William Stallings
Computer Organization and Architecture
8th Edition

Chapter 17

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

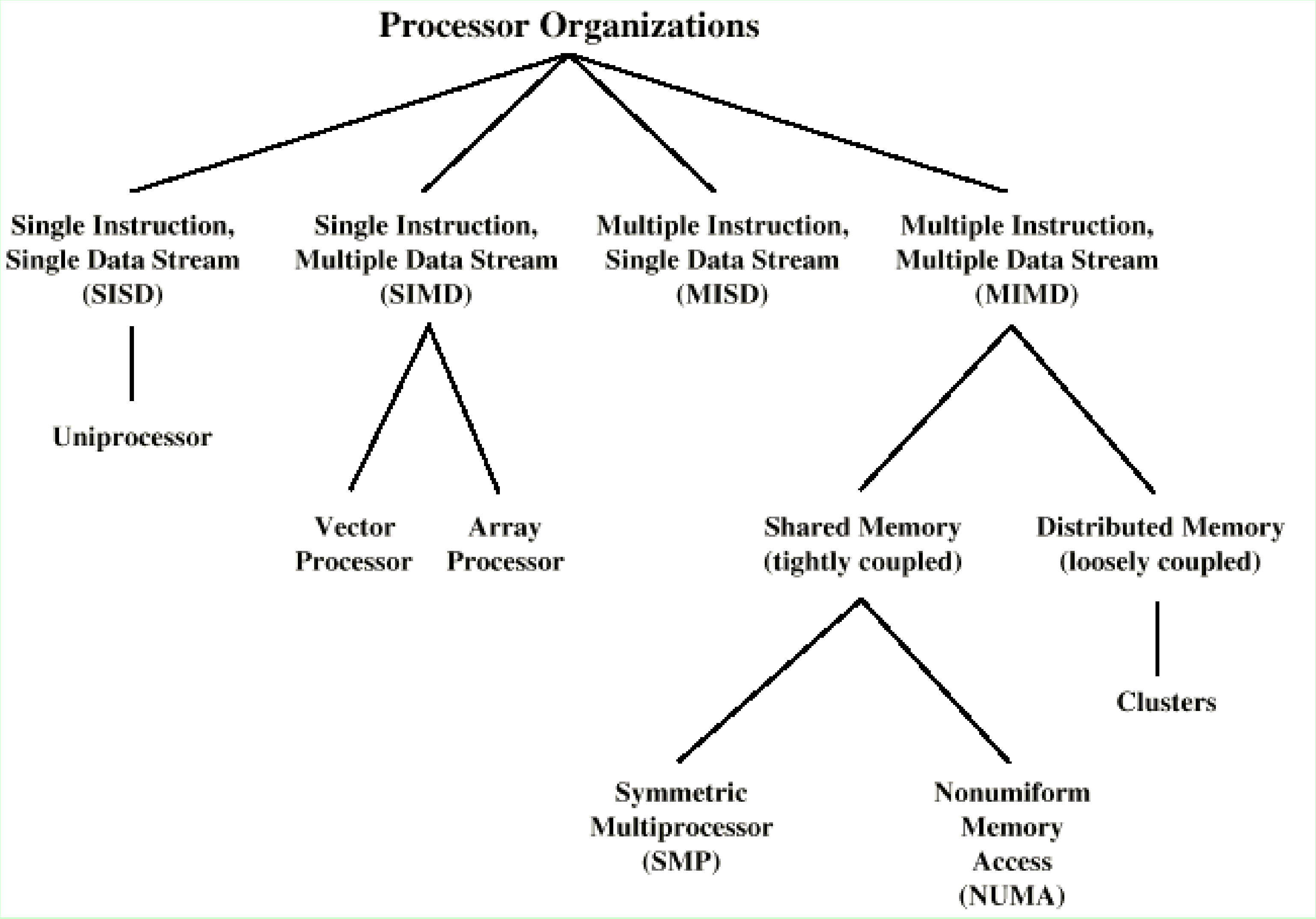
Multiple Instruction, Single Data Stream - MISD

- 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

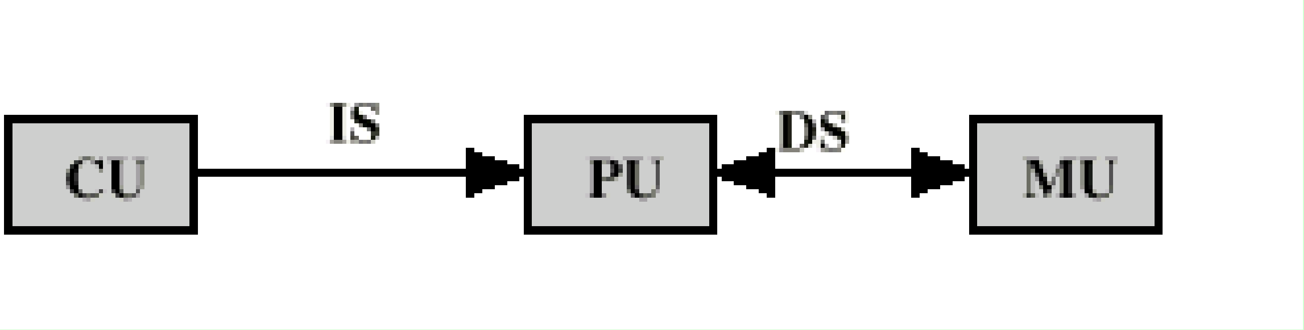
Tightly Coupled - NUMA

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

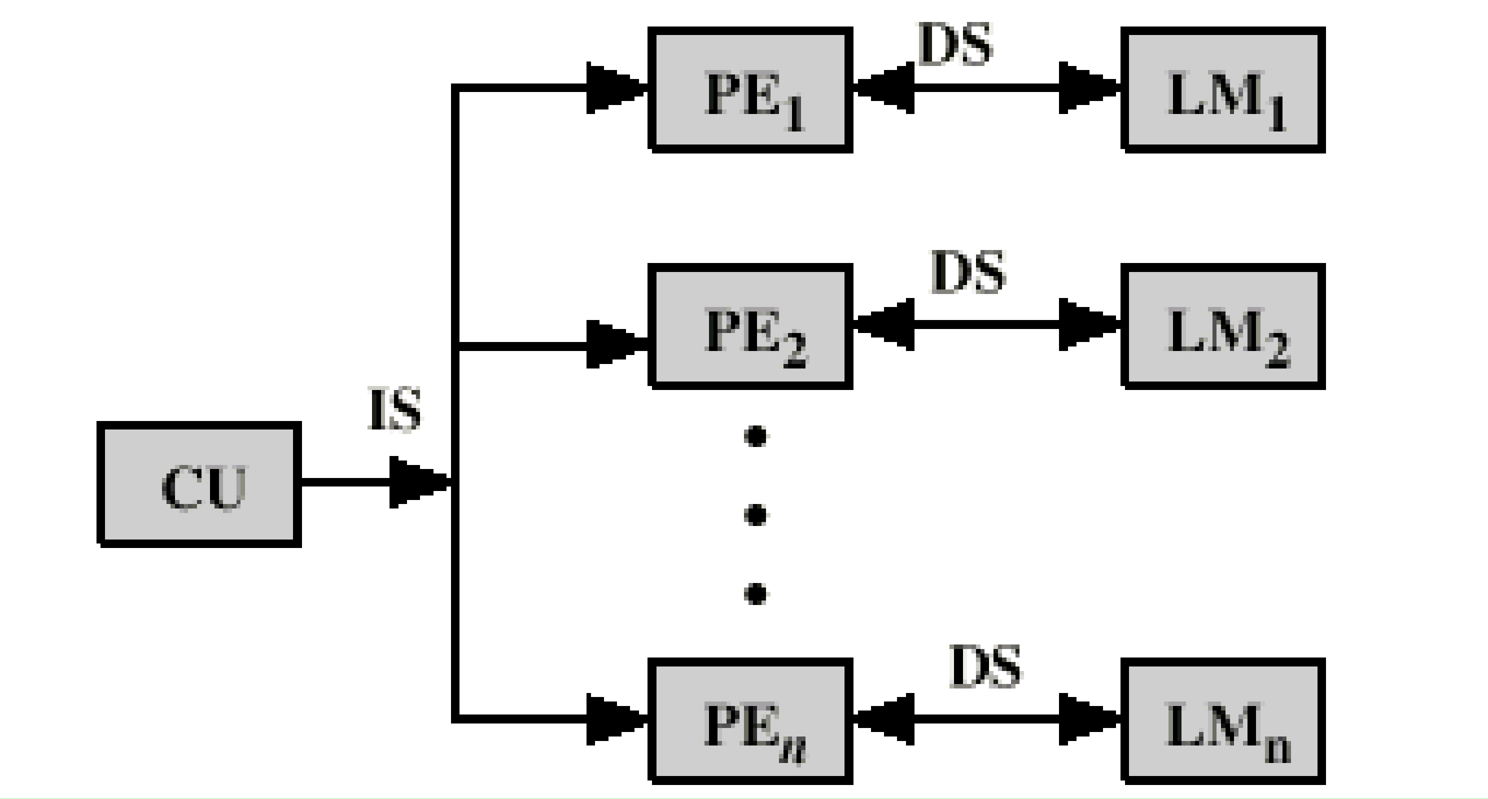
Loosely Coupled - Clusters

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

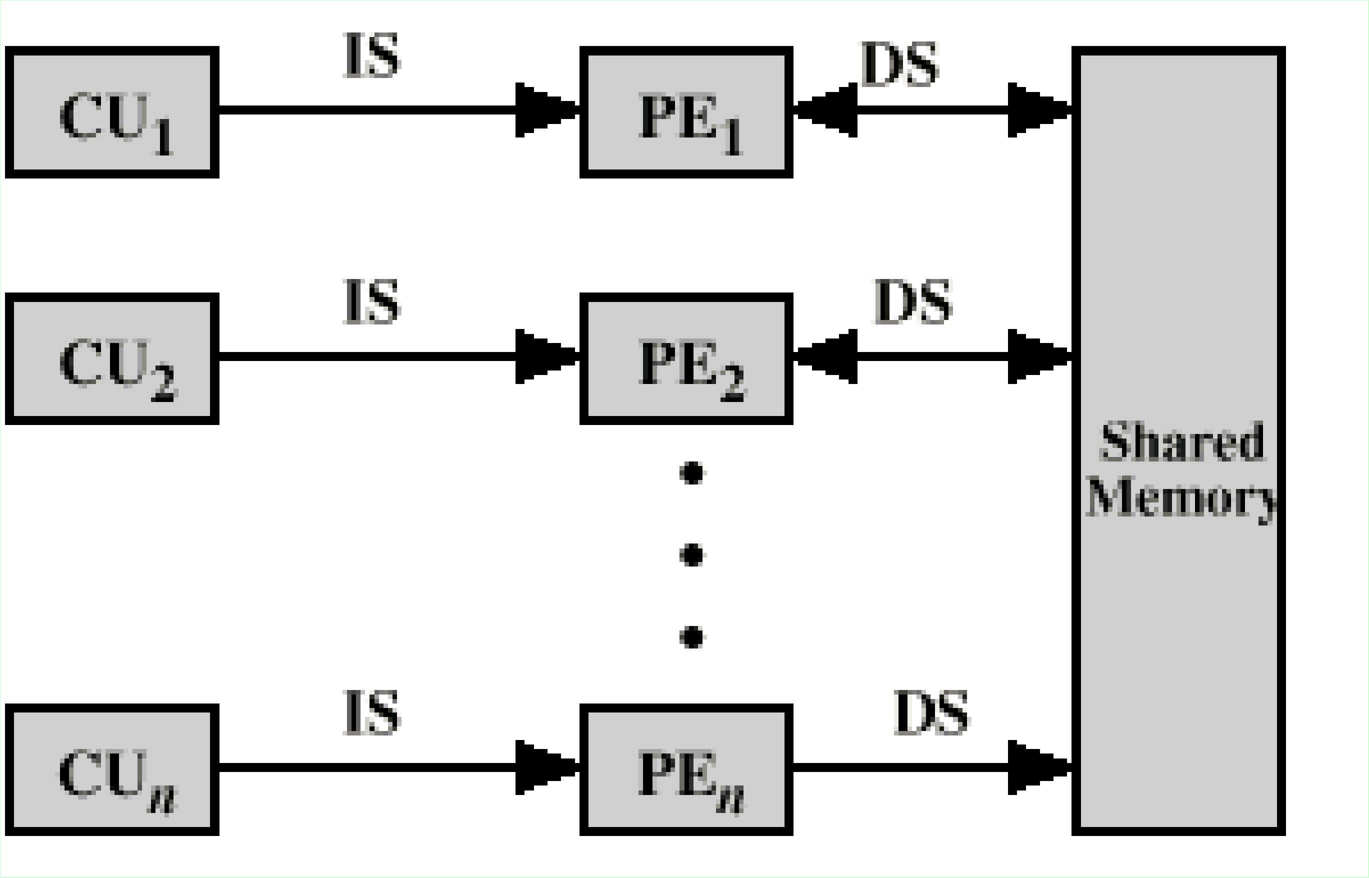
Parallel Organizations - SISD



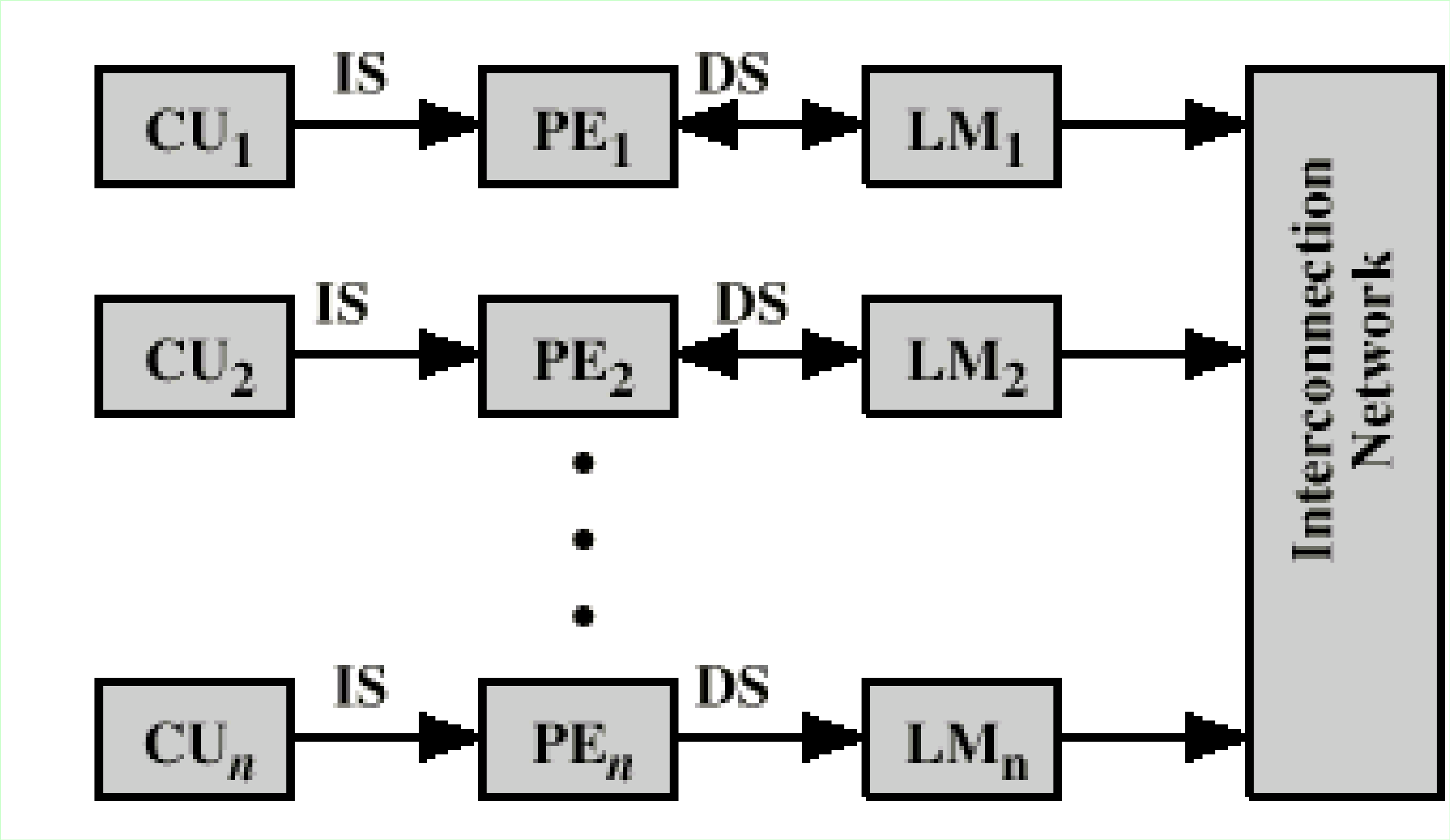
Parallel Organizations - SIMD



Parallel Organizations - MIMD Shared Memory



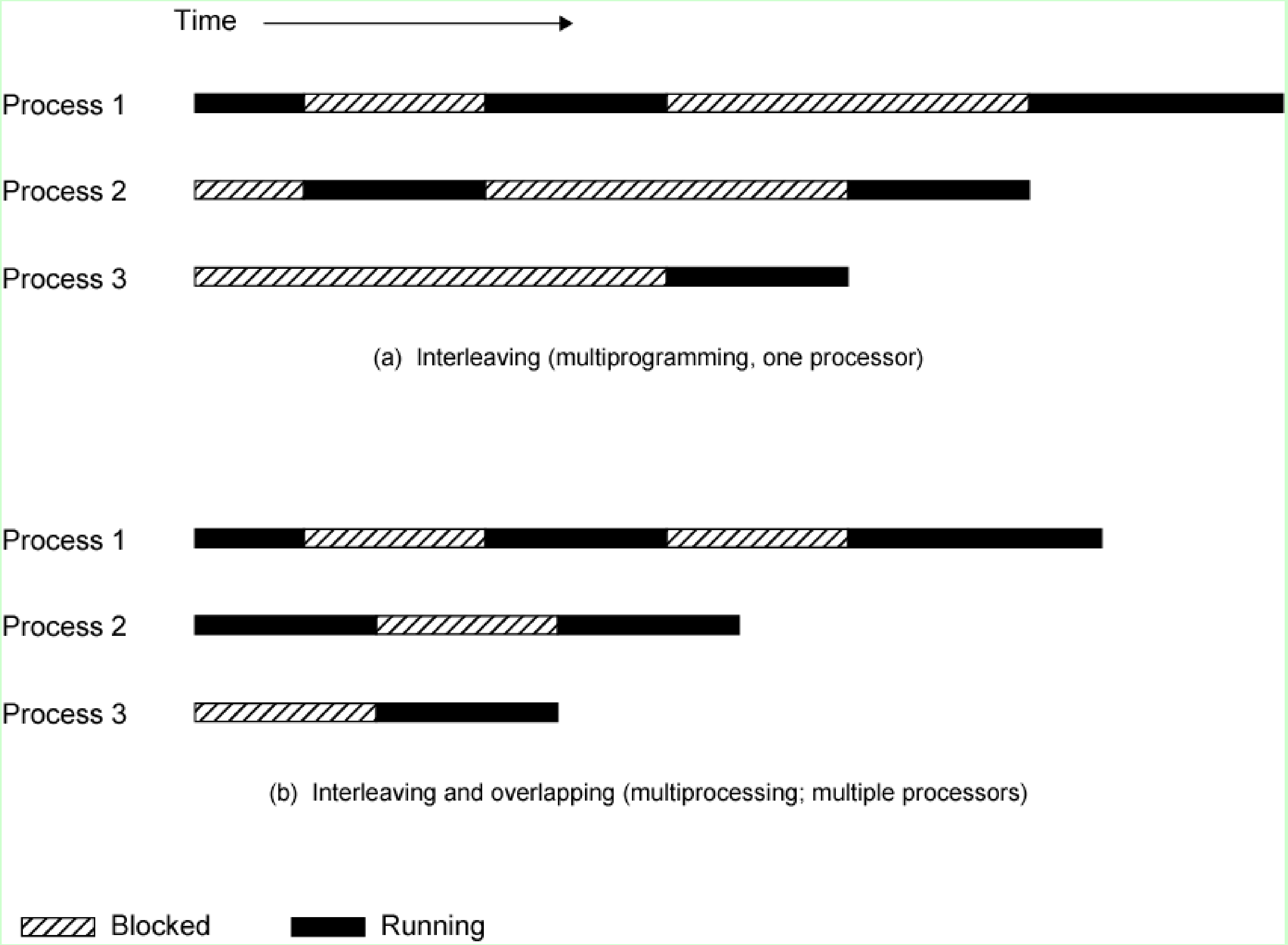
Parallel Organizations - MIMD Distributed Memory



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

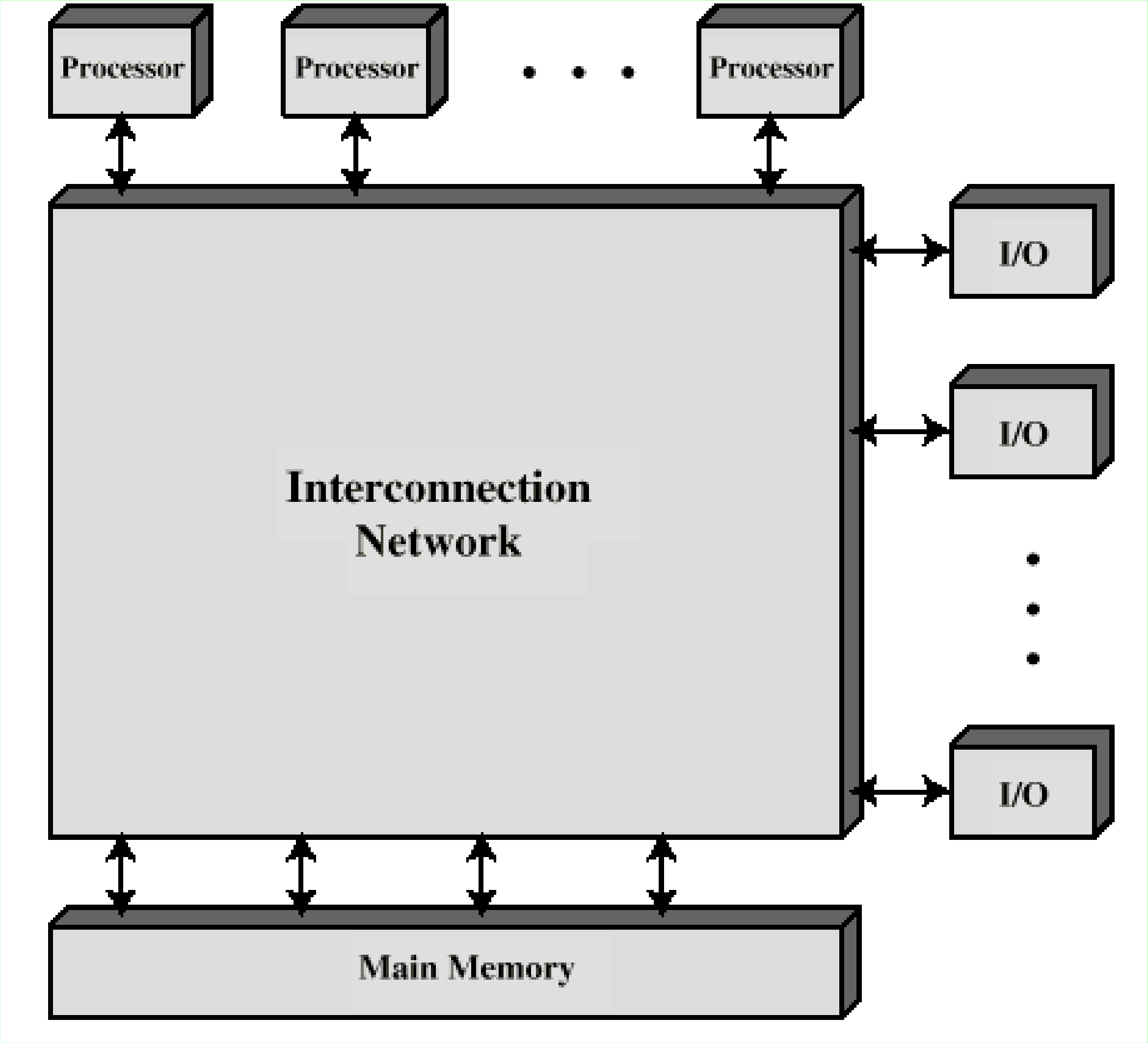
Multiprogramming and Multiprocessing



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



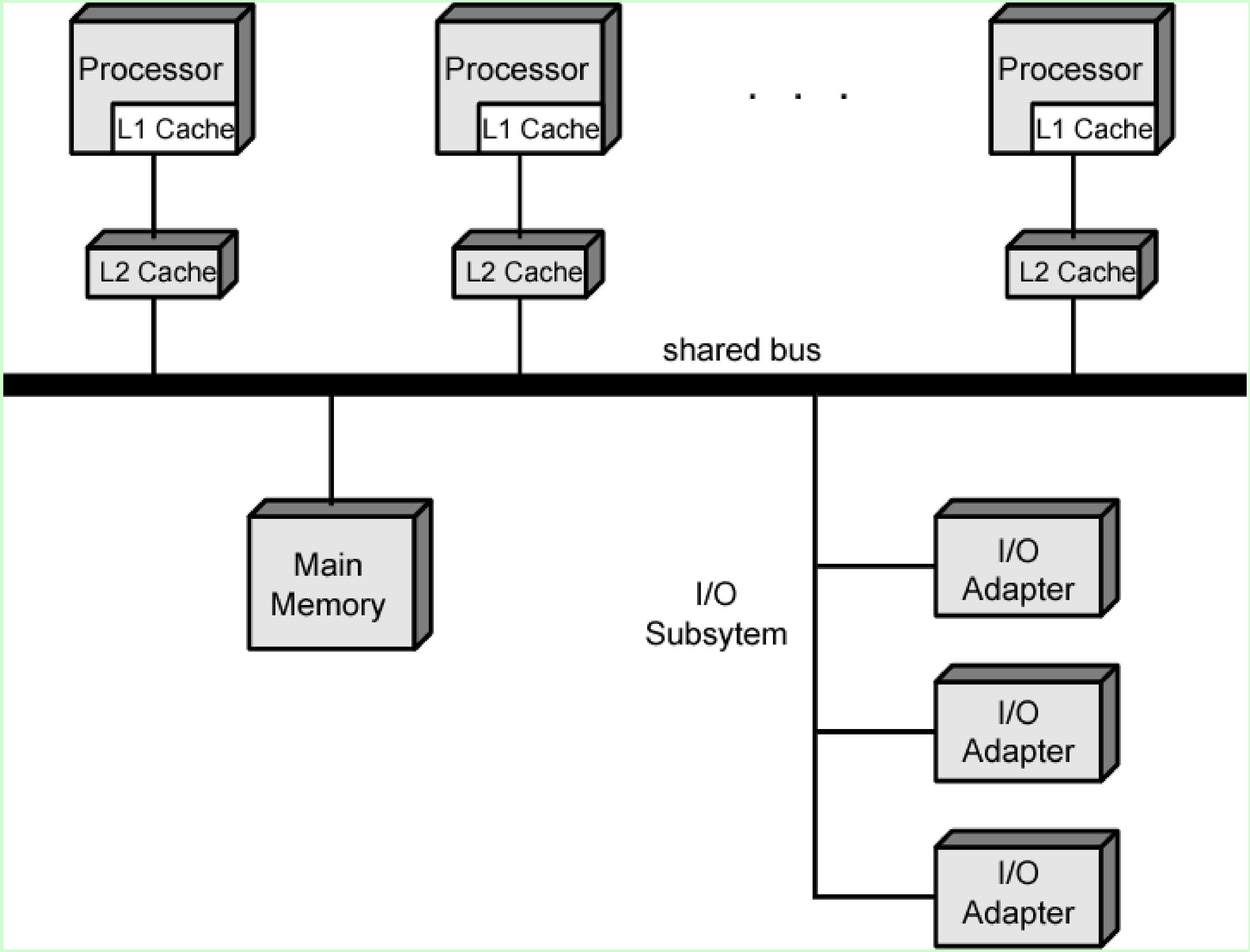
Organization Classification

- 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

Symmetric Multiprocessor Organization



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

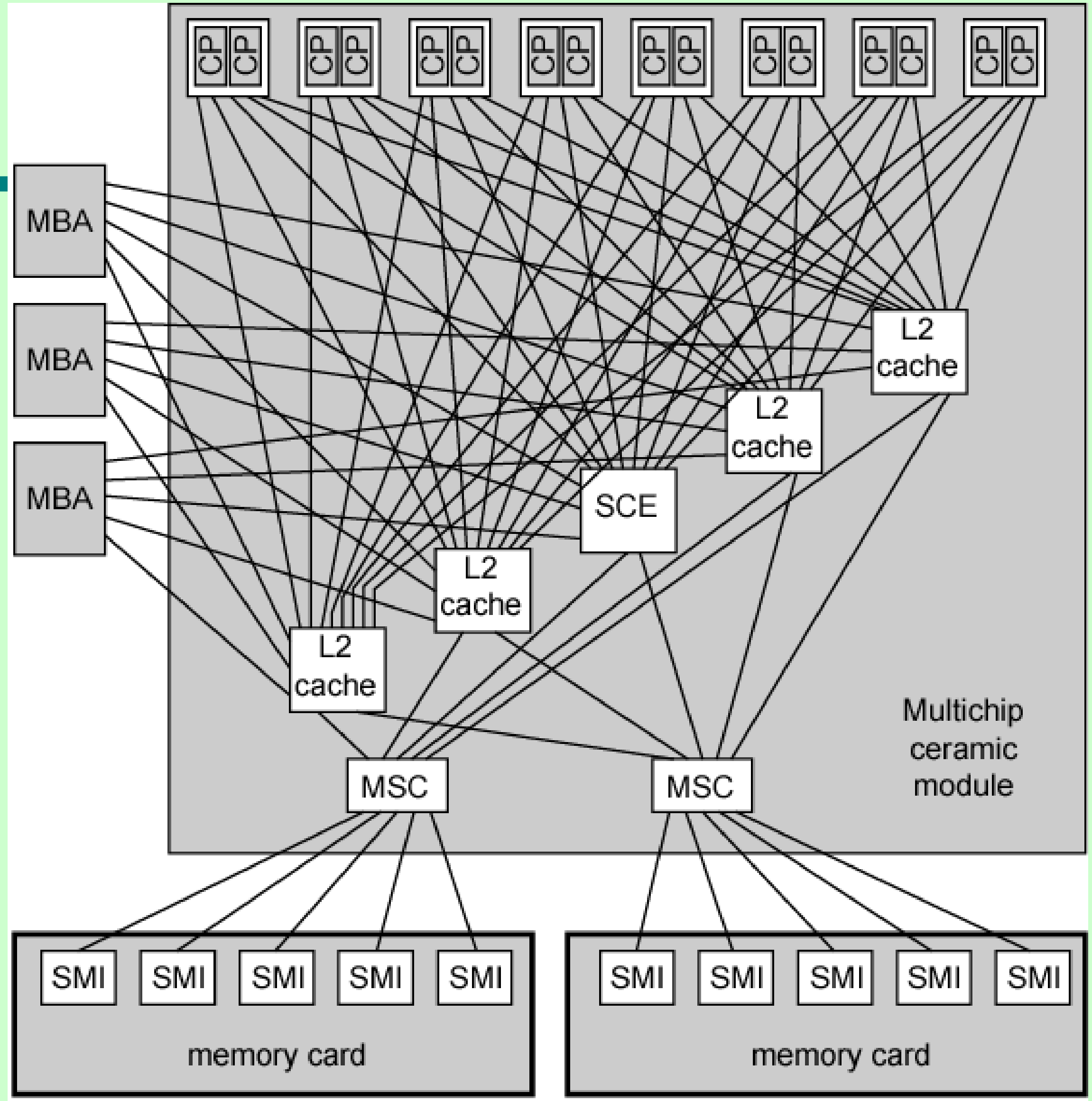
Operating System Issues

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

A Mainframe SMP IBM zSeries

- Uniprocessor with one main memory card to a high-end system with 48 processors and 8 memory cards
- Dual-core processor chip
  - Each includes two identical central processors (CPs)
  - CISC superscalar microprocessor
  - Mostly hardwired, some vertical microcode
  - 256-kB L1 instruction cache and a 256-kB L1 data cache
- L2 cache 32 MB
  - Clusters of five
  - Each cluster supports eight processors and access to entire main memory space
- System control element (SCE)
  - Arbitrates system communication
  - Maintains cache coherence
- Main store control (MSC)
  - Interconnect L2 caches and main memory
- Memory card
  - Each 32 GB, Maximum 8 , total of 256 GB
  - Interconnect to MSC via synchronous memory interfaces (SMIs)
- Memory bus adapter (MBA)
  - Interface to I/O channels, go directly to L2 cache

IBM z990 Multiprocessor Structure



CP = central processor
MBA = memory bus adapter
MSC = main store control
SCE = system control element
SMI = synchronous memory interface

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

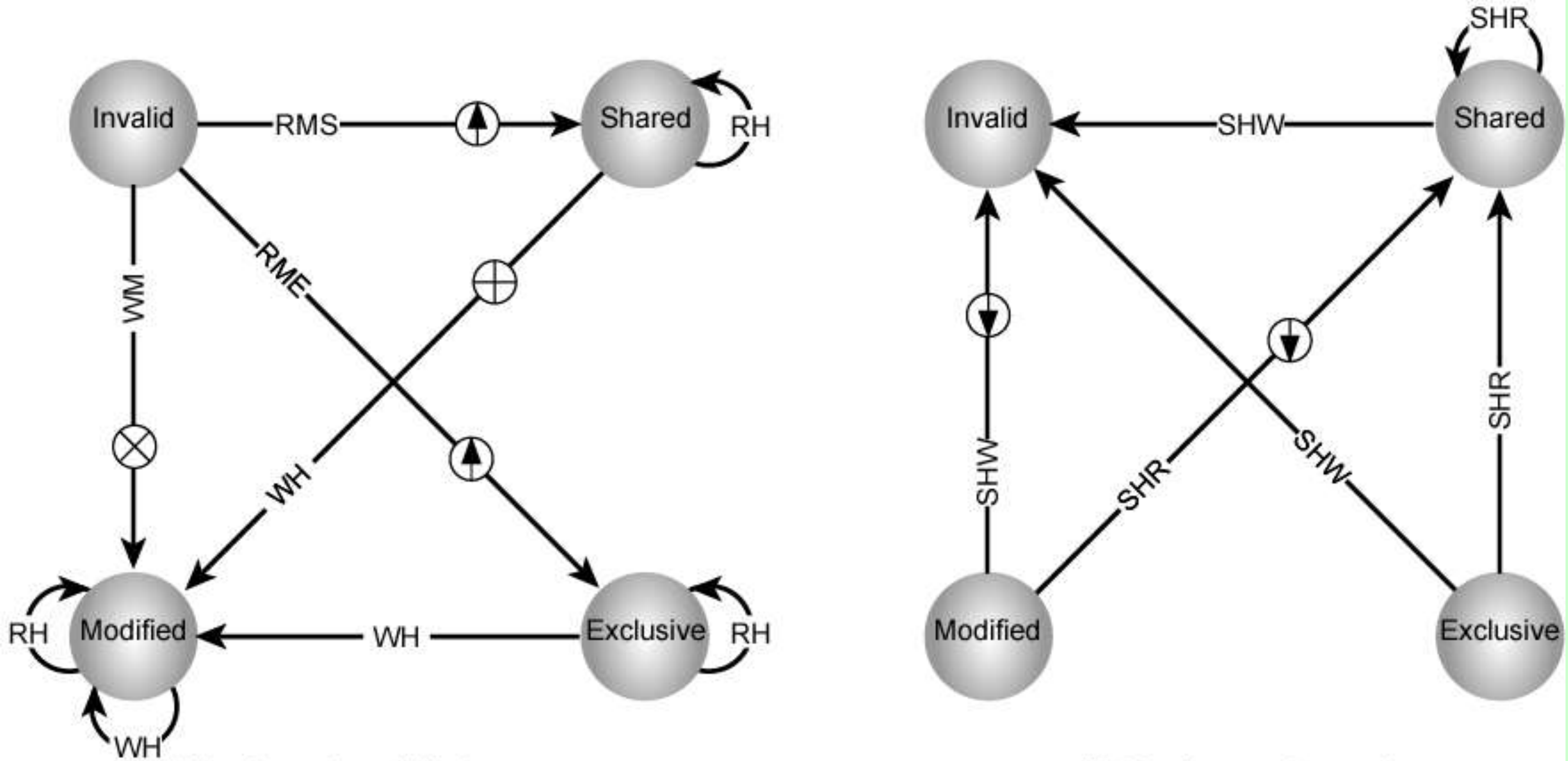
- Multiple readers, one writer
- When a write is required, all other caches of the line are invalidated
- Writing processor then has exclusive (cheap) access until line required by another processor
- Used in Pentium II and PowerPC systems
- State of every line is marked as modified, exclusive, shared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



(a) Line in cache at initiating processor

(b) Line in snooping cache

RH	Read hit
RMS	Read miss, shared
RME	Read miss, exclusive
WH	Write hit
WM	Write miss
SHR	Snoop hit on read
SHW	Snoop hit on write or read-with-intent-to-modify

- Dirty line copyback
- Invalidate transaction
- Read-with-intent-to-modify
- Cache line fill

Increasing Performance

- Processor performance can be measured by the rate at which it executes instructions
- MIPS rate = f * IPC
  - —f processor clock frequency, in MHz
  - —IPC is average instructions per cycle
- Increase performance by increasing clock frequency and increasing instructions that complete during cycle
- May be reaching limit
  - —Complexity
  - —Power consumption

Multithreading and Chip Multiprocessors

- Instruction stream divided into smaller streams (threads)
- Executed in parallel
- Wide variety of multithreading designs

Definitions of Threads and Processes

- Thread in multithreaded processors may or may not be same as software threads
- Process:
  - An instance of program running on computer
  - Resource ownership
    - Virtual address space to hold process image
  - Scheduling/execution
  - Process switch
- Thread: dispatchable unit of work within process
  - Includes processor context (which includes the program counter and stack pointer) and data area for stack
  - Thread executes sequentially
  - Interruptible: processor can turn to another thread
- Thread switch
  - Switching processor between threads within same process
  - Typically less costly than process switch

Implicit and Explicit Multithreading

- All commercial processors and most experimental ones use explicit multithreading
  - Concurrently execute instructions from different explicit threads
  - Interleave instructions from different threads on shared pipelines or parallel execution on parallel pipelines
- Implicit multithreading is concurrent execution of multiple threads extracted from single sequential program
  - Implicit threads defined statically by compiler or dynamically by hardware

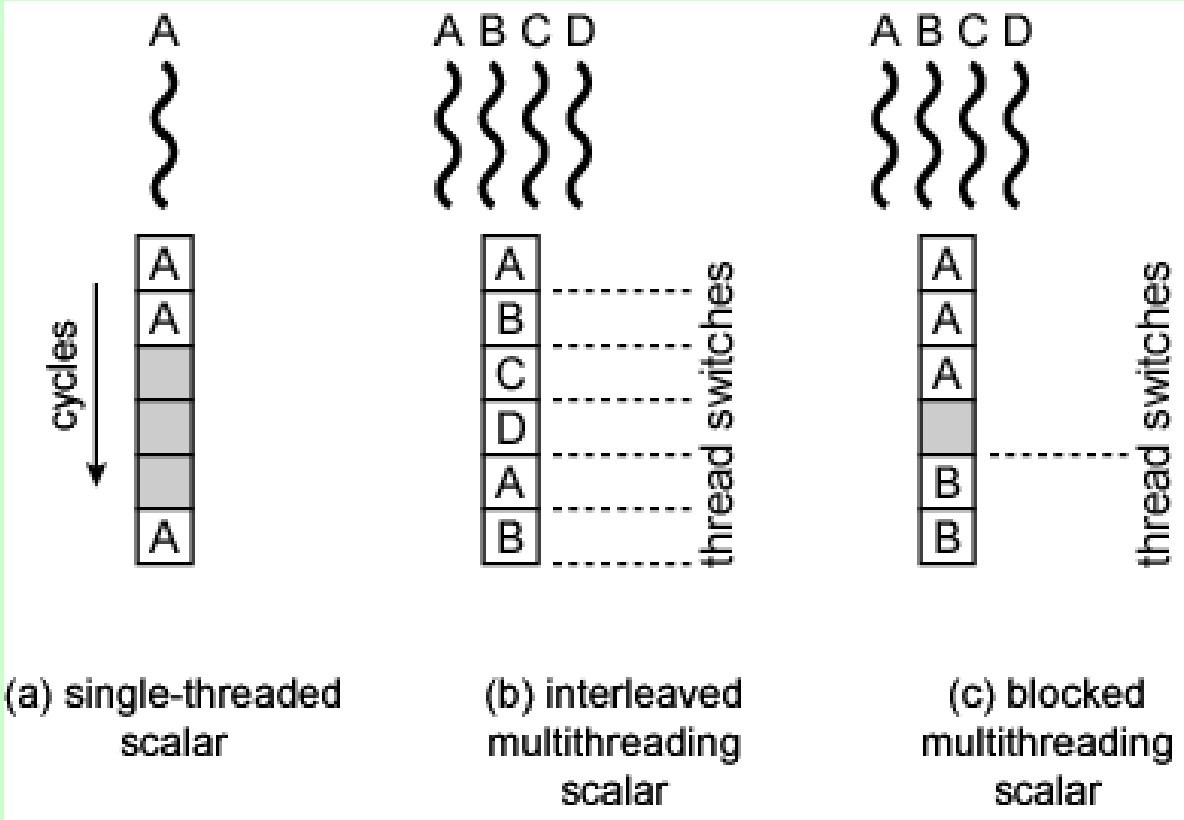
Approaches to Explicit Multithreading

- Interleaved
  - Fine-grained
  - Processor deals with two or more thread contexts at a time
  - Switching thread at each clock cycle
  - If thread is blocked it is skipped
- Blocked
  - Coarse-grained
  - Thread executed until event causes delay
  - E.g.Cache miss
  - Effective on in-order processor
  - Avoids pipeline stall
- Simultaneous (SMT)
  - Instructions simultaneously issued from multiple threads to execution units of superscalar processor
- Chip multiprocessing
  - Processor is replicated on a single chip
  - Each processor handles separate threads

Scalar Processor Approaches

- Single-threaded scalar
  - Simple pipeline
  - No multithreading
- Interleaved multithreaded scalar
  - Easiest multithreading to implement
  - Switch threads at each clock cycle
  - Pipeline stages kept close to fully occupied
  - Hardware needs to switch thread context between cycles
- Blocked multithreaded scalar
  - Thread executed until latency event occurs
  - Would stop pipeline
  - Processor switches to another thread

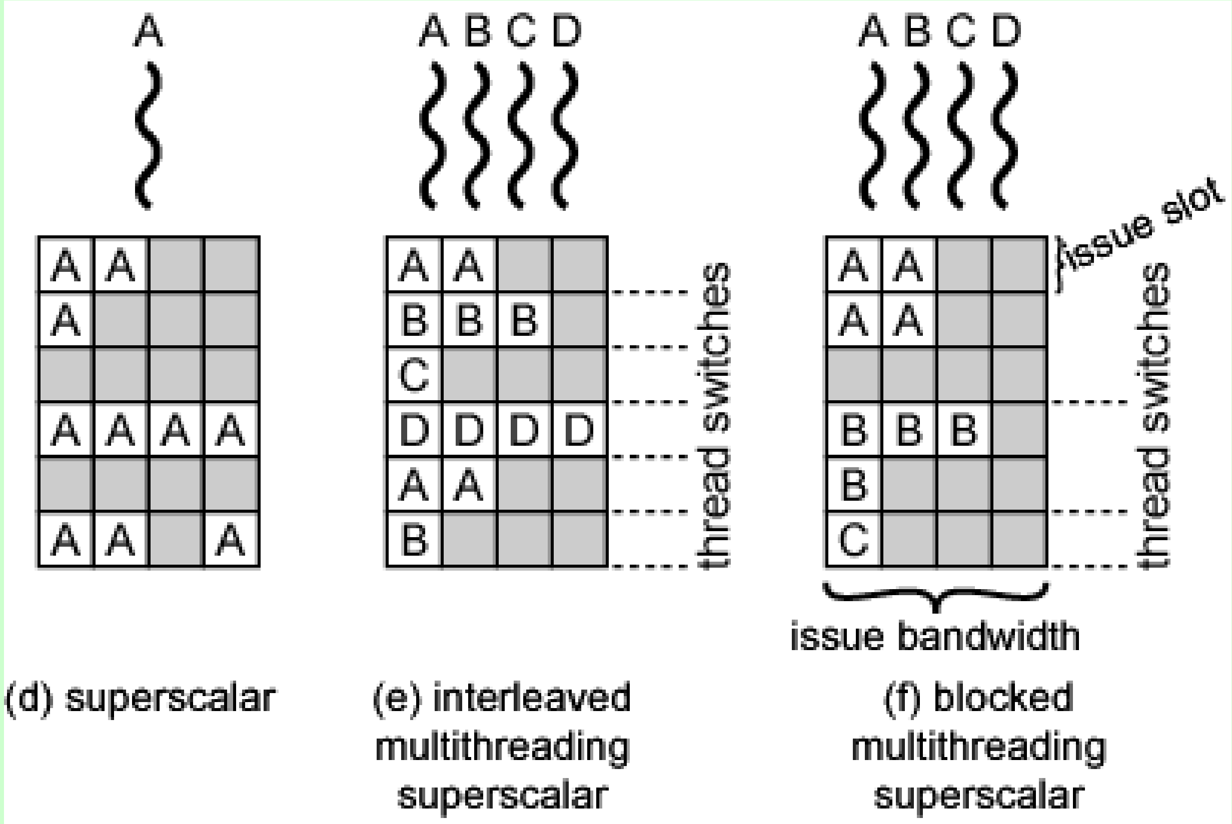
Scalar Diagrams



Multiple Instruction Issue Processors (1)

- Superscalar
  - No multithreading
- Interleaved multithreading superscalar:
  - Each cycle, as many instructions as possible issued from single thread
  - Delays due to thread switches eliminated
  - Number of instructions issued in cycle limited by dependencies
- Blocked multithreaded superscalar
  - Instructions from one thread
  - Blocked multithreading used

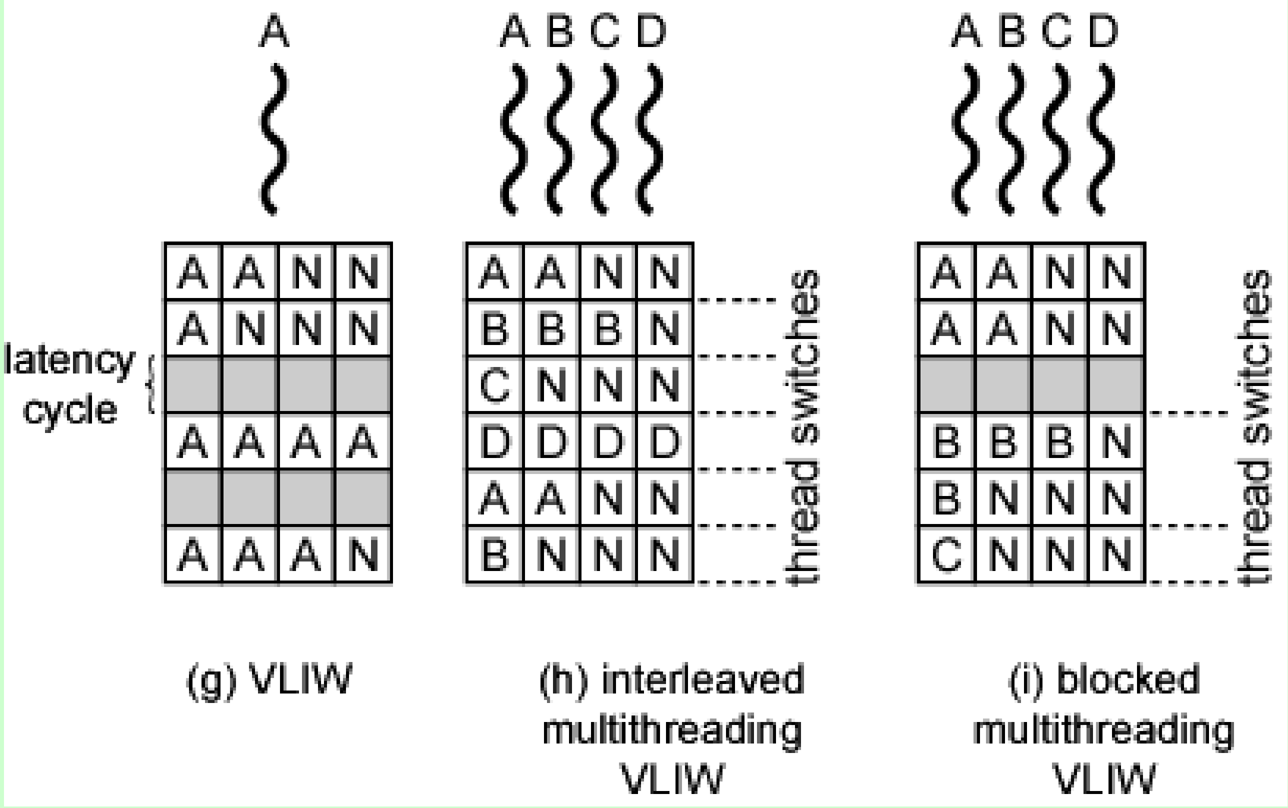
Multiple Instruction Issue Diagram (1)



Multiple Instruction Issue Processors (2)

- Very long instruction word (VLIW)
  - E.g. IA-64
  - Multiple instructions in single word
  - Typically constructed by compiler
  - Operations that may be executed in parallel in same word
  - May pad with no-ops
- Interleaved multithreading VLIW
  - Similar efficiencies to interleaved multithreading on superscalar architecture
- Blocked multithreaded VLIW
  - Similar efficiencies to blocked multithreading on superscalar architecture

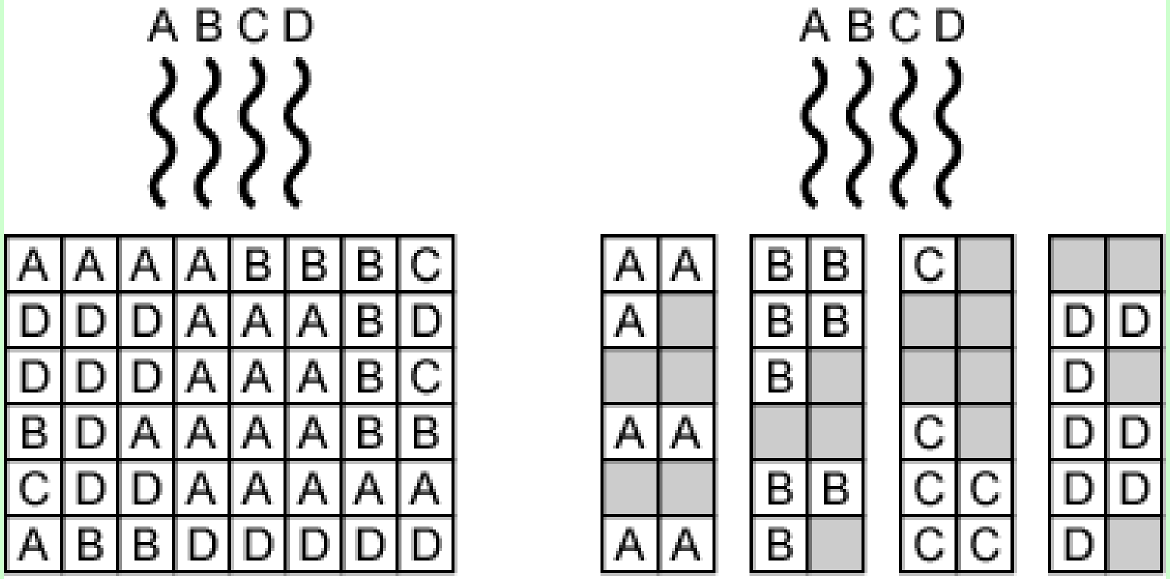
Multiple Instruction Issue Diagram (2)



Parallel, Simultaneous Execution of Multiple Threads

- Simultaneous multithreading
  - Issue multiple instructions at a time
  - One thread may fill all horizontal slots
  - Instructions from two or more threads may be issued
  - With enough threads, can issue maximum number of instructions on each cycle
- Chip multiprocessor
  - Multiple processors
  - Each has two-issue superscalar processor
  - Each processor is assigned thread
    - Can issue up to two instructions per cycle per thread

Parallel Diagram



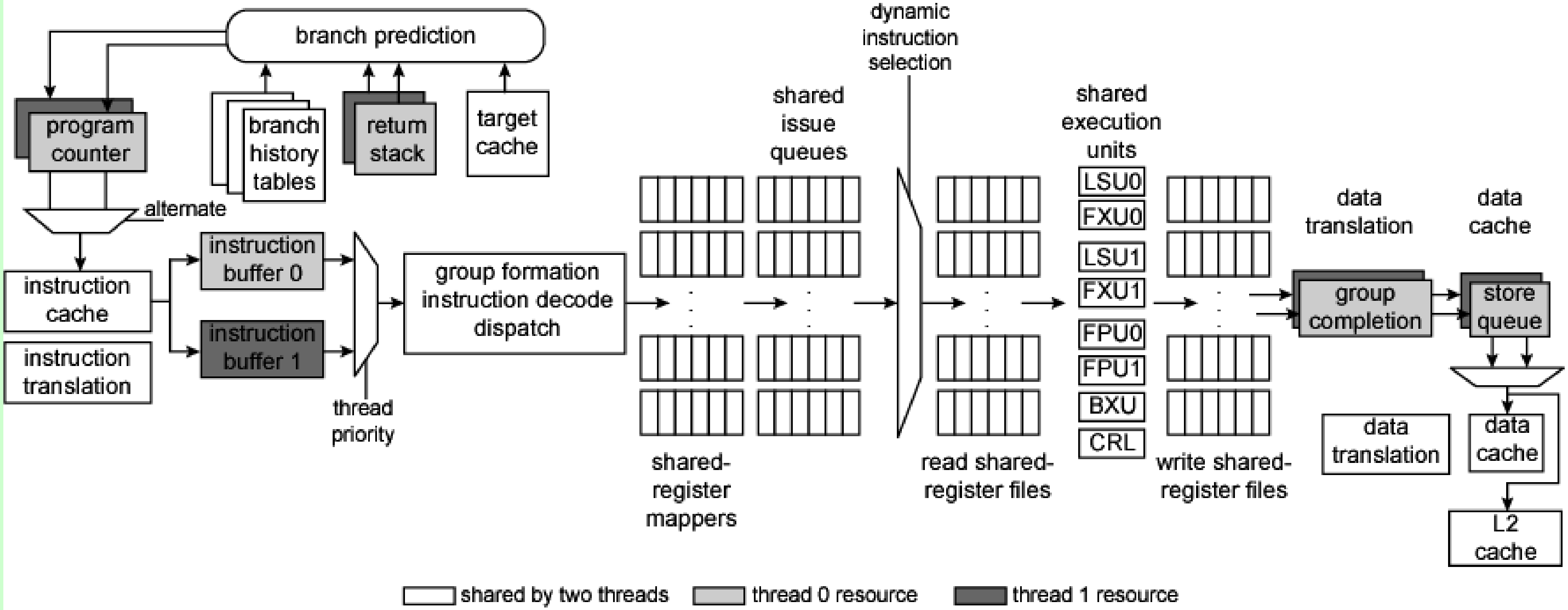
(j) simultaneous multithreading (SMT)

(k) chip multiprocessor

Examples

- Some Pentium 4
  - Intel calls it hyperthreading
  - SMT with support for two threads
  - Single multithreaded processor, logically two processors
- IBM Power5
  - High-end PowerPC
  - Combines chip multiprocessing with SMT
  - Chip has two separate processors
  - Each supporting two threads concurrently using SMT

Power5 Instruction Data Flow



BXU = branch execution unit and
CRL = condition register logical execution unit
FPU = floating-point execution unit
FXU = fixed-point execution unit
LSU = load/store unit

Clusters

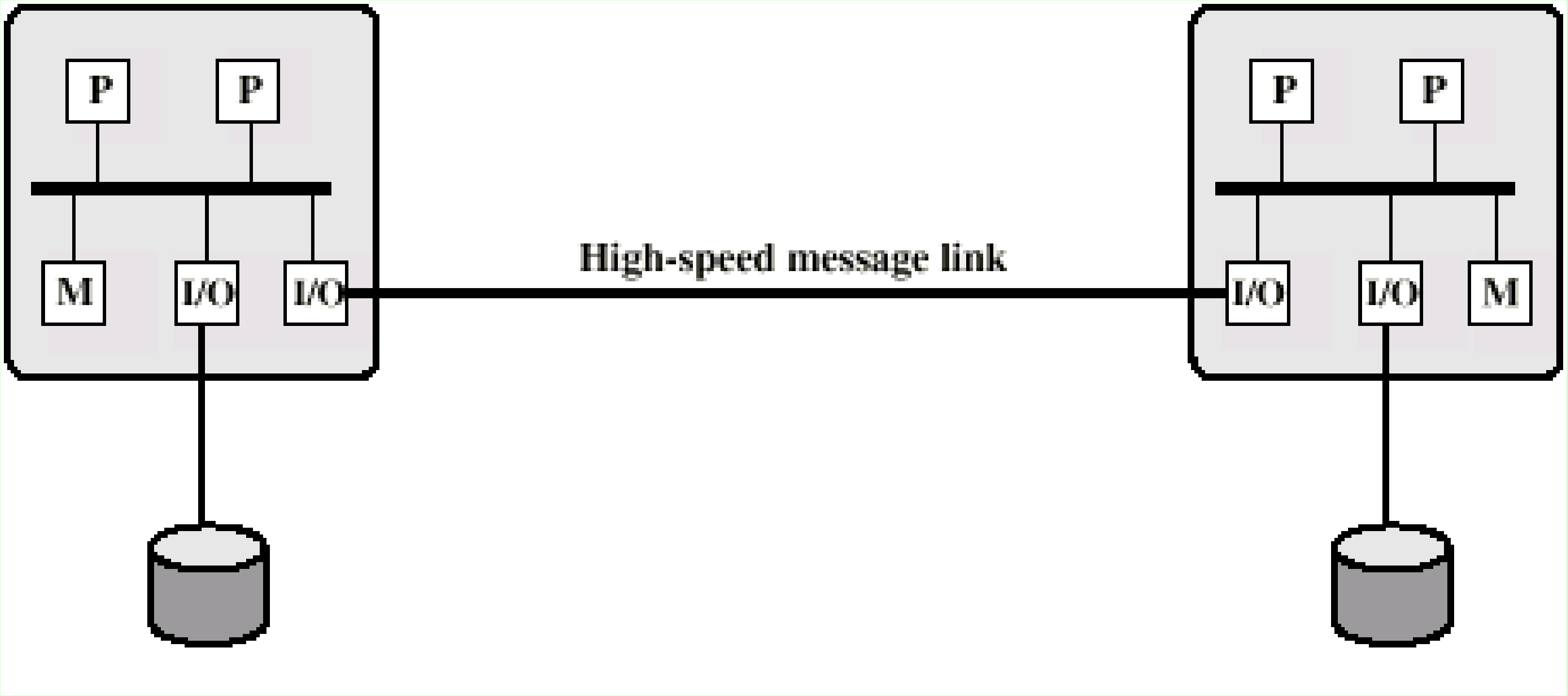
- Alternative to SMP
- High performance
- High availability
- Server applications

- A group of interconnected whole computers
- Working together as unified resource
- Illusion of being one machine
- Each computer called a node

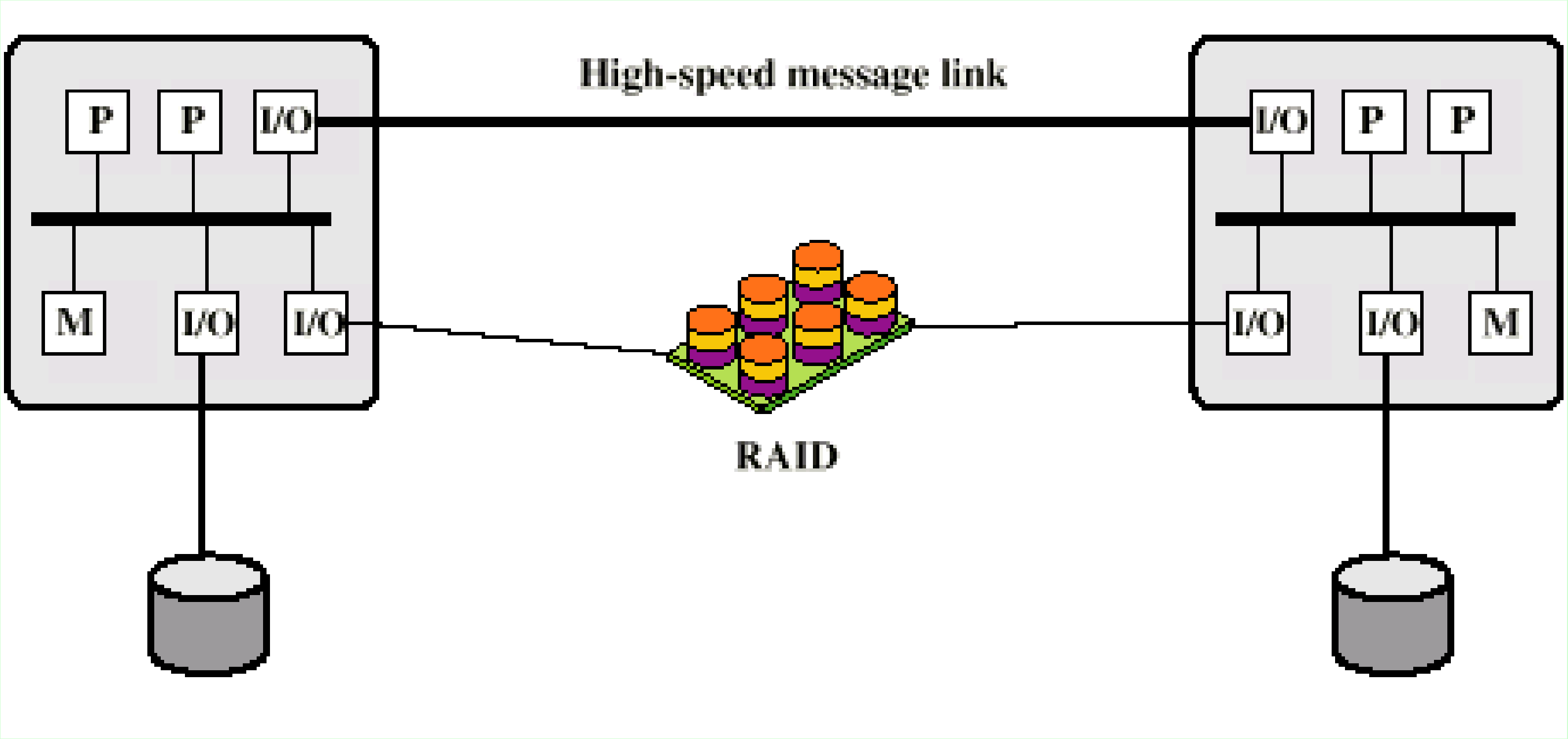
Cluster Benefits

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

Cluster Configurations - Standby Server, No Shared Disk



Cluster Configurations - Shared Disk



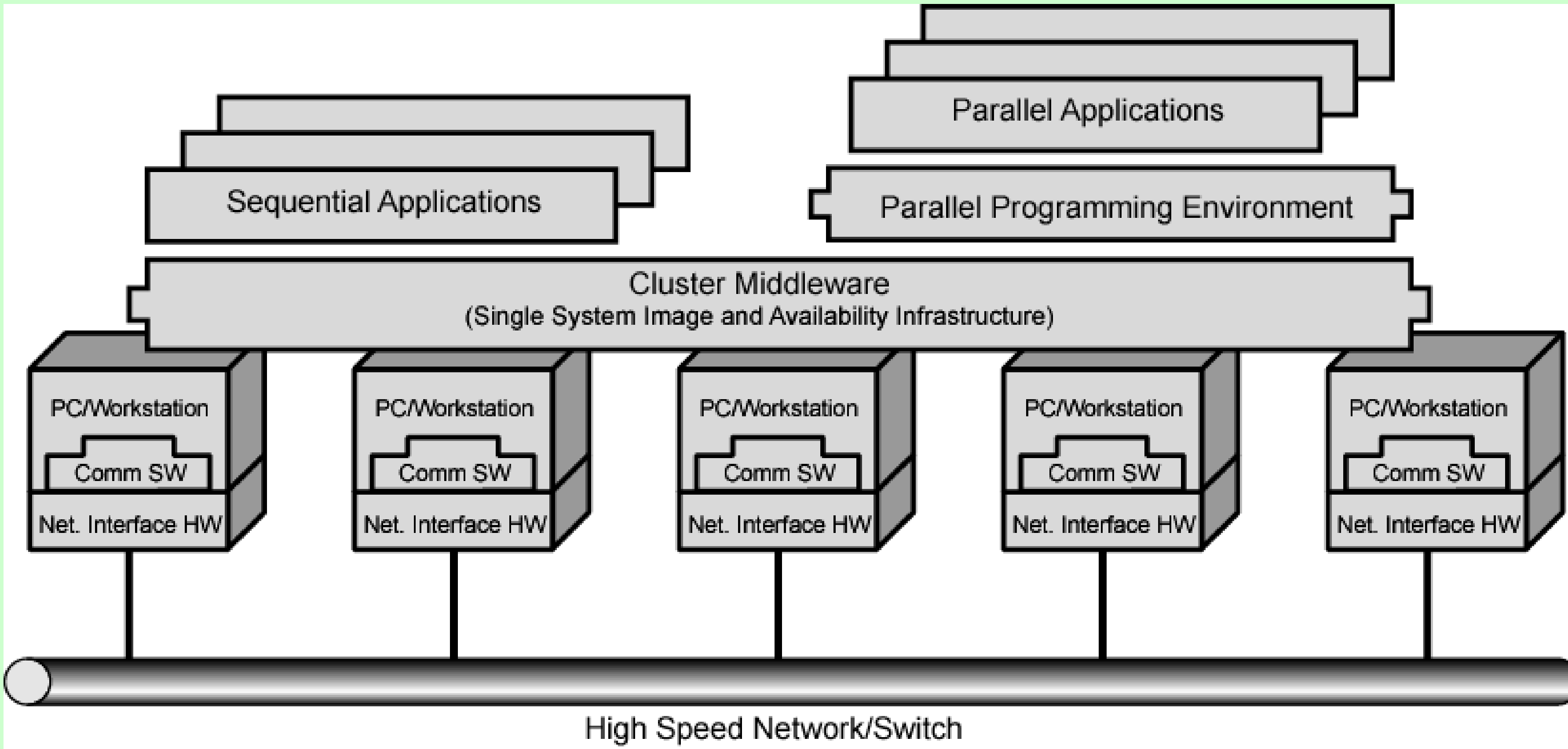
Operating Systems Design Issues

- Failure Management
  - High availability
  - Fault tolerant
  - Failover
    - Switching applications & data from failed system to alternative within cluster
  - Failback
    - Restoration of applications and data to original system
    - After problem is fixed
- Load balancing
  - Incremental scalability
  - Automatically include new computers in scheduling
  - Middleware needs to recognise that processes may switch between machines

Parallelizing

- Single application executing in parallel on a number of machines in cluster
  - Complier
    - Determines at compile time which parts can be executed in parallel
    - Split off for different computers
  - Application
    - Application written from scratch to be parallel
    - Message passing to move data between nodes
    - Hard to program
    - Best end result
  - Parametric computing
    - If a problem is repeated execution of algorithm on different sets of data
    - e.g. simulation using different scenarios
    - Needs effective tools to organize and run

Cluster Computer Architecture



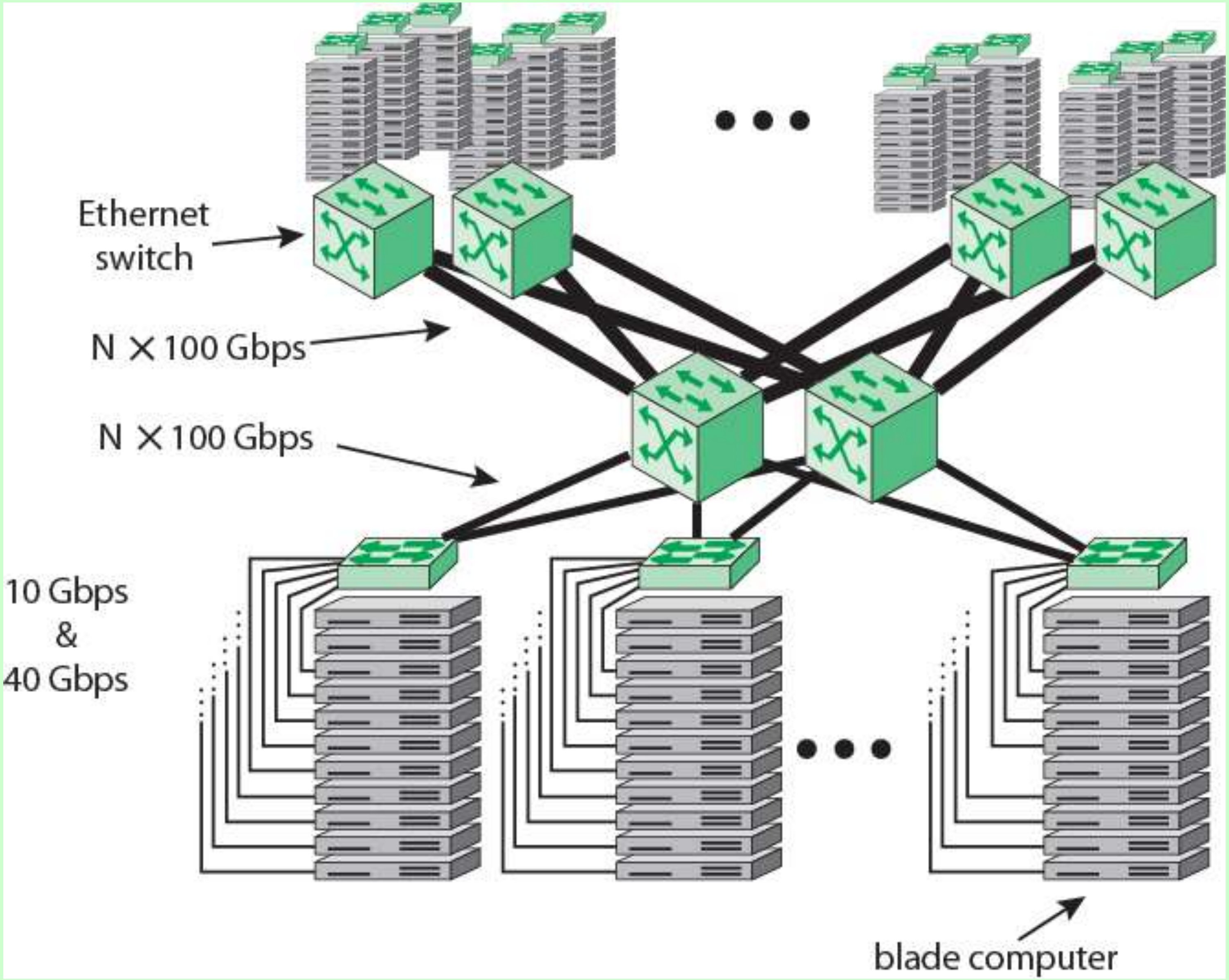
Cluster Middleware

- Unified image to user
  - Single system image
- Single point of entry
- Single file hierarchy
- Single control point
- Single virtual networking
- Single memory space
- Single job management system
- Single user interface
- Single I/O space
- Single process space
- Checkpointing
- Process migration

Blade Servers

- Common implementation of cluster
- Server houses multiple server modules (blades) in single chassis
  - Save space
  - Improve system management
  - Chassis provides power supply
  - Each blade has processor, memory, disk

Example 100-Gbps Ethernet Configuration for Massive Blade Server Site



Cluster v. SMP

- Both provide multiprocessor support to high demand applications.
- Both available commercially
  - SMP for longer
- SMP:
  - Easier to manage and control
  - Closer to single processor systems
    - Scheduling is main difference
    - Less physical space
    - Lower power consumption
- Clustering:
  - Superior incremental & absolute scalability
  - Superior availability
    - Redundancy

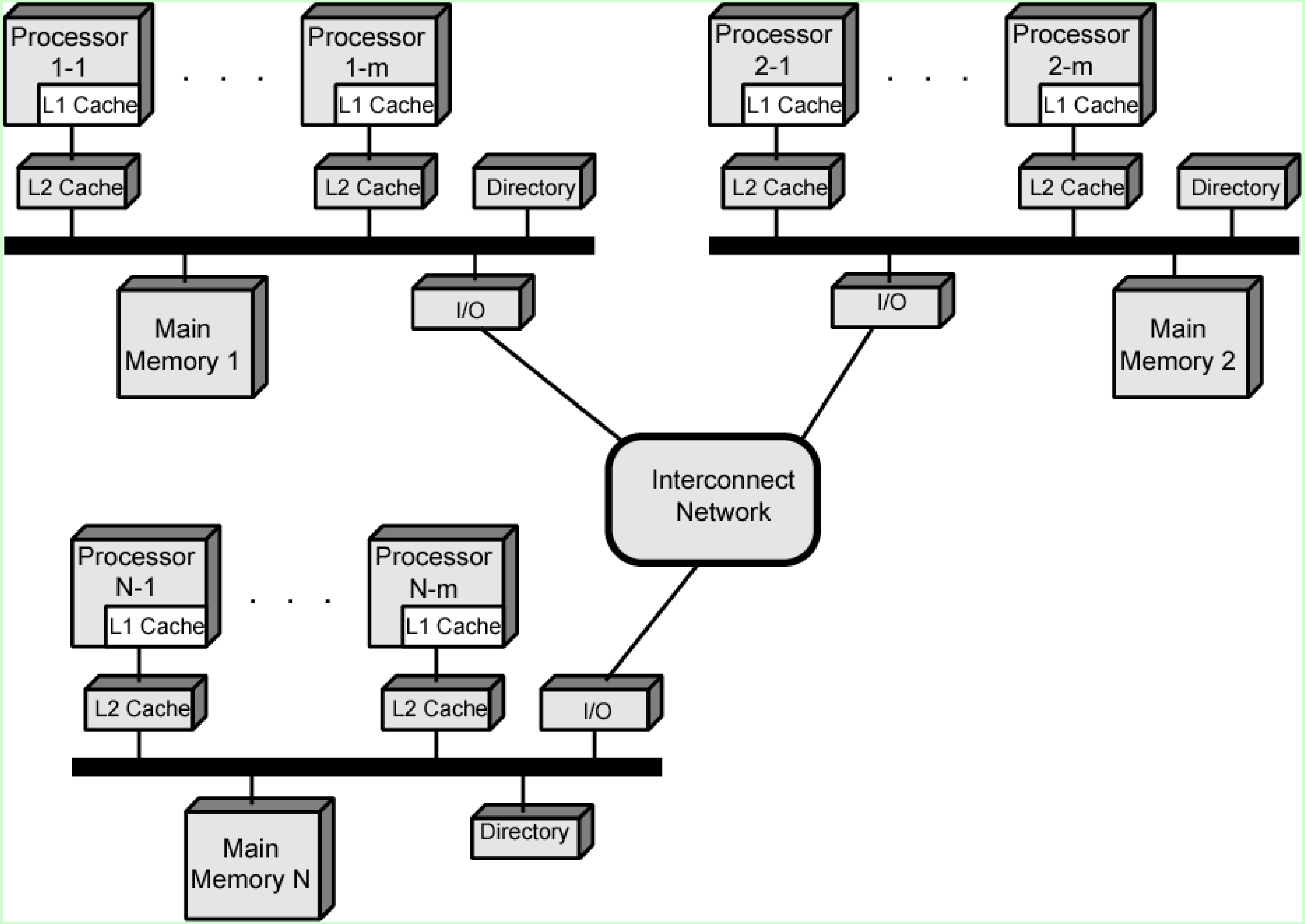
Nonuniform Memory Access (NUMA)

- Alternative to SMP & clustering
- Uniform memory access
  - All processors have access to all parts of memory
    - Using load & store
  - Access time to all regions of memory is the same
  - Access time to memory for different processors same
  - As used by SMP
- Nonuniform memory access
  - All processors have access to all parts of memory
    - Using load & store
  - Access time of processor differs depending on region of memory
  - Different processors access different regions of memory at different speeds
- Cache coherent NUMA
  - Cache coherence is maintained among the caches of the various processors
  - Significantly different from SMP and clusters

Motivation

- SMP has practical limit to number of processors
  - Bus traffic limits to between 16 and 64 processors
- In clusters each node has own memory
  - Apps do not see large global memory
  - Coherence maintained by software not hardware
- NUMA retains SMP flavour while giving large scale multiprocessing
  - e.g. Silicon Graphics Origin NUMA 1024 MIPS R10000 processors
- Objective is to maintain transparent system wide memory while permitting multiprocessor nodes, each with own bus or internal interconnection system

CC-NUMA Organization



CC-NUMA Operation

- Each processor has own L1 and L2 cache
- Each node has own main memory
- Nodes connected by some networking facility
- Each processor sees single addressable memory space
- Memory request order:
  - L1 cache (local to processor)
  - L2 cache (local to processor)
  - Main memory (local to node)
  - Remote memory
    - Delivered to requesting (local to processor) cache
- Automatic and transparent

Memory Access Sequence

- Each node maintains directory of location of portions of memory and cache status
- e.g. node 2 processor 3 (P2-3) requests location 798 which is in memory of node 1
  - P2-3 issues read request on snoopy bus of node 2
  - Directory on node 2 recognises location is on node 1
  - Node 2 directory requests node 1's directory
  - Node 1 directory requests contents of 798
  - Node 1 memory puts data on (node 1 local) bus
  - Node 1 directory gets data from (node 1 local) bus
  - Data transferred to node 2's directory
  - Node 2 directory puts data on (node 2 local) bus
  - Data picked up, put in P2-3's cache and delivered to processor

Cache Coherence

- Node 1 directory keeps note that node 2 has copy of data
- If data modified in cache, this is broadcast to other nodes
- Local directories monitor and purge local cache if necessary
- Local directory monitors changes to local data in remote caches and marks memory invalid until writeback
- Local directory forces writeback if memory location requested by another processor

NUMA Pros & Cons

- Effective performance at higher levels of parallelism than SMP
- No major software changes
- Performance can breakdown if too much access to remote memory
  - Can be avoided by:
    - L1 & L2 cache design reducing all memory access
      - Need good temporal locality of software
    - Good spatial locality of software
    - Virtual memory management moving pages to nodes that are using them most
- Not transparent
  - Page allocation, process allocation and load balancing changes needed
- Availability?

Vector Computation

- Maths problems involving physical processes present different difficulties for computation
  - Aerodynamics, seismology, meteorology
  - Continuous field simulation
- High precision
- Repeated floating point calculations on large arrays of numbers
- Supercomputers handle these types of problem
  - Hundreds of millions of flops
  - $10-15 million
  - Optimised for calculation rather than multitasking and I/O
  - Limited market
    - Research, government agencies, meteorology
- Array processor
  - Alternative to supercomputer
  - Configured as peripherals to mainframe & mini
  - Just run vector portion of problems

Vector Addition Example

$$\begin{bmatrix} 1.5 \\ 7.1 \\ 6.9 \\ 100.5 \\ 0 \\ 59.7 \end{bmatrix} + \begin{bmatrix} 2.0 \\ 39.7 \\ 1000.003 \\ 11 \\ 21.1 \\ 19.7 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 46.8 \\ 1006.903 \\ 111.5 \\ 21.1 \\ 79.4 \end{bmatrix}$$

$$A \quad + \quad B \quad = \quad C$$

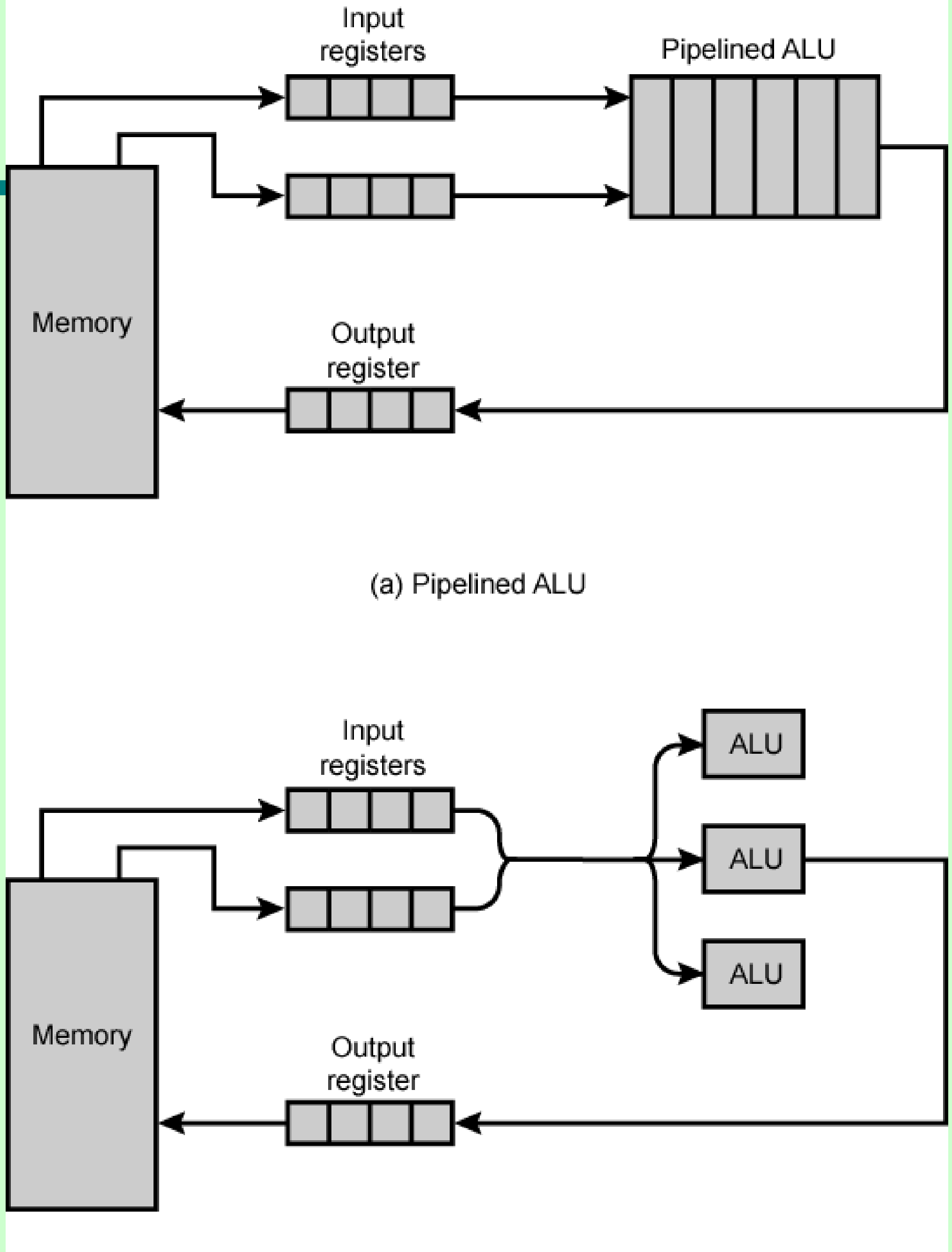
Approaches

- General purpose computers rely on iteration to do vector calculations
- In example this needs six calculations
- Vector processing
  - Assume possible to operate on one-dimensional vector of data
  - All elements in a particular row can be calculated in parallel
- Parallel processing
  - Independent processors functioning in parallel
  - Use FORK N to start individual process at location N
  - JOIN N causes N independent processes to join and merge following JOIN
    - O/S Co-ordinates JOINs
    - Execution is blocked until all N processes have reached JOIN

Processor Designs

- Pipelined ALU
  - Within operations
  - Across operations
- Parallel ALUs
- Parallel processors

Approaches to Vector Computation



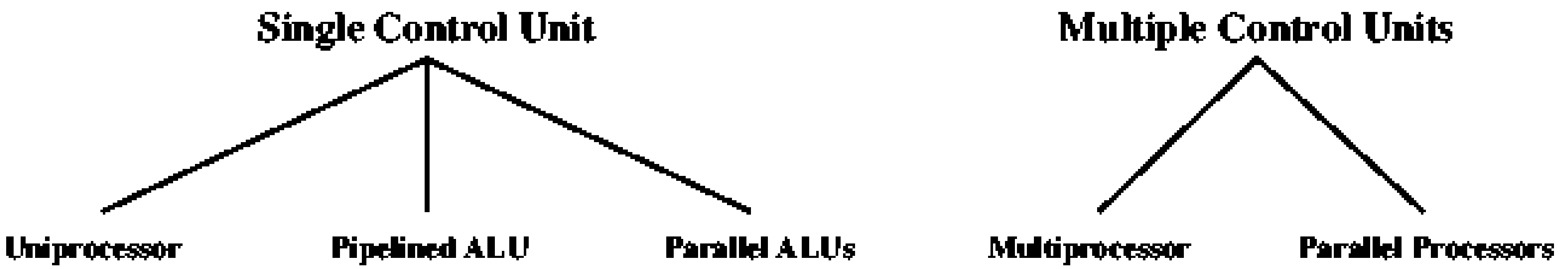
(a) Pipelined ALU

(b) Parallel ALUs

Chaining

- Cray Supercomputers
- Vector operation may start as soon as first element of operand vector available and functional unit is free
- Result from one functional unit is fed immediately into another
- If vector registers used, intermediate results do not have to be stored in memory

Computer Organizations



IBM 3090 with Vector Facility