Parallel Processors

Forecast

- Motivation
- Connect Processors At
    - Processor
    - I/O
    - Memory
- Symmetric Multiprocessors & Cache Coherence
- Taxonomies & Examples

Motivation

So far we have had one processor in a computer system

Why not use **N processors** to get

- **Higher throughput** via many jobs in parallel
- **Improved cost-effectiveness** (e.g., adding 3 processors may quadruple throughput for twice the system cost)
- To get **lower latency** from shrink-wrapped software (e.g., databases & web servers today, but more tomorrow)
- Lower latency through **parallelizing** your application (but this is not easy)

Where Should Processors Be Connected?

At Processor

- E.g., Single-Instruction Multiple-Data (SIMD)

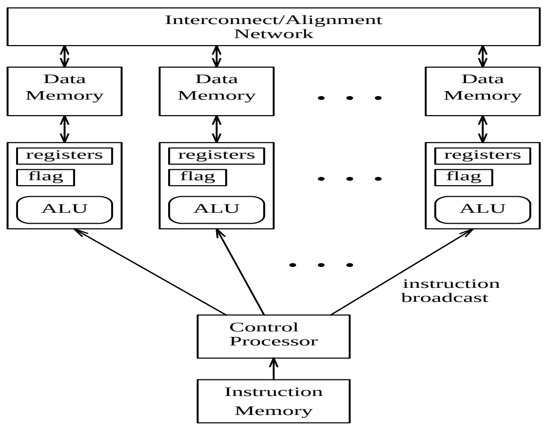
At I/O System

- Clusters or Multicomputers

At Memory System

- Shared-Memory Multiprocessors
- Especially Symmetric Multiprocessors (SMPs)

Connect at Processor: SIMD



Connect at Processor

SIMD Assessment

+ Amortizes cost of control unit over many datapaths

– Does not leverage current microprocessors

– Programming model has limited flexibility
(compare w/ Multiple-Instruction Multiple-Data (**MIMD**))

Other Designs that join at processor

Systolic, etc., have not been commercial success

Connect at I/O

Connect with Standard Network (e.g., with Ethernet)

- Called a **Cluster** (e.g, Beowulf Cluster)
- Adequate bandwidth, but high-latency communication
- Cheap, but “you get what you pay for”

Connect with Custom Network (e.g., IBM SP/[1-3])

- Sometimes called a **Multicomputer**
- Higher cost than cluster & poorer communication than multiprocessors (coming next)

Connect at Memory: Multiprocessors

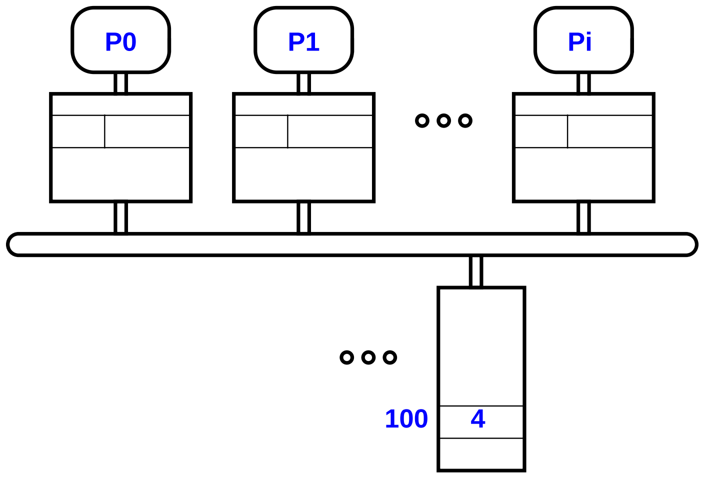
Shared Memory Multiprocessors

- All processors can address all physical memory
- Demands evolutionary operating systems changes
- Higher throughput with no application changes
- Lower-latency, however, requires parallelization with synchronization

Most Successful: **Symmetric Multiprocessor (SMP)**

- 4-32 microprocessors on a bus
- Too much bus traffic so add caches

Symmetric Multiprocessor (SMP)



Cache Coherence

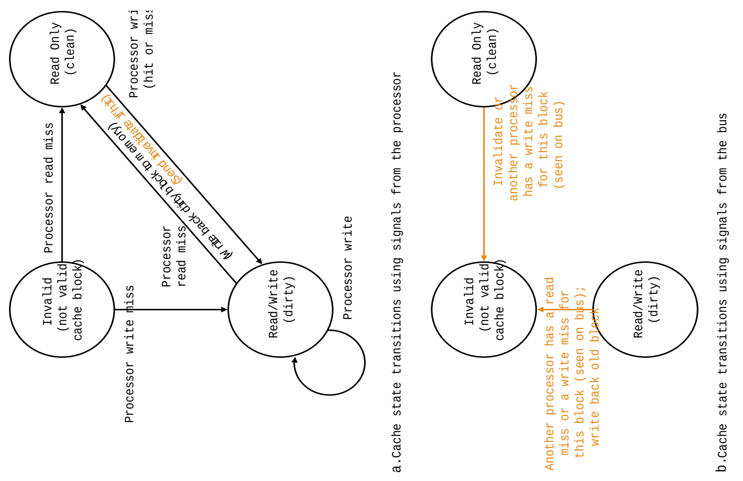
Snooping in SMP

- All requests broadcast on bus
- All processors & memory snoop & respond appropriately
- Cache blocks read/write at one processor or read-only at several

Larger Multiprocessors

- Eschew bus
- Use "level of indirection" through **directory**

Coherence Protocol (Fig. 9.5)


Read Only (clean)
Invalid (not valid cache block)
Read/Write (dirty)
Processor read miss
Processor write (hit or miss)
(Send invalidate if hit)
(Write back dirty block to memory)
Processor read miss
Processor write miss
Processor write
a. Cache state transitions using signals from the processor
Read Only (clean)
Invalid (not valid cache block)
Read/Write (dirty)
Invalidate or another processor has a write miss for this block (seen on bus)
Another processor has a read miss or a write miss for this block (seen on bus); write back old block
b. Cache state transitions using signals from the bus

Taxonomies

In General [Flynn 1966]

	single data	**multiple data**
single instrn	SISD	SIMD
multiple instrn	MISD	MIMD

For Shared Memory Multiprocessors

	uniform memory access	**non-uniform memory access**
cache coherence	CC-UMA	CC-NUMA
no cache coherence	NCC-UMA	NCC-NUMA

Example Commercial Systems

CC-UMA (SMP)

- Sun E10000: http://computer.org/micro/mi1998/m1039abs.htm

CC-NUMA

- SGI Origin 2000: http://computer.org/micro/mi1998/m1039abs.htm

NCC-NUMA

- Cray T3E: http://www.cs.wisc.edu/~markhill/Misc/asplos96_t3e_comm.pdf

Clusters

- ASCI: http://www.llnl.gov/asci/platforms