

William Stallings

Computer Organization and Architecture

Chapter 16

Parallel Processing

Multiple Processor Organization

- ⌘ Single instruction, single data stream - SISD
- ⌘ Single instruction, multiple data stream - SIMD
- ⌘ Multiple instruction, single data stream - MISD
- ⌘ Multiple instruction, multiple data stream- MIMD

Single Instruction, Single Data Stream - SISD

- ⌘ Single processor
- ⌘ Single instruction stream
- ⌘ Data stored in single memory
- ⌘ Uni-processor

Single Instruction, Multiple Data Stream - SIMD

- ⌘ Single machine instruction
- ⌘ Controls simultaneous execution
- ⌘ Number of processing elements
- ⌘ Lockstep basis
- ⌘ Each processing element has associated data memory
- ⌘ Each instruction executed on different set of data by different processors
- ⌘ Vector and array processors

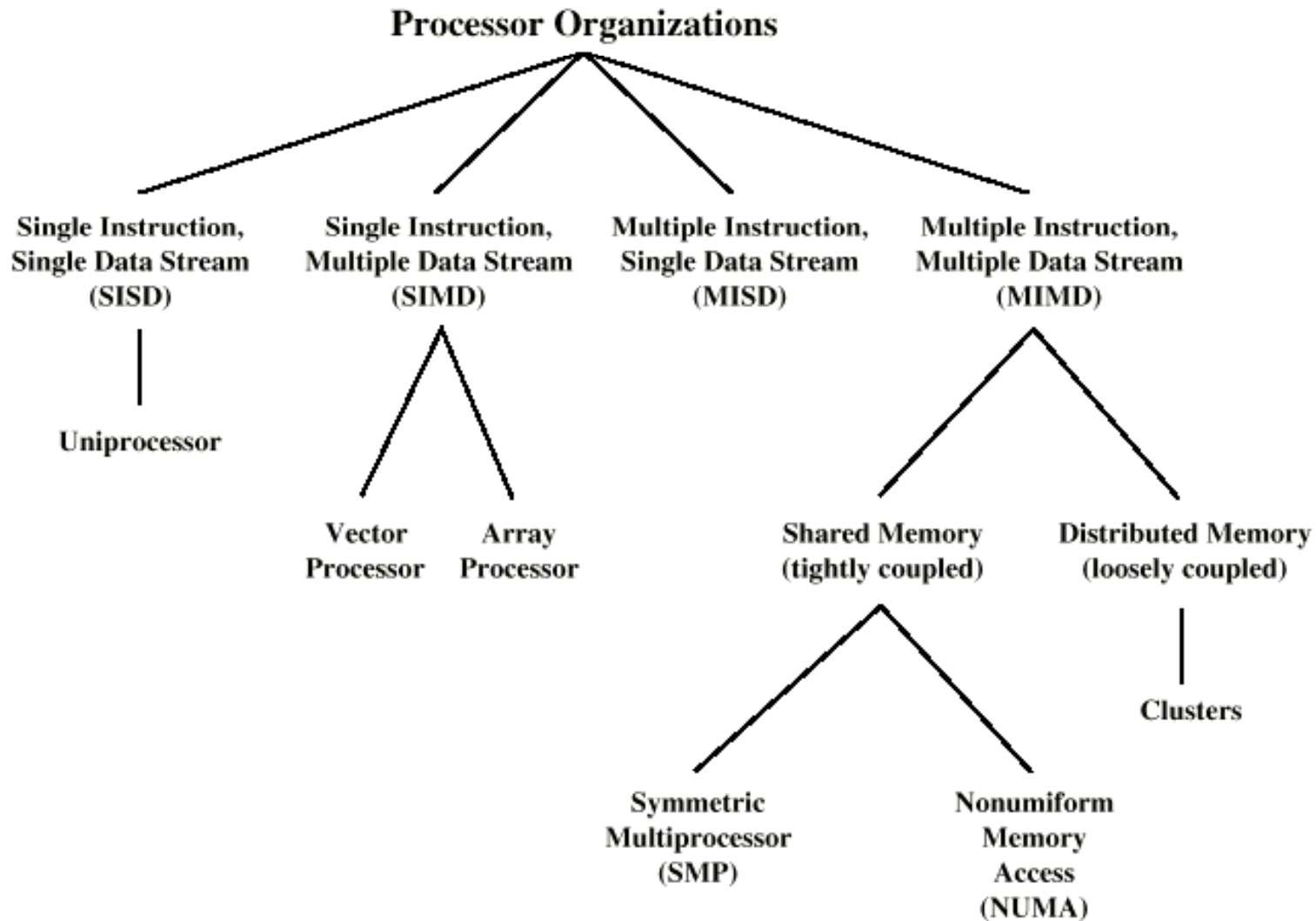
Multiple Instruction, Single Data Stream - MISD

- ⌘ Sequence of data
- ⌘ Transmitted to set of processors
- ⌘ Each processor executes different instruction sequence
- ⌘ Never been implemented

Multiple Instruction, Multiple Data Stream- MIMD

- ⌘ Set of processors
- ⌘ Simultaneously execute different instruction sequences
- ⌘ Different sets of data
- ⌘ SMPs, clusters and NUMA systems

Taxonomy of Parallel Processor Architectures



MIMD - Overview

- ⌘ General purpose processors
- ⌘ Each can process all instructions necessary
- ⌘ Further classified by method of processor communication

Tightly Coupled - SMP

- ⌘ Processors share memory
- ⌘ Communicate via that shared memory
- ⌘ Symmetric Multiprocessor (SMP)
 - ☑ Share single memory or pool
 - ☑ Shared bus to access memory
 - ☑ Memory access time to given area of memory is approximately the same for each processor

Tightly Coupled - NUMA

- ⌘ Nonuniform memory access
- ⌘ Access times to different regions of memory may differ

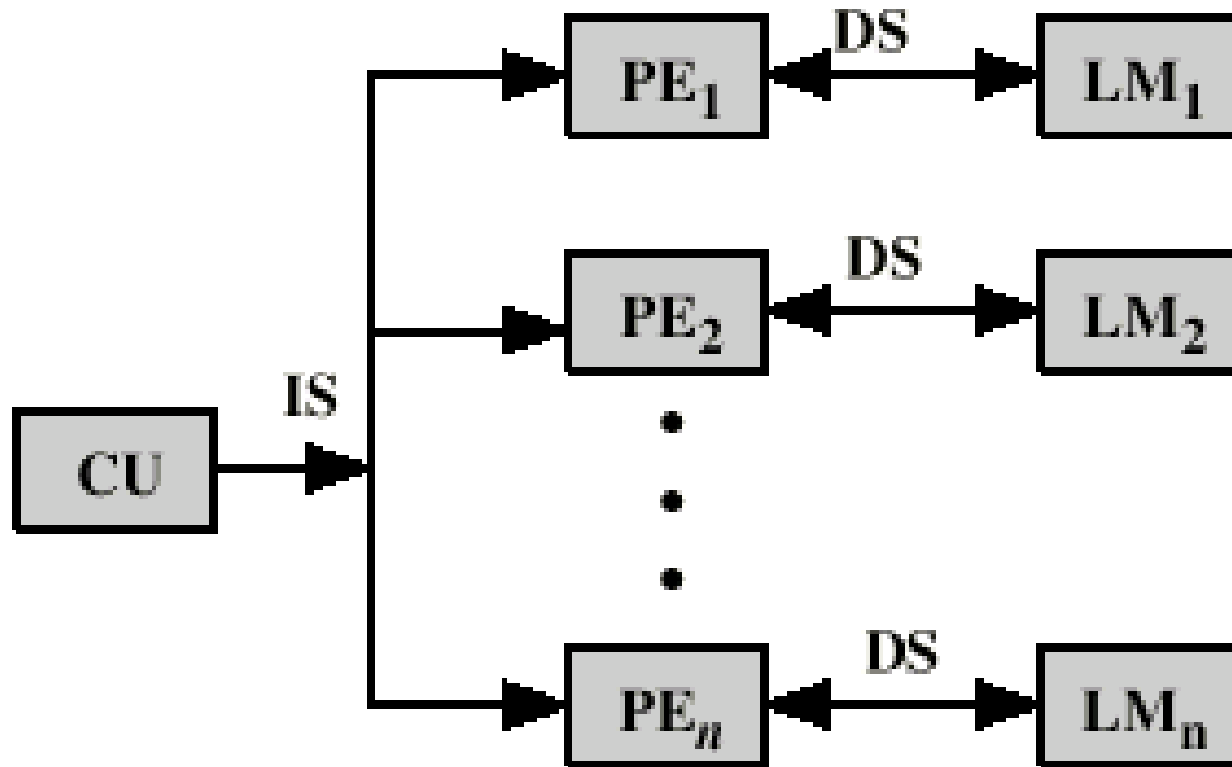
Loosely Coupled - Clusters

- ⌘ Collection of independent uniprocessors or SMPs
- ⌘ Interconnected to form a cluster
- ⌘ Communication via fixed path or network connections

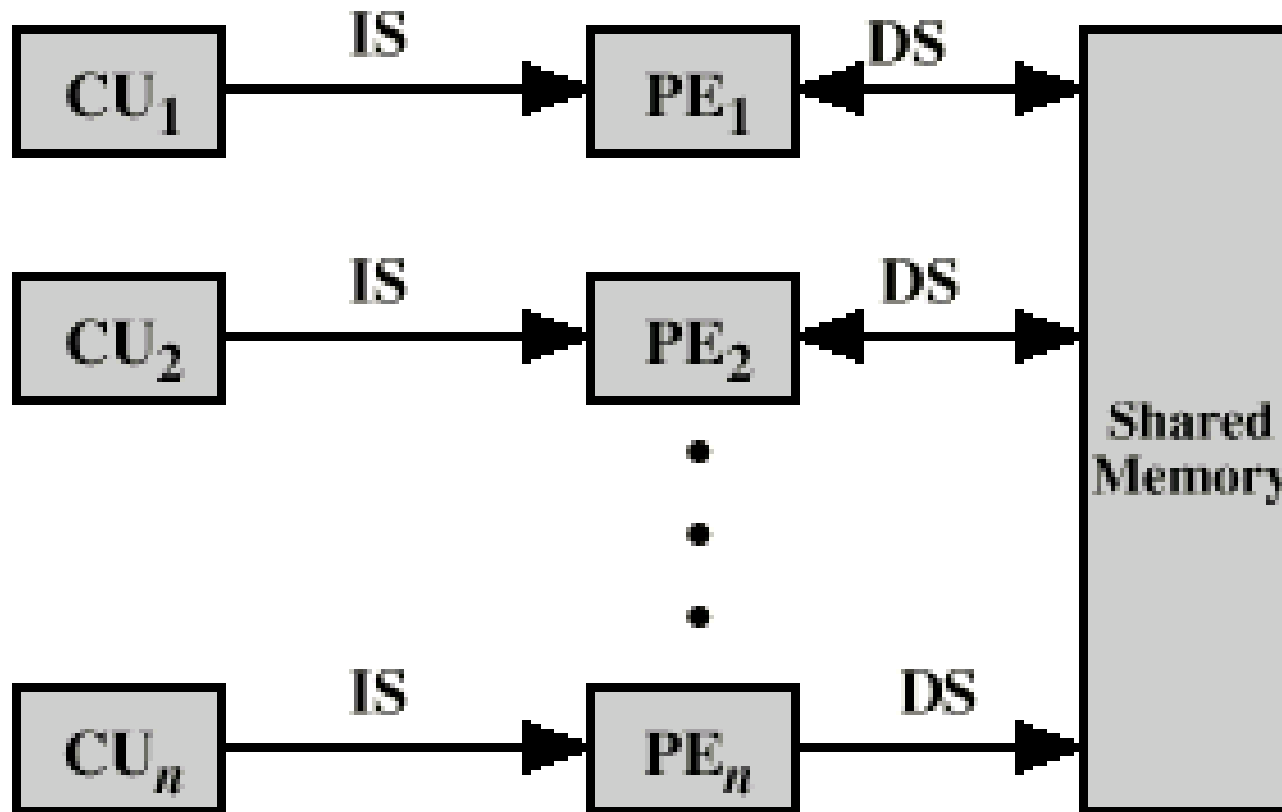
Parallel Organizations - SISD



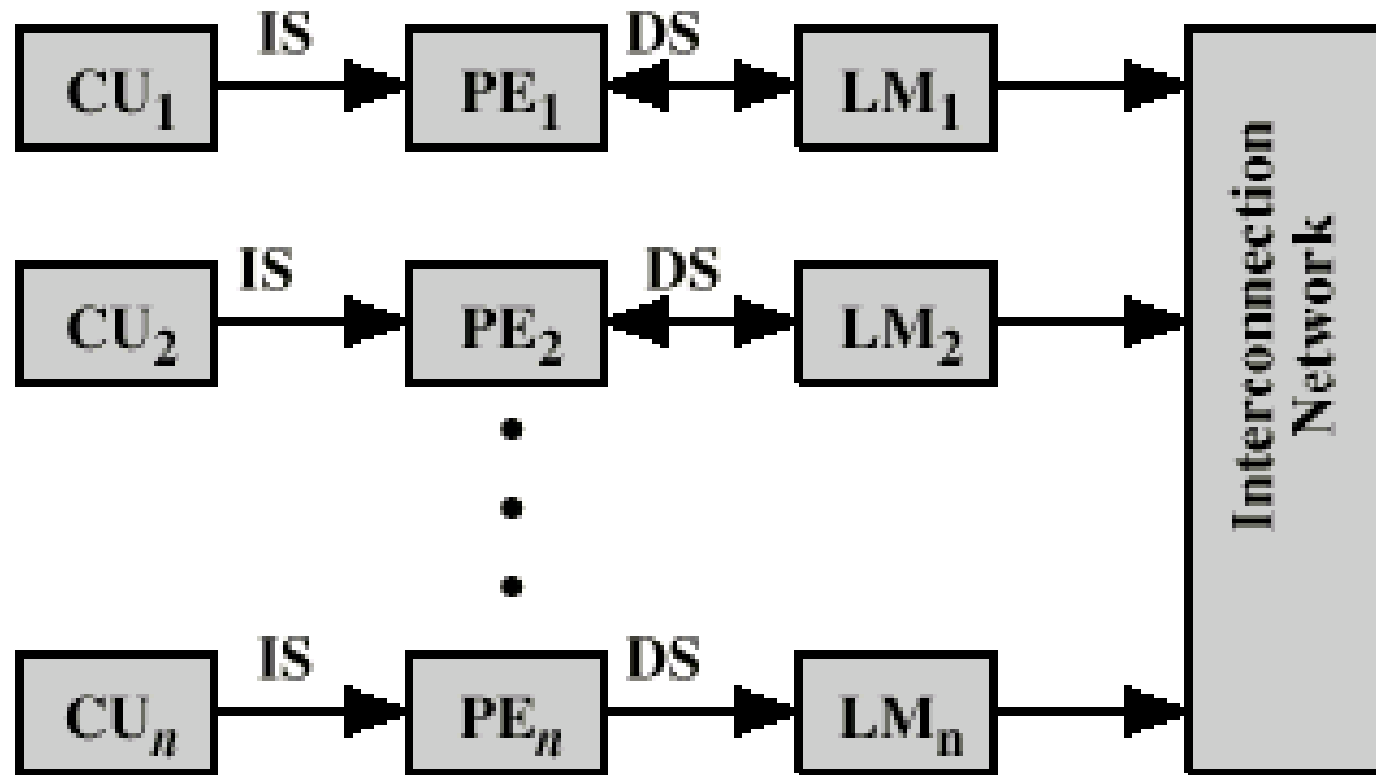
Parallel Organizations - SIMD



Parallel Organizations - MIMD Shared Memory



Parallel Organizations - MIMD Distributed Memory



Symmetric Multiprocessors

- ⌘ A stand alone computer with the following characteristics
 - ☒ Two or more similar processors of comparable capacity
 - ☒ Processors share same memory and I/O
 - ☒ Processors are connected by a bus or other internal connection
 - ☒ Memory access time is approximately the same for each processor
 - ☒ All processors share access to I/O
 - ☒ Either through same channels or different channels giving paths to same devices
 - ☒ All processors can perform the same functions (hence *symmetric*)
 - ☒ System controlled by integrated operating system
 - ☒ providing interaction between processors
 - ☒ Interaction at job, task, file and data element levels

SMP Advantages

⌘ Performance

- ☑ If some work can be done in parallel

⌘ Availability

- ☑ Since all processors can perform the same functions, failure of a single processor does not halt the system

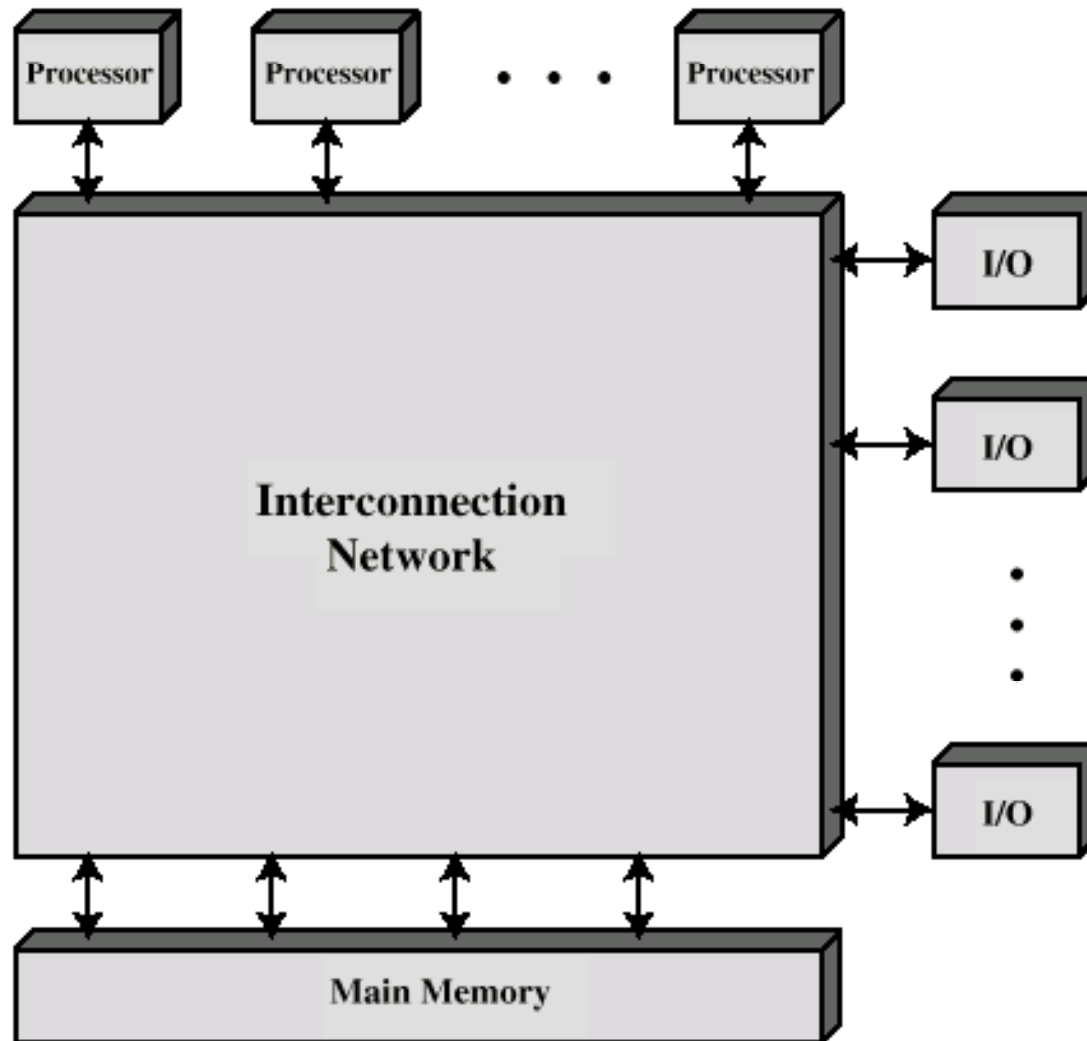
⌘ Incremental growth

- ☑ User can enhance performance by adding additional processors

⌘ Scaling

- ☑ Vendors can offer range of products based on number of processors

Block Diagram of Tightly Coupled Multiprocessor



Organization Classification

- ⌘ Time shared or common bus
- ⌘ Multiport memory
- ⌘ Central control unit

Time Shared Bus

- ⌘ Simplest form
- ⌘ Structure and interface similar to single processor system
- ⌘ Following features provided
 - ☑ Addressing - distinguish modules on bus
 - ☑ Arbitration - any module can be temporary master
 - ☑ Time sharing - if one module has the bus, others must wait and may have to suspend
- ⌘ Now have multiple processors as well as multiple I/O modules

Time Share Bus - Advantages

- ⌘ Simplicity
- ⌘ Flexibility
- ⌘ Reliability

Time Share Bus - Disadvantage

- ⌘ Performance limited by bus cycle time
- ⌘ Each processor should have local cache
 - ☑ Reduce number of bus accesses
- ⌘ Leads to problems with cache coherence
 - ☑ Solved in hardware - see later

Multiport Memory

- ⌘ Direct independent access of memory modules by each processor
- ⌘ Logic required to resolve conflicts
- ⌘ Little or no modification to processors or modules required

Multiport Memory - Advantages and Disadvantages

⌘ More complex

- ⊠ Extra login in memory system

⌘ Better performance

- ⊠ Each processor has dedicated path to each module

⌘ Can configure portions of memory as private to one or more processors

- ⊠ Increased security

⌘ Write through cache policy

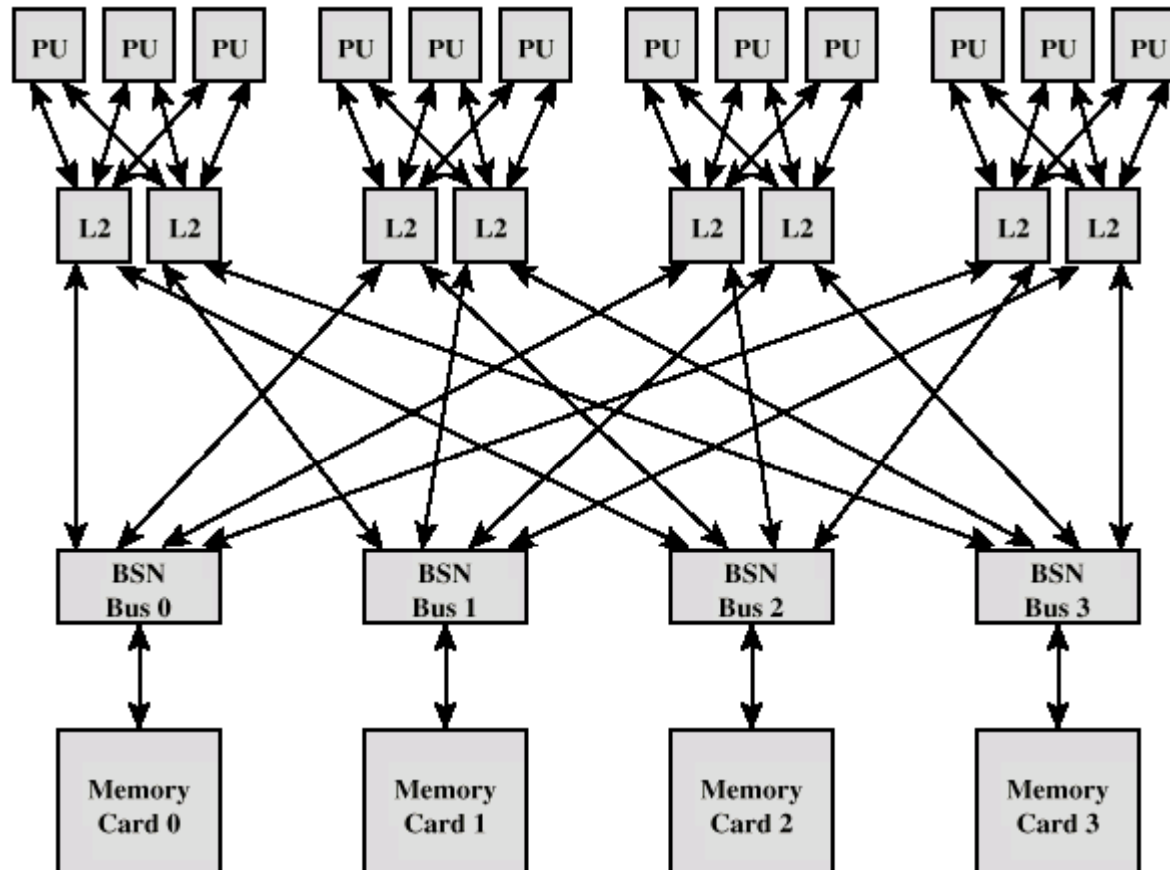
Central Control Unit

- ⌘ Funnel separates data streams between independent modules
- ⌘ Can buffer requests
- ⌘ Performs arbitration and timing
- ⌘ Pass status and control
- ⌘ Perform cache update alerting
- ⌘ Interfaces to modules remain the same
- ⌘ e.g. IBM S/370

Operating System Issues

- ⌘ Simultaneous concurrent processes
- ⌘ Scheduling
- ⌘ Synchronization
- ⌘ Memory management
- ⌘ Reliability and fault tolerance

IBM S/390 Mainframe SMP



S/390 - Key components

⌘ Processor unit (PU)

- ☑ CISC microprocessor
- ☑ Frequently used instructions hard wired
- ☑ 64k L1 unified cache with 1 cycle access time

⌘ L2 cache

- ☑ 384k

⌘ Bus switching network adapter (BSN)

- ☑ Includes 2M of L3 cache

⌘ Memory card

- ☑ 8G per card

Cache Coherence and MESI Protocol

- ⌘ Problem - multiple copies of same data in different caches
- ⌘ Can result in an inconsistent view of memory
- ⌘ Write back policy can lead to inconsistency
- ⌘ Write through can also give problems unless caches monitor memory traffic

Software Solutions

- ⌘ Compiler and operating system deal with problem
- ⌘ Overhead transferred to compile time
- ⌘ Design complexity transferred from hardware to software
- ⌘ However, software tends to make conservative decisions
 - ⊠ Inefficient cache utilization
- ⌘ Analyze code to determine safe periods for caching shared variables

Hardware Solution

- ⌘ Cache coherence protocols
- ⌘ Dynamic recognition of potential problems
- ⌘ Run time
- ⌘ More efficient use of cache
- ⌘ Transparent to programmer
- ⌘ Directory protocols
- ⌘ Snoopy protocols

Directory Protocols

- ⌘ Collect and maintain information about copies of data in cache
- ⌘ Directory stored in main memory
- ⌘ Requests are checked against directory
- ⌘ Appropriate transfers are performed
- ⌘ Creates central bottleneck
- ⌘ Effective in large scale systems with complex interconnection schemes

Snoopy Protocols

- ⌘ Distribute cache coherence responsibility among cache controllers
- ⌘ Cache recognizes that a line is shared
- ⌘ Updates announced to other caches
- ⌘ Suited to bus based multiprocessor
- ⌘ Increases bus traffic

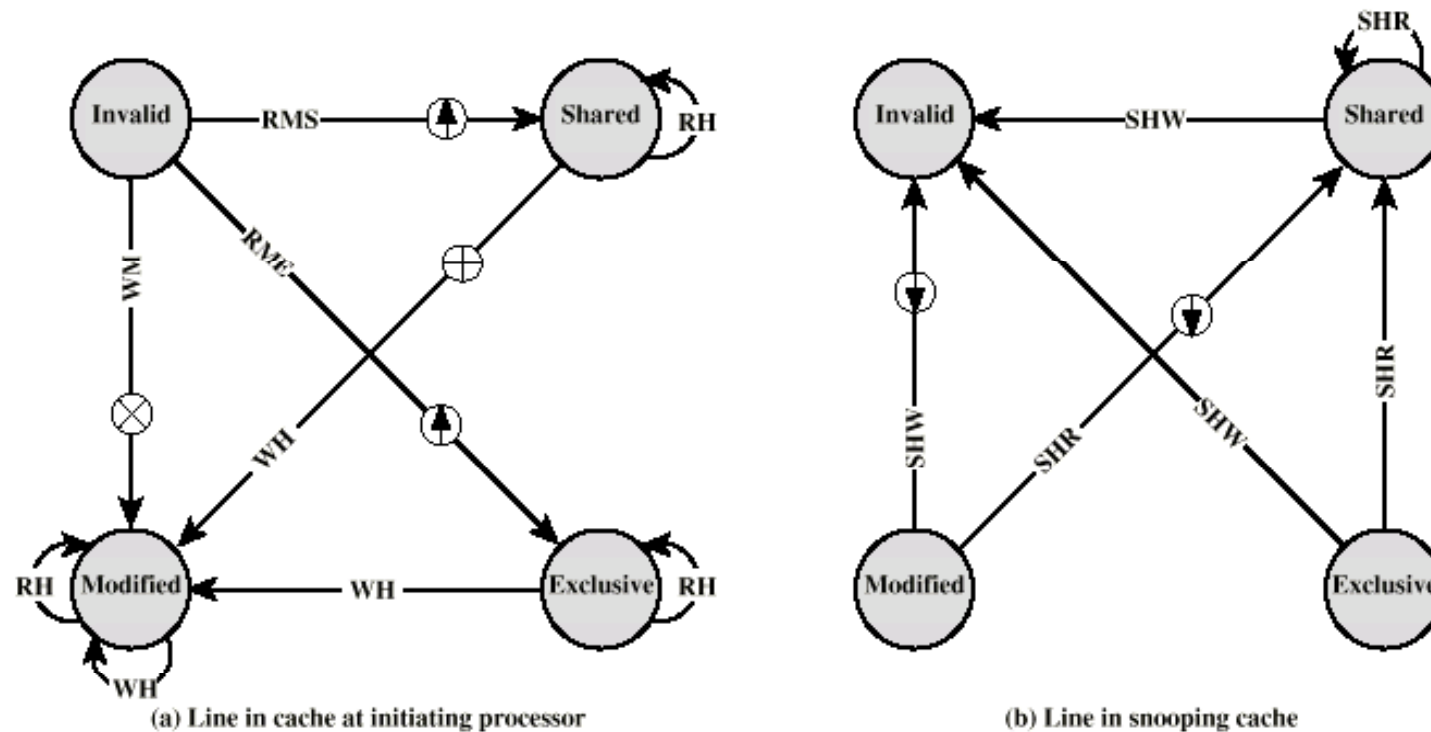
Write Invalidate

- ⌘ Multiple readers, one writer
- ⌘ When a write is required, all other caches of the line are invalidated
- ⌘ Writing processor then has exclusive (cheap) access until line required by another processor
- ⌘ Used in Pentium II and PowerPC systems
- ⌘ State of every line is marked as modified, exclusive, shared or invalid
- ⌘ MESI

Write Update

- ⌘ Multiple readers and writers
- ⌘ Updated word is distributed to all other processors
- ⌘ Some systems use an adaptive mixture of both solutions

MESI State Transition Diagram



RH	Read hit		Dirty line copyback
RMS	Read miss, shared		Invalidate transaction
RME	Read miss, exclusive		Read-with-intent-to-modify
WH	Write hit		Cache line fill
WM	Write miss		
SHR	Snoop hit on read		
SHW	Snoop hit on write or read-with-intent-to-modify		

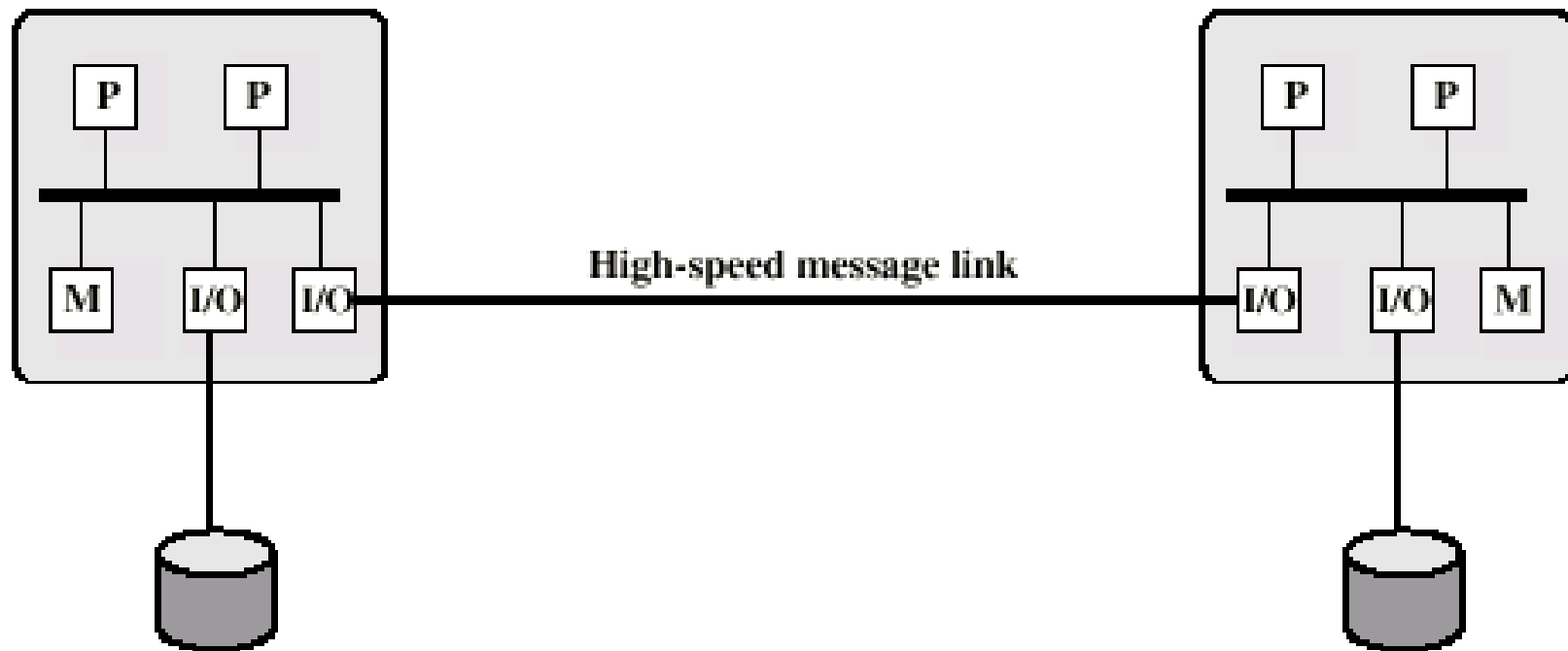
Clusters

- ⌘ Alternative to SMP
 - ⌘ High performance
 - ⌘ High availability
 - ⌘ Server applications
-
- ⌘ A group of interconnected whole computers
 - ⌘ Working together as unified resource
 - ⌘ Illusion of being one machine
 - ⌘ Each computer called a node

Cluster Benefits

- ⌘ Absolute scalability
- ⌘ Incremental scalability
- ⌘ High availability
- ⌘ Superior price/performance

Cluster Configurations - Standby Server, No Shared Disk



Cluster Configurations - Shared Disk

