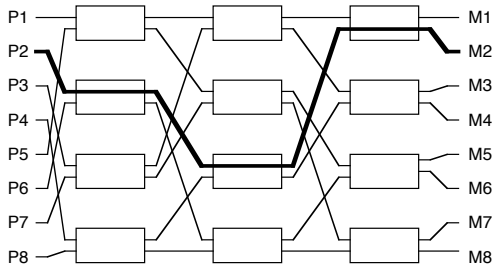
✔ Usually no contention for memory access — multiple memories can be accessed in parallel

✔ Simple routing

✘ Number of crossbar switches grows quadratically

Multiprocessors — Connecting Processors to Memories (cont.)

- Multistage switch



✔ Reduced number of switches

✘ Increased communication delay

✘ Increased contention for memory access

✘ Complex network

Classification of Multiprocessors and Multicomputers, Based on Memory Access

- Multiprocessors (shared memory)
  - UMA — Uniform Memory Access
    - Main memory is at a central location
  - NUMA — Non-Uniform Memory Access
    - Main memory is physically partitioned, each processor assigned to a partition
    - One unified memory space, but access to local partition is fast, while access to remote partitions is slow
  - Usually considered a parallel architecture
- Multicomputers (distributed memory)
  - NORMA — No Remote Memory Access
    - Main memory is physically partitioned, with each partition attached to a different processor
    - A processor can not access the memory of another processor
  - Usually considered a distributed architecture

Classification of Operating Systems (For Multicomputers on a LAN)

- Network Operating System
  - Loosely-coupled software (running on loosely-coupled hardware)
    - Each computer runs its own OS
    - User knows which machine he/she is on
  - Goal: share network resources (printer, file system, etc.)
- "True" Distributed Operating System
  - Tightly-coupled software (running on loosely-coupled hardware)
    - One single OS, or at least the feel of one
    - User may not know what processor is processing his or her job
  - OS responsible for distributing load among the processors
  - OS responsible for providing mutual exclusion, deadlock handling, etc.

Distributed System Models

- Minicomputer model
  - Several minicomputers connected to a network, each with several terminals
- Workstation-server model
  - Specialized workstations running servers: file server, print server, etc.
  - Good resource sharing (printers, etc.), cheap workstations (don't need big disks)
- Workstation model
  - Many workstations on a network, but one workstation per user
  - System automatically migrates processes to idle workstations
- Processor-pool model
  - Terminals connect to pool of processors allocated to users as necessary

Goals of a Distributed System: Transparency

- *Access* transparency
  - User is unaware whether a resource is local or remote
- *Location* transparency
  - User is unaware of physical location of hardware or software resources
- *Migration* transparency
  - User is unaware if OS moves processes or resources (e.g., files) move to a different physical locations
- *Replication* transparency
  - Resource duplication is invisible to users
- *Concurrency* transparency
  - Resource sharing is invisible to users

Other Goals of a Distributed System

- *Performance*
  - Cache data, minimize copying data, minimize network traffic, use threads to take advantage of parallelism
- *Scalability*
  - System should adapt well to increased load (avoid central control, do as much work locally as possible)
- *Flexibility*
  - System should be easy to modify and enhance (use microkernel and user-level processes providing services)
- *Reliability*
  - Must avoid, tolerate, detect, and recover from *faults* — mechanical or algorithmic defects that can generate an error

Why Use Distributed Systems? What are the Advantages?

- Natural programming model
  - Some applications (database in large company) are inherently distributed
- Resource sharing
  - Expensive (scarce) resources need not be replicated for each processor
- Price / performance
  - Network of workstations provides more MIPS for less $ than a mainframe does
- Reliability
  - Replication of processors and resources yields fault tolerance
- Scalability
  - Modular structure makes it easier to add or replace processors and resources