

Parallel Processing - 02

Md. Biplob Hosen

Lecturer, IIT-JU

Email: biplob.hosen@juniv.edu

Overview

- The two most common multiple-processor organizations are **symmetric multiprocessors (SMPs)** and **clusters**. More recently, **non-uniform memory access (NUMA)** systems have been introduced commercially.
- An **SMP** consists of multiple similar processors within the same computer, interconnected by a bus or some sort of switching arrangement.
- The most critical problem to address in an SMP is that of **cache coherence**.
- When more than one processor are implemented on a single chip, the configuration is referred to as **chip multiprocessing**.
- A related design scheme is to replicate some of the components of a single processor so that the processor can execute multiple threads concurrently; this is known as a **multithreaded processor**.
- A **cluster** is a group of interconnected, whole computers working together as a unified computing resource that can create the illusion of being one machine.

Parallel Organizations

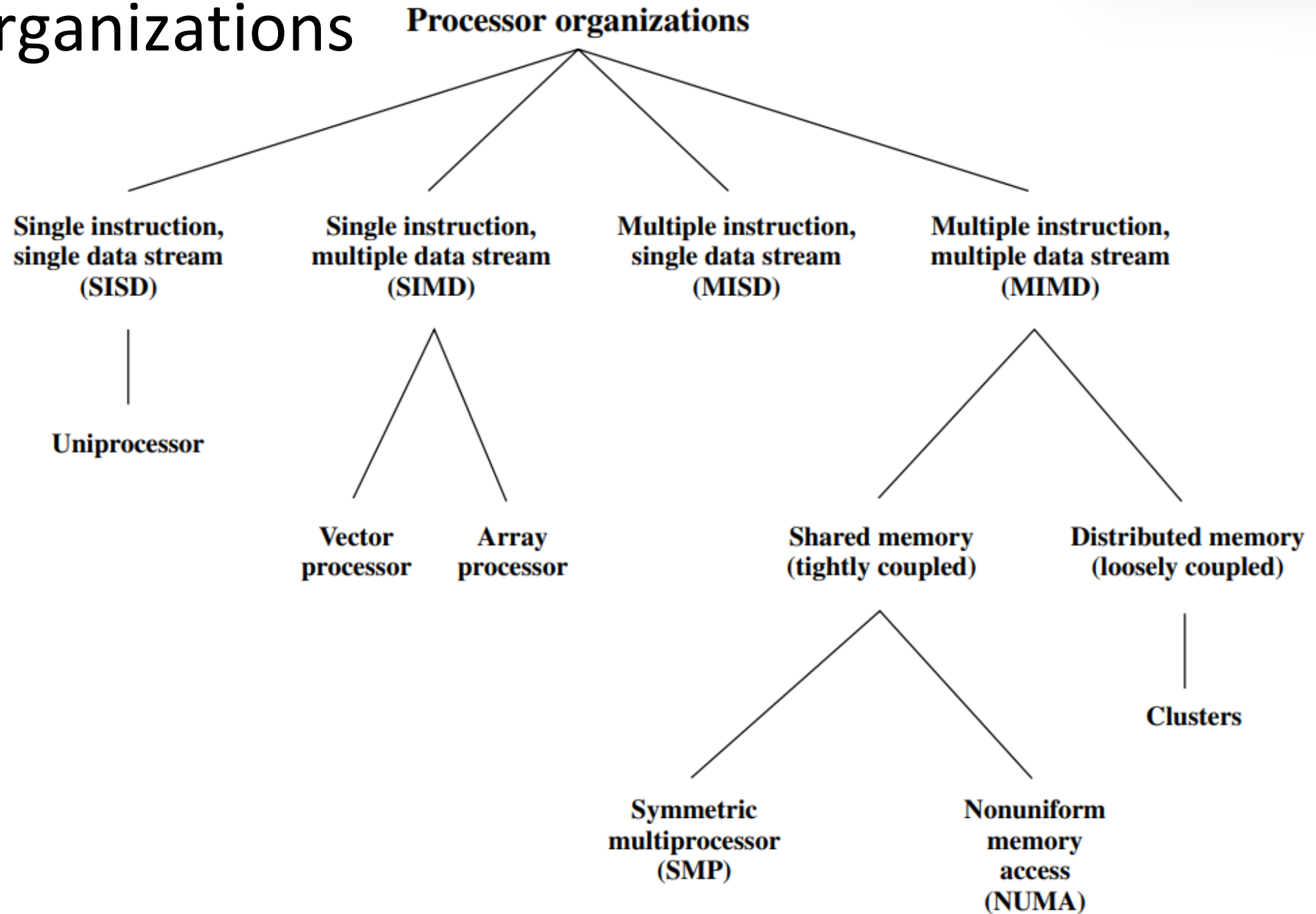
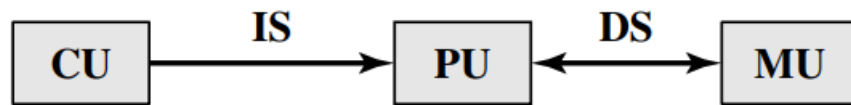


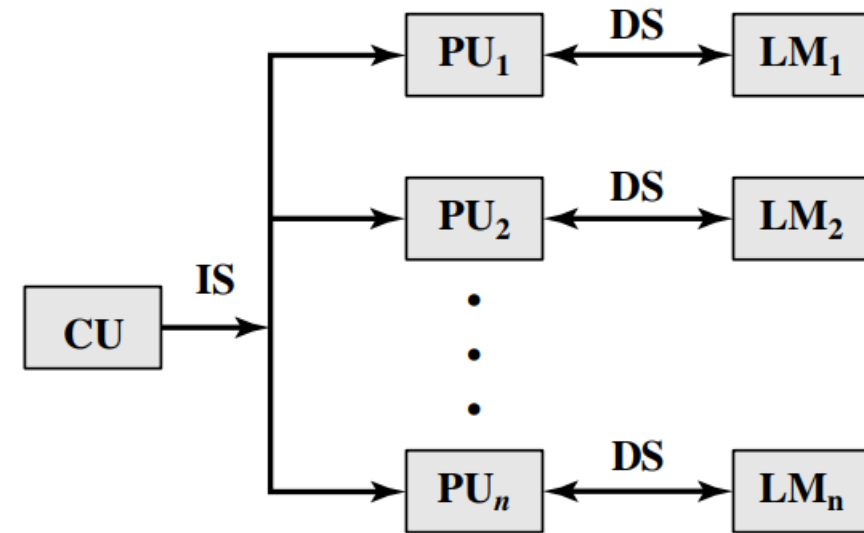
Figure A Taxonomy of Parallel Processor Architectures

Parallel Organizations

- Figure-a shows the structure of an SISD. There is some sort of control unit (CU) that provides an instruction stream (IS) to a processing unit (PU). The processing unit operates on a single data stream (DS) from a memory unit (MU). With an SIMD, there is still a single control unit, now feeding a single instruction stream to multiple PUs. Each PU may have its own dedicated memory (Figure-b), or there may be a shared memory.



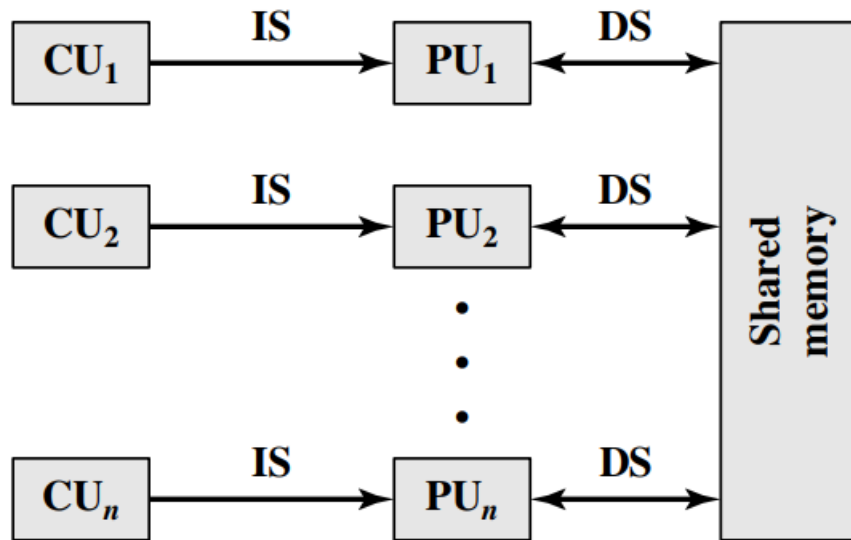
(a) SISD



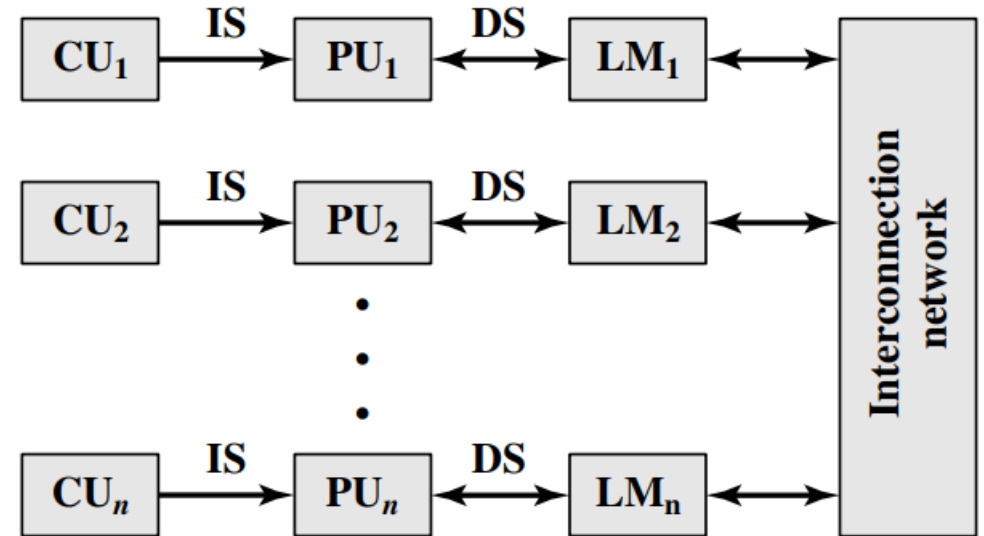
(b) SIMD (with distributed memory)

Continue...

- Finally, with the MIMD, there are multiple control units, each feeding a separate instruction stream to its own PU. The MIMD may be a shared-memory multiprocessor (Figure-c) or a distributed-memory multicomputer (Figure-d).



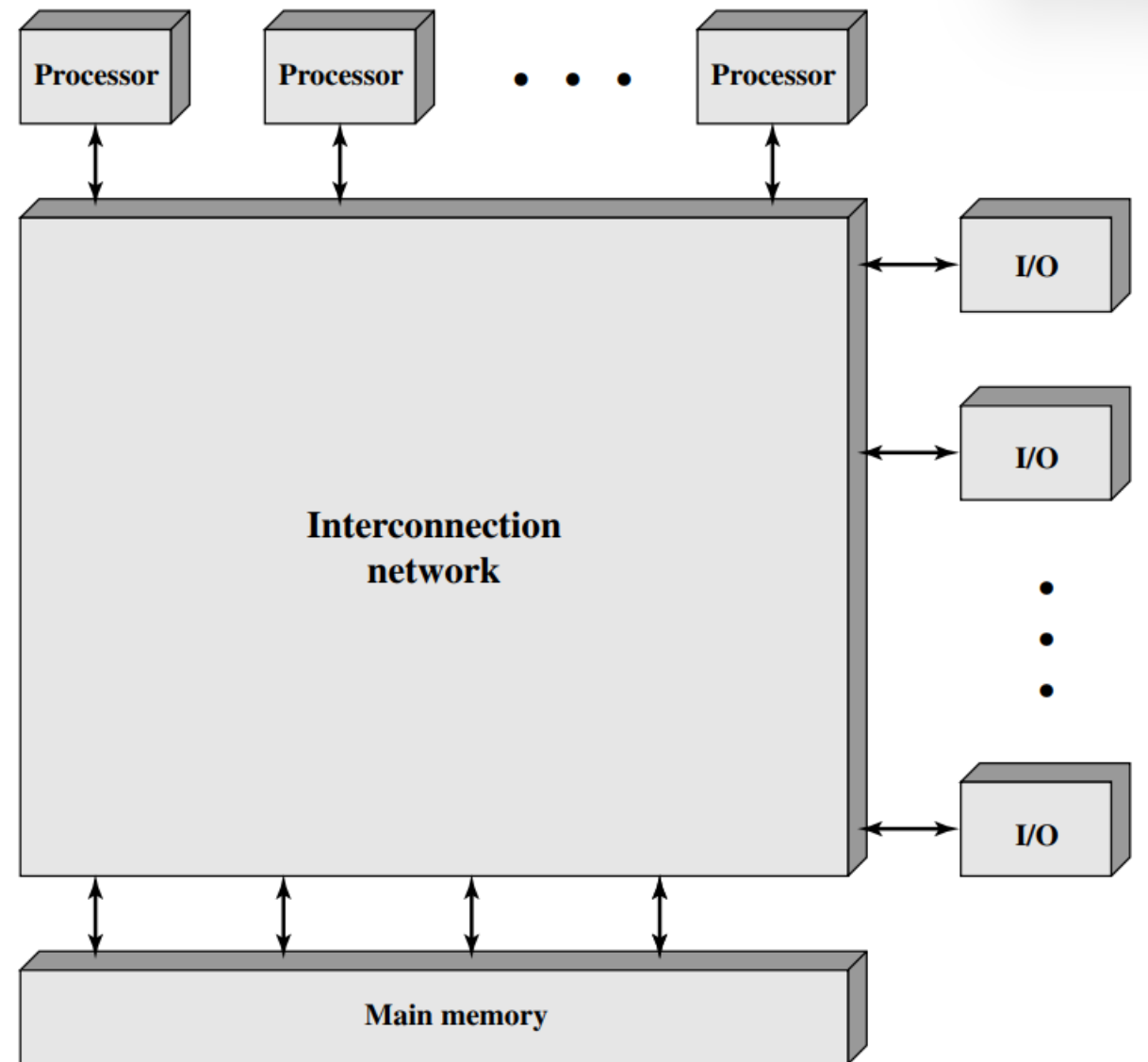
(c) MIMD (with shared memory)



(d) MIMD (with distributed memory)

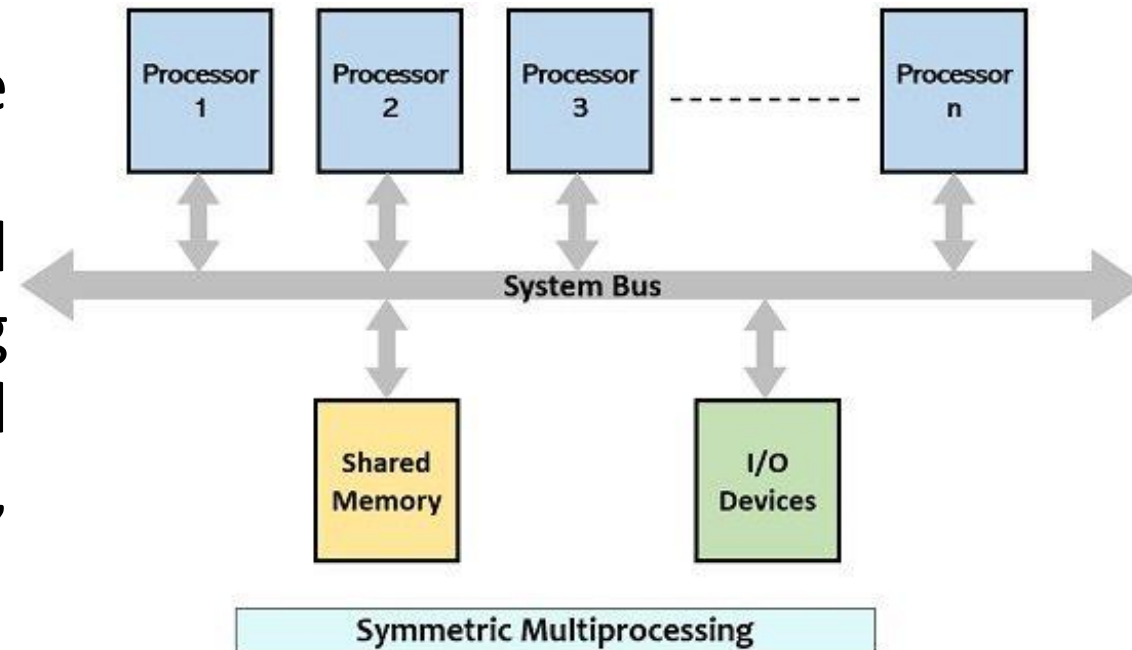
Block Diagram of Tightly Coupled Multiprocessor

- There are two or more processors.
- Each processor is self-contained, including a control unit, ALU, registers, and, typically, one or more levels of cache.
- Each processor has access to a shared main memory and the I/O devices through some form of interconnection mechanism.



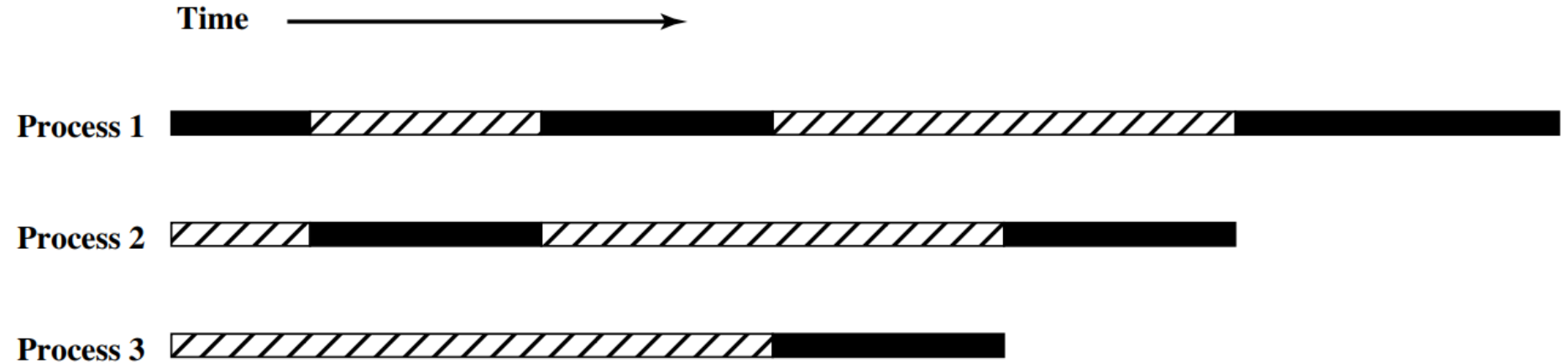
Symmetric Multiprocessors

- A stand alone computer with the following characteristics:
 - 1) Two or more similar processors of comparable capacity.
 - 2) 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.
 - 3) All processors share access to I/O: Either through same channels or different channels giving paths to same devices.
 - 4) All processors can perform the same functions (hence symmetric).
 - 5) System controlled by integrated operating system providing interaction between processors and their programs at the job, task, file, and data element levels.

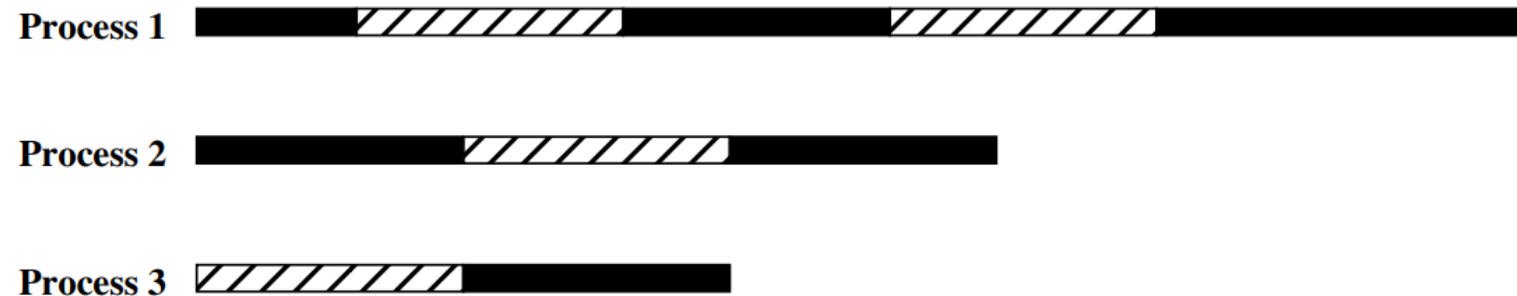


SMP - Advantages

- **Performance:** If the work to be done by a computer can be organized so that some portions of the work can be done in parallel, then a system with multiple processors will yield greater performance than one with a single processor of the same type.



(a) Interleaving (multiprogramming, one processor)



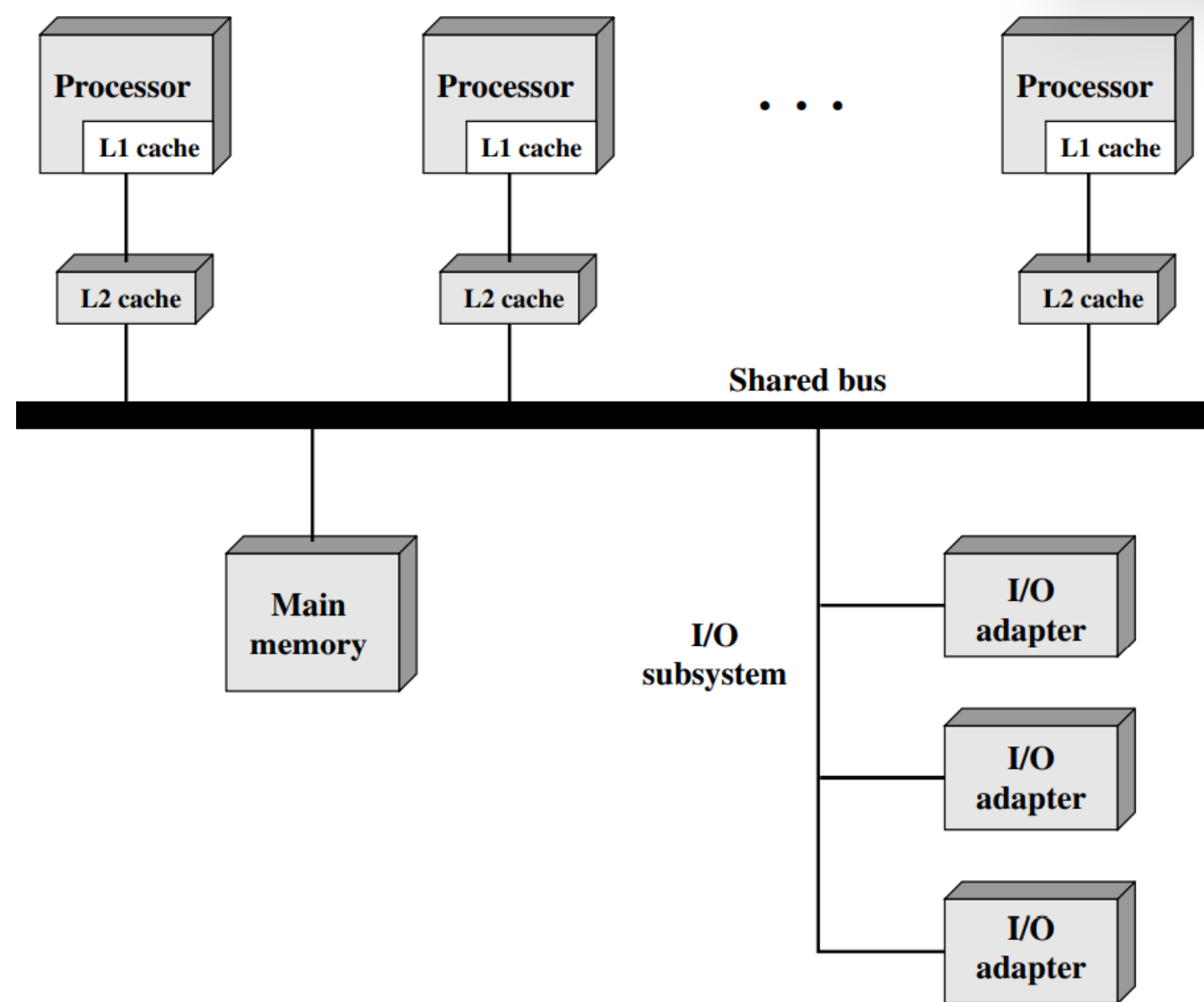
(b) Interleaving and overlapping (multiprocessing; multiple processors)

SMP - Advantages

- **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.

Symmetric Multiprocessor Organization

- The most common organization for personal computers, workstations, and servers is the time-shared bus.
- The time-shared bus is the simplest mechanism for constructing a multiprocessor system.
- The bus consists of control, address, and data lines.

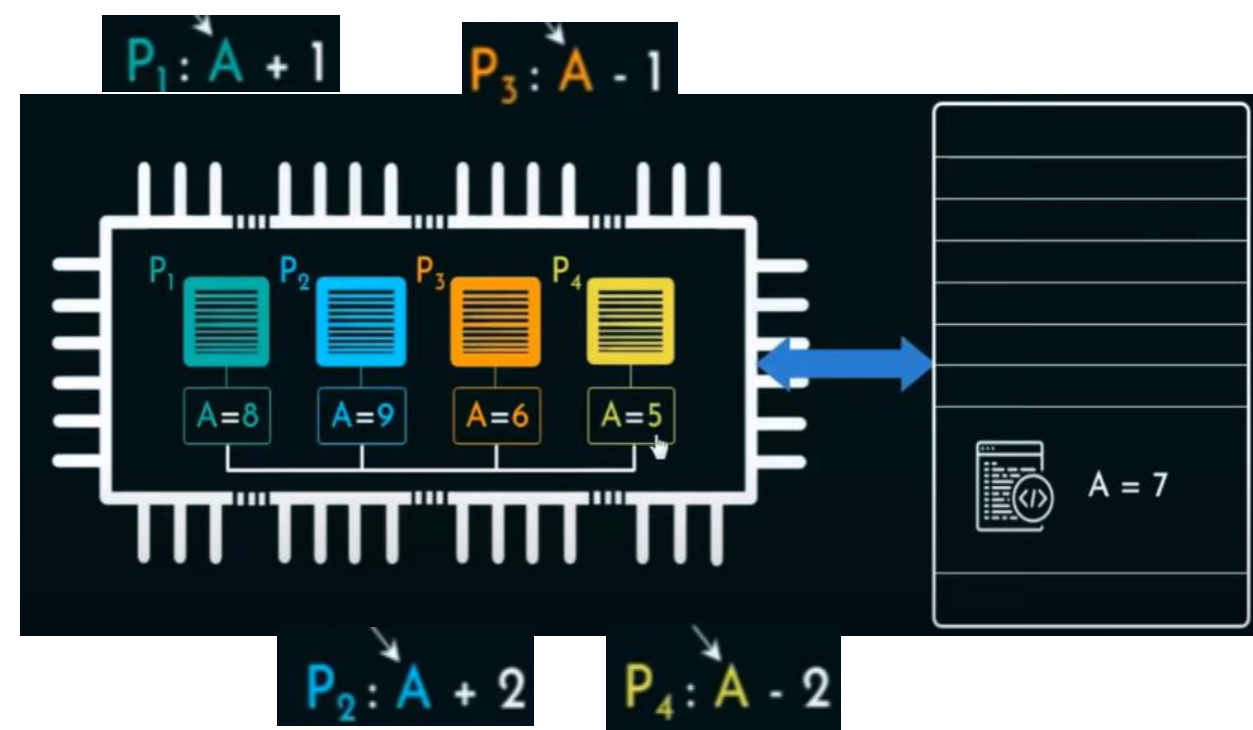


Multiprocessor OS Design

- An SMP operating system manages processor and other computer resources so that the user perceives a single operating system controlling system resources.
- In both the SMP and uniprocessor cases, multiple jobs or processes may be active at one time, and it is the responsibility of the operating system to schedule their execution and to allocate resources.
- A user may construct applications that use multiple processes or multiple threads within processes without regard to whether a single processor or multiple processors will be available.
- Thus a multiprocessor operating system must provide all the functionality of a multiprogramming system plus additional features to accommodate multiple processors.
- **Key design issues:** i) Simultaneous concurrent processes; ii) Scheduling; iii) Synchronization; iv) Memory management; v) Reliability and fault tolerance.

Cache Coherence

- **Problem** - Multiple copies of same data in different caches. Can result in an inconsistent view of memory.



- **Write back:** Write operations are usually made only to the cache. Main memory is only updated when the corresponding cache line is flushed from the cache.
 - This policy can lead to inconsistency.
- **Write through:** All write operations are made to main memory as well as to the cache, ensuring that main memory is always valid.
 - Write through can also give problems unless caches monitor memory traffic.

Solutions

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 Solutions

- Cache coherence protocols.
- Dynamic recognition of potential problems at run time.
- More efficient use of cache.
- Transparent to programmer.
- Two categories – i) Directory protocols; ii) Snoopy protocols.

Directory Protocols

- Collect and maintain information about copies of data in cache.
- There is a **centralized controller**.
- Directory stored in main memory.
- The directory contains global state information about the contents of the various local caches.
- Requests are checked against directory.
- Appropriate transfers are performed.
- Creates central bottleneck.
- Overhead of communication between the various cache controllers and the central controller.
- 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.
- Two basic approaches to the snoopy protocol have been explored:
 - i) Write invalidate; ii) Write update.

Continue...

Write Update:

- Multiple readers and writers.
- Updated word is distributed to all other processors.

Write Invalidate:

- Multiple readers, one writer.
- When a write is required, all other caches of the line are invalidated.
- Writing processor then has exclusive access until line required by another processor.
- Most widely used in commercial multiprocessor systems, such as the Pentium 4 and PowerPC.
- It marks the state of every cache line (using two extra bits in the cache tag) as modified, exclusive, shared, or invalid. For this reason, the write-invalidate protocol is called **MESI**.

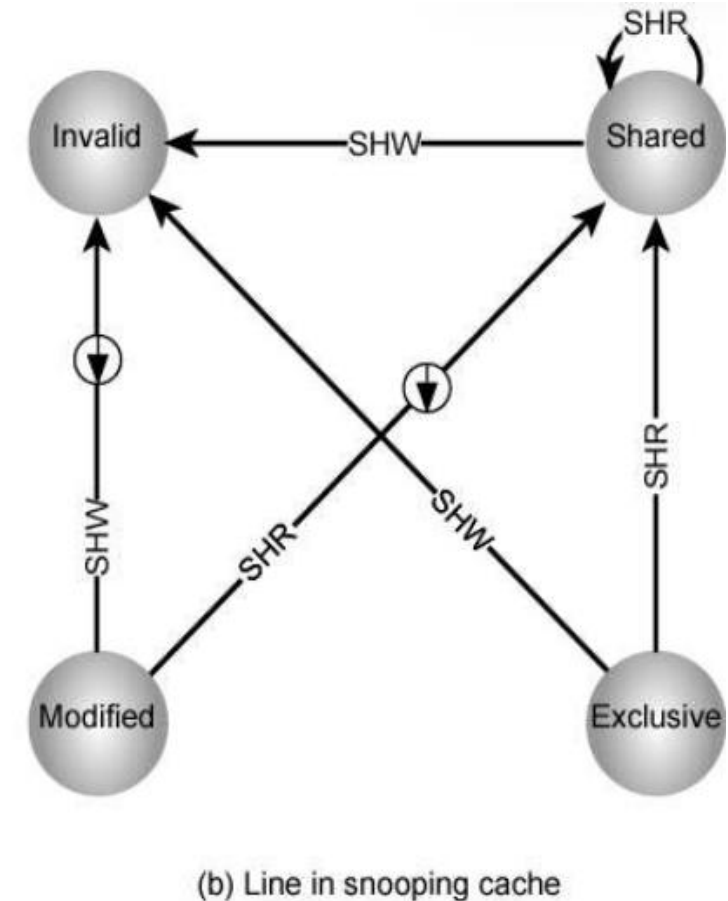
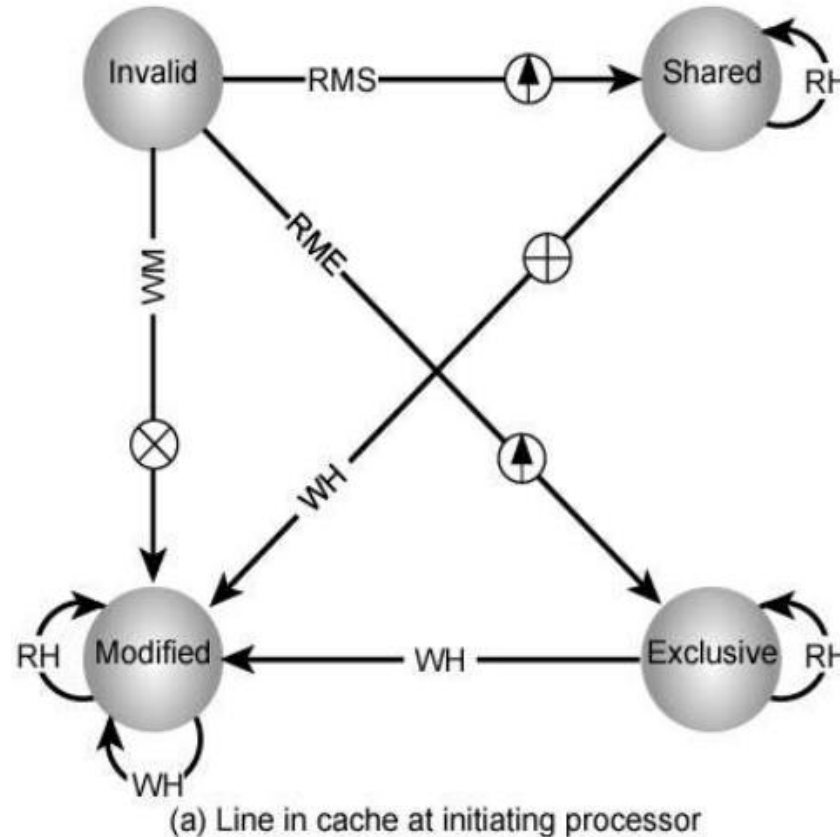
The MESI Protocol





- To provide cache consistency on an SMP, the data cache often supports a protocol known as MESI. For MESI, the data cache includes two status bits per tag, so that each line can be in one of four states:
- **Modified:** The line in the cache has been modified (different from main memory) and is available only in this cache.
- **Exclusive:** The line in the cache is the same as that in main memory and is not present in any other cache.
- **Shared:** The line in the cache is the same as that in main memory and may be present in another cache.
- **Invalid:** The line in the cache does not contain valid data.

	M Modified	E Exclusive	S Shared	I Invalid
This cache line valid?	Yes	Yes	Yes	No
The memory copy is ...	out of date	valid	valid	—
Copies exist in other caches?	No	No	Maybe	Maybe
A write to this line ...	does not go to bus	does not go to bus	goes to bus and updates cache	goes directly to bus

MESI State Transition Diagram

- Figure-a shows the transitions that occur due to actions initiated by the processor attached to this cache.
- Figure-b shows the transitions that occur due to events that are snooped on the common bus.

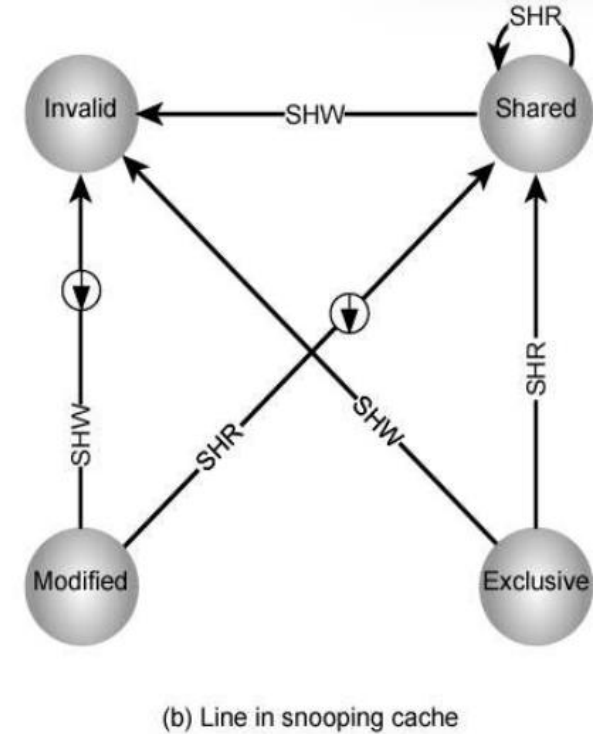
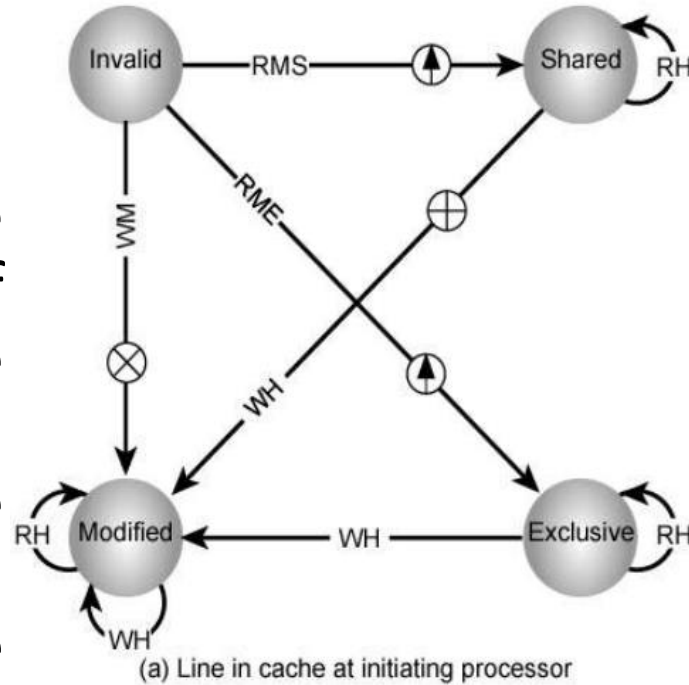






RH	Read hit		Dirty line copyback
RMS	Read miss, shared		
RME	Read miss, exclusive		Invalidate transaction
WH	Write hit		
WM	Write miss		Read-with-intent-to-modify
SHR	Snoop hit on read		
SHW	Snoop hit on write or read-with-intent-to-modify		Cache line fill

Continue...

READ MISS: There are a number of possible outcomes:

1. If one other cache has a clean (unmodified since read from memory) copy of the line in the exclusive state (**RMS+SHR**).
2. If one or more caches have a clean copy of the line in the shared state (**RMS+SHR**).
3. If one other cache has a modified copy of the line (**SHW**).
4. If no other cache has a copy of the line (**RME**).

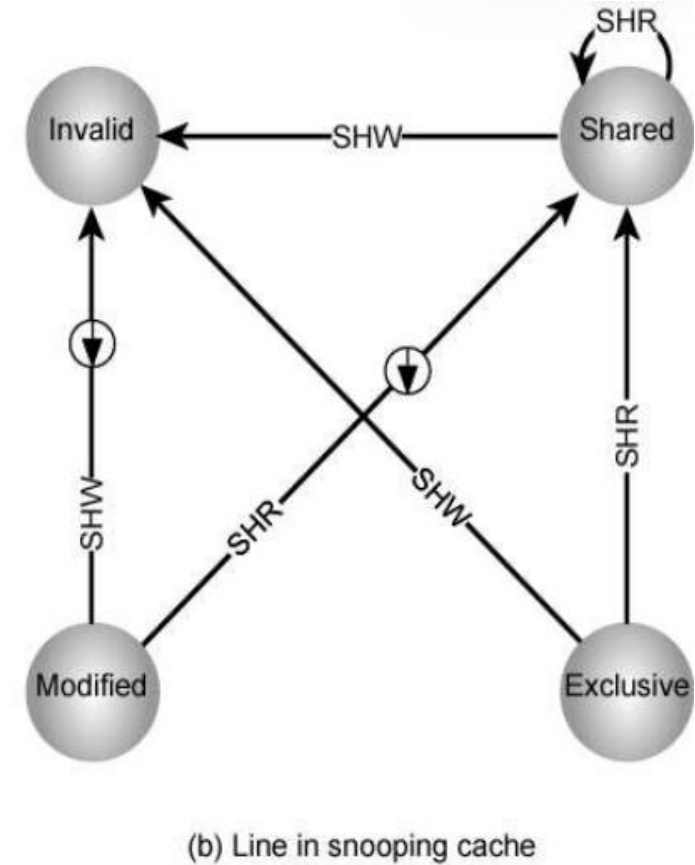
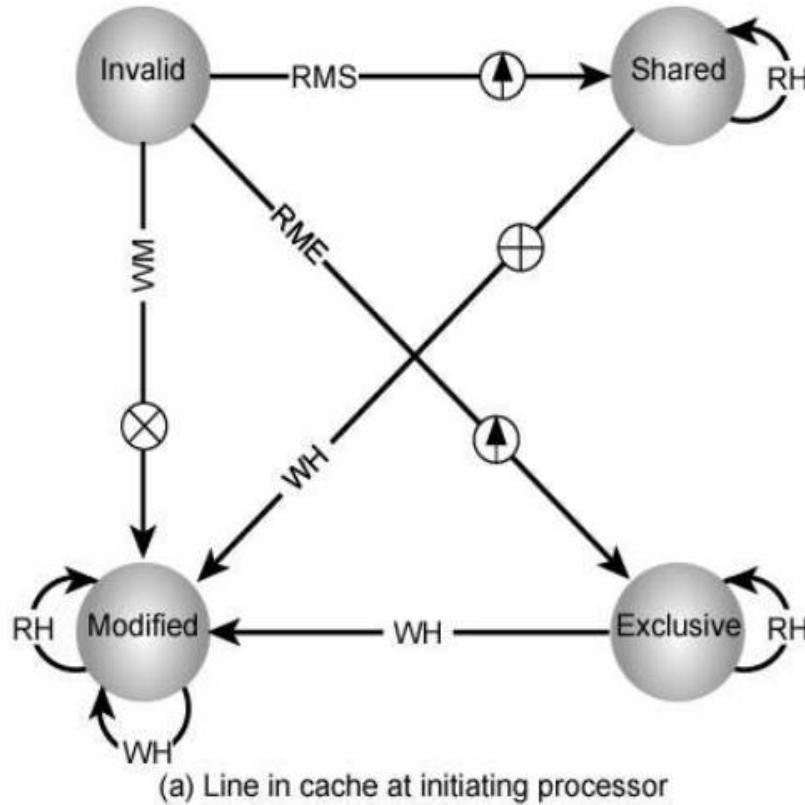






RH	Read hit		Dirty line copyback
RMS	Read miss, shared		
RME	Read miss, exclusive		Invalidate transaction
WH	Write hit		
WM	Write miss		Read-with-intent-to-modify
SHR	Snoop hit on read		
SHW	Snoop hit on write or read-with-intent-to-modify		Cache line fill

Continue...

READ HIT:

- When a read hit occurs on a line currently in the local cache, the processor simply reads the required item. There is no state change: The state remains modified, shared, or exclusive.

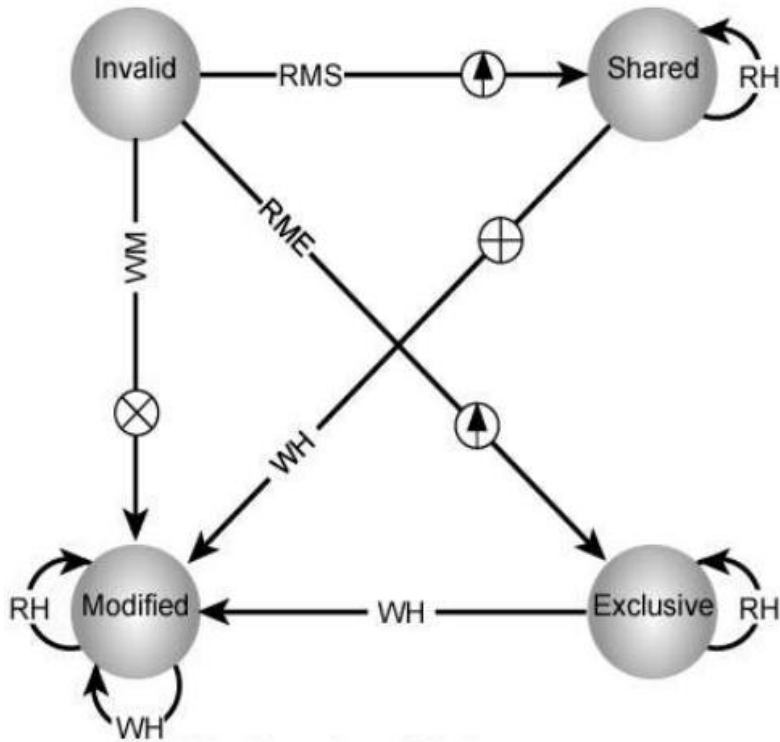


RH	Read hit		Dirty line copyback
RMS	Read miss, shared		
RME	Read miss, exclusive		Invalidate transaction
WH	Write hit		
WM	Write miss		Read-with-intent-to-modify
SHR	Snoop hit on read		
SHW	Snoop hit on write or read-with-intent-to-modify		Cache line fill

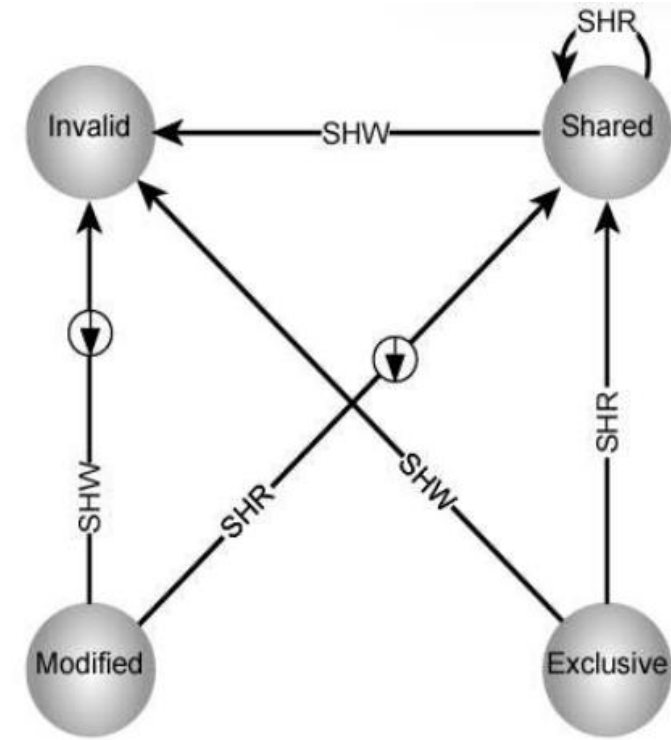
Continue...

WRITE MISS: read-with-intent-to-modify (RWITM).





1. Some other cache may have a modified copy of this line (**WM+SHW**).
2. No other cache has a modified copy of the requested line (**WM**).



(a) Line in cache at initiating processor



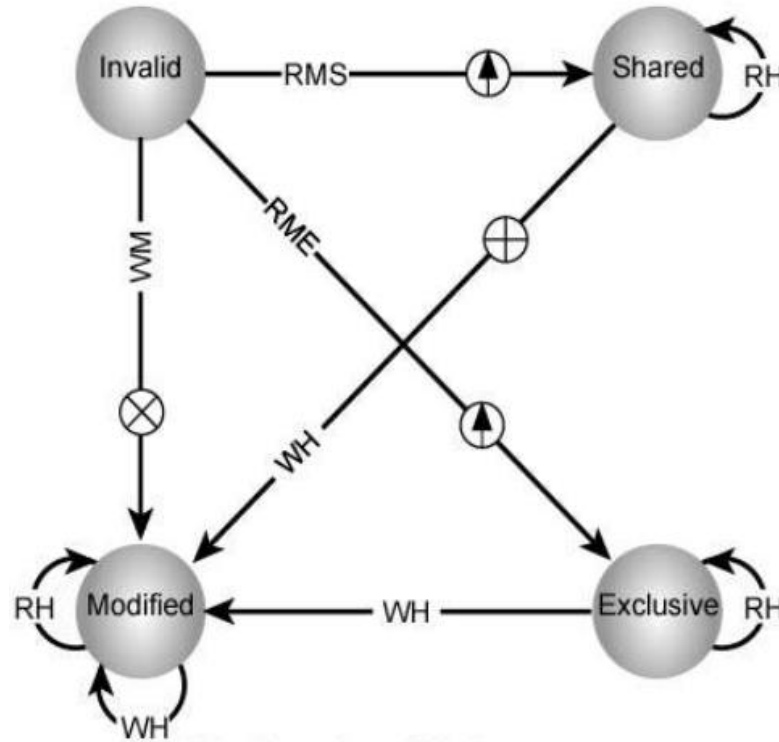
(b) Line in snooping cache

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

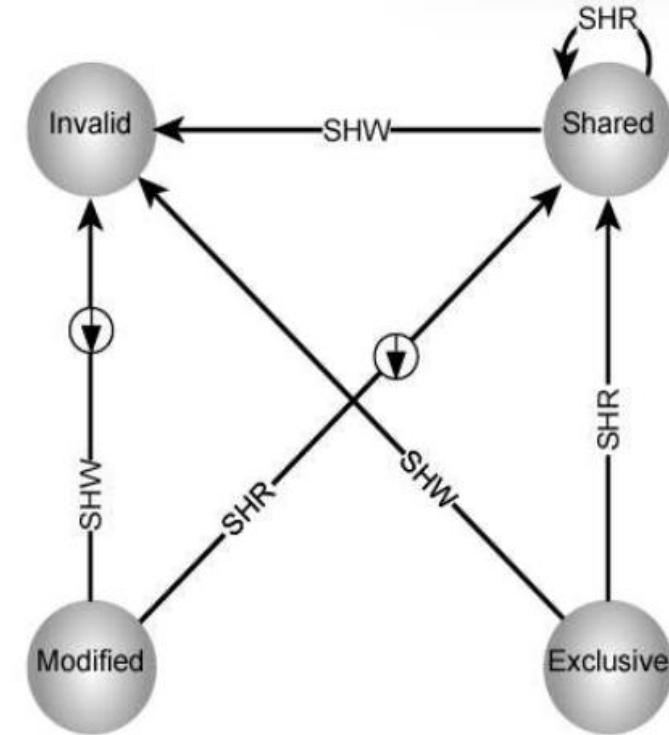
Continue...

WRITE HIT:

1. The current state of that line in the local cache – Shared **(WH+SHW)**.
2. The current state of that line in the local cache – Exclusive **(WH)**.
3. The current state of that line in the local cache – Modified **(WH)**.



(a) Line in cache at initiating processor



(b) Line in snooping cache

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		

Increasing Performance

- Processor performance can be measured by the rate at which it executes instructions:
- $\text{MIPS_rate} = f * \text{IPC}$
 - 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

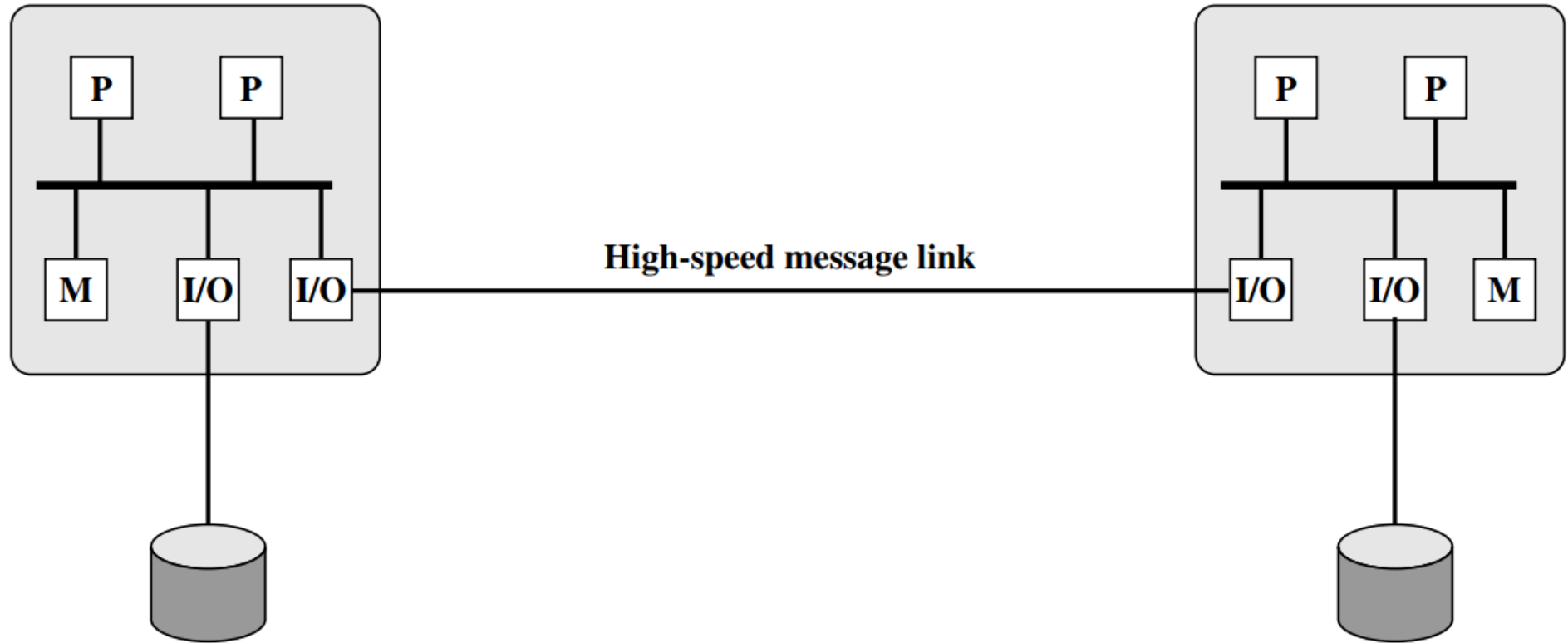
Clusters

- Alternative to SMP
- High performance
- High availability
- Server applications
- Group of interconnected computers working as unified resource.
- Illusion of being one machine.
- Each computer called a node.

Cluster Benefits:

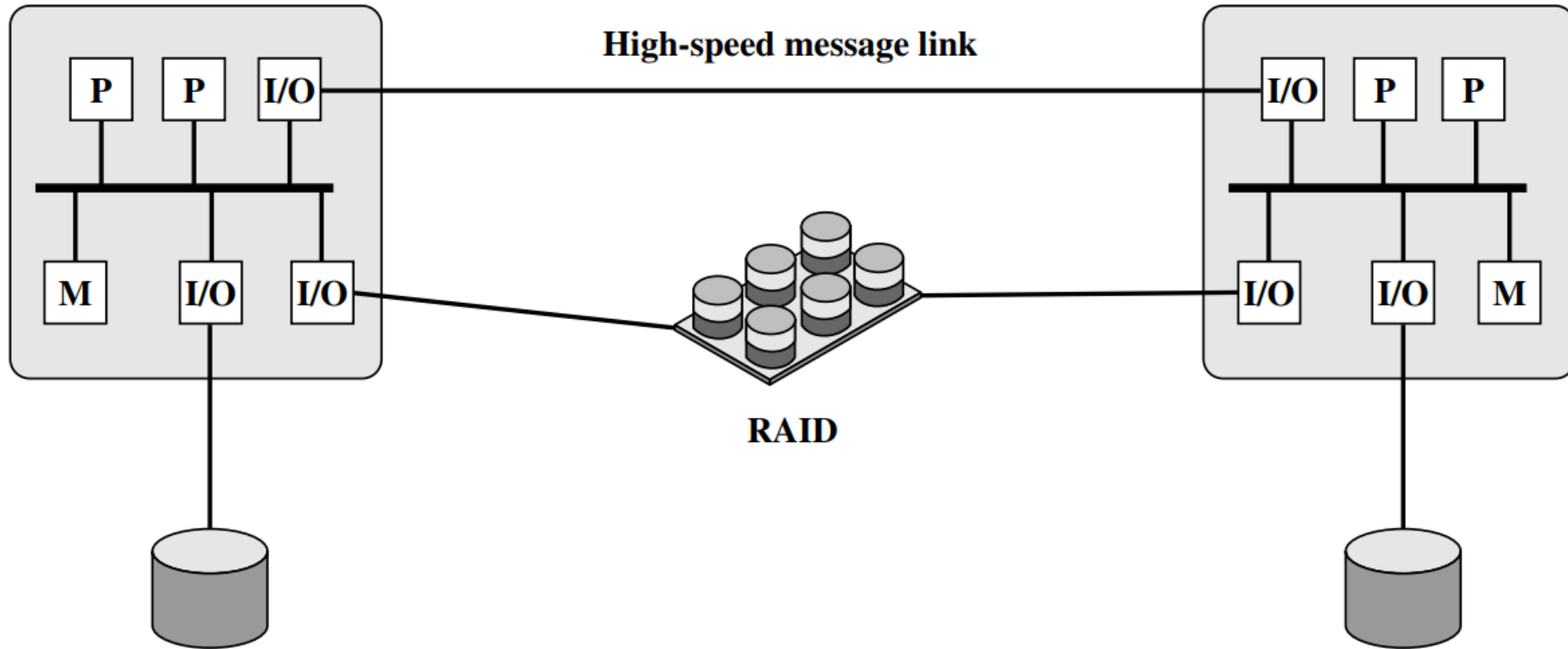
- Absolute scalability
- Incremental scalability
- High availability
- Superior price/performance

Cluster Configurations



(a) Standby server with no shared disk

Cluster Configurations



(b) Shared Disk

Clustering Methods: Benefits and Limitations

Clustering Method	Description	Benefits	Limitations
Passive Standby	A secondary server takes over in case of primary server failure.	Easy to implement.	High cost because the secondary server is unavailable for other processing tasks.
Active Secondary:	The secondary server is also used for processing tasks.	Reduced cost because secondary servers can be used for processing.	Increased complexity.
Separate Servers	Separate servers have their own disks. Data is continuously copied from primary to secondary server.	High availability.	High network and server overhead due to copying operations.
Servers Connected to Disks	Servers are cabled to the same disks, but each server owns its disks. If one server fails, its disks are taken over by the other server.	Reduced network and server overhead due to elimination of copying operations.	Usually requires disk mirroring or RAID technology to compensate for risk of disk failure.
Servers Share Disks	Multiple servers simultaneously share access to disks.	Low network and server overhead. Reduced risk of downtime caused by disk failure.	Requires lock manager software. Usually used with disk mirroring or RAID technology.

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 recognize that processes may switch between machines.

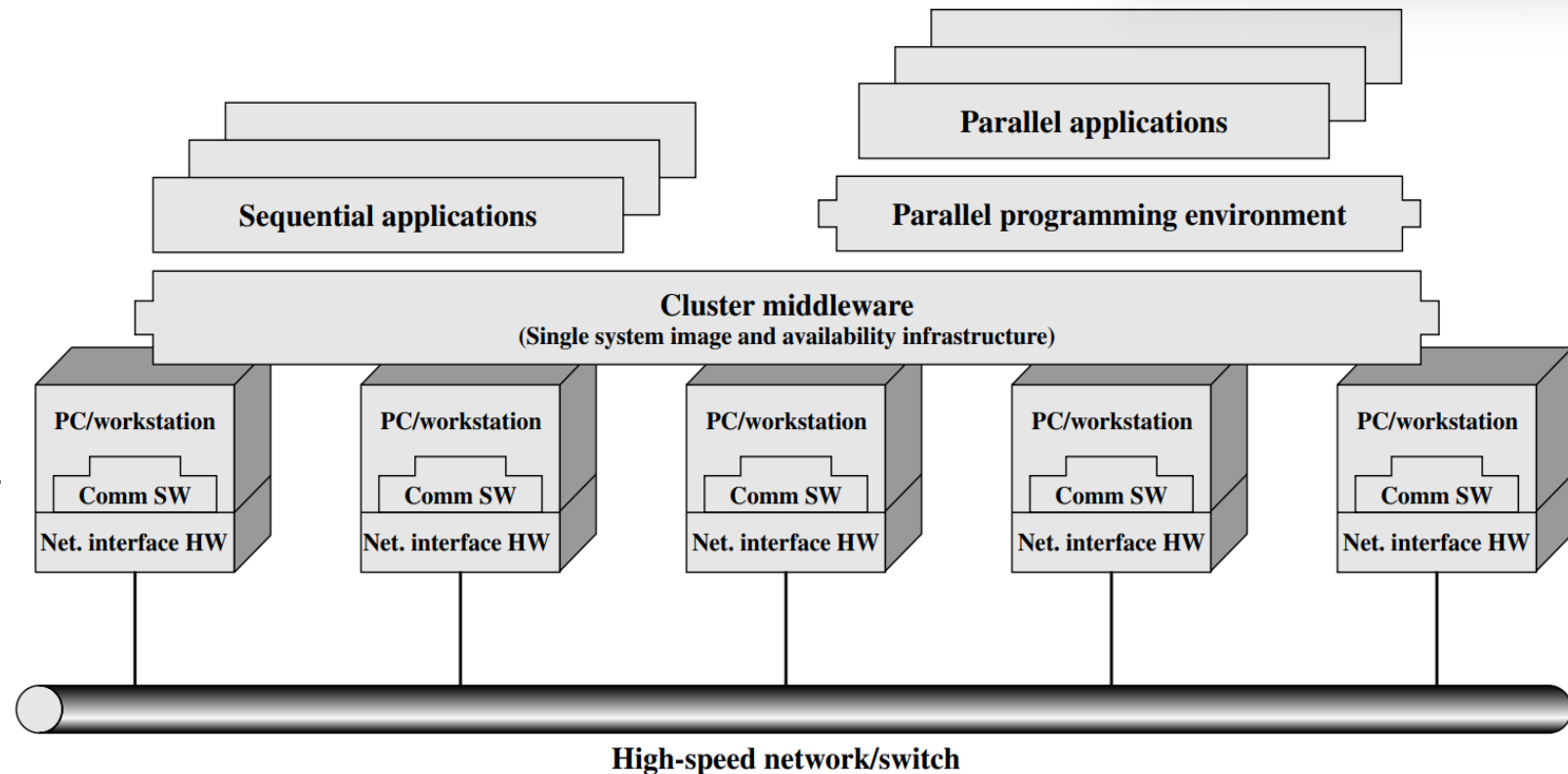
Continue...

Parallelizing: Single application executing in parallel on a number of machines in cluster.

- **Compiler:**
 - 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

- The individual computers are connected by some high-speed LAN or switch hardware.
- Each computer is capable of operating independently.



- In addition, a **middleware layer of software** is installed in each computer to enable cluster operation. The cluster middleware provides a unified system image to the user, known as a **single-system image**. The middleware is also responsible for providing high availability, by means of load balancing and responding to failures in individual components.

Blade Servers

- A common implementation of the cluster approach is the blade server.
- A blade server is a server architecture that houses multiple server modules (“blades”) in a single chassis.
- It is widely used in data centers to save space and improve system management. Either self-standing or rack mounted, the chassis provides the power supply, and each blade has its own processor, memory, and hard disk.
- The trend at large data centers, with substantial banks of blade servers, is the deployment of 10-Gbps ports on individual servers to handle the massive multimedia traffic provided by these servers.
- A 100-Gbps rate provides the bandwidth required to handle the increased traffic load. The 100-Gbps Ethernet switches are deployed in switch uplinks inside the data center as well as providing inter-building, intercampus, wide area connections for enterprise networks.

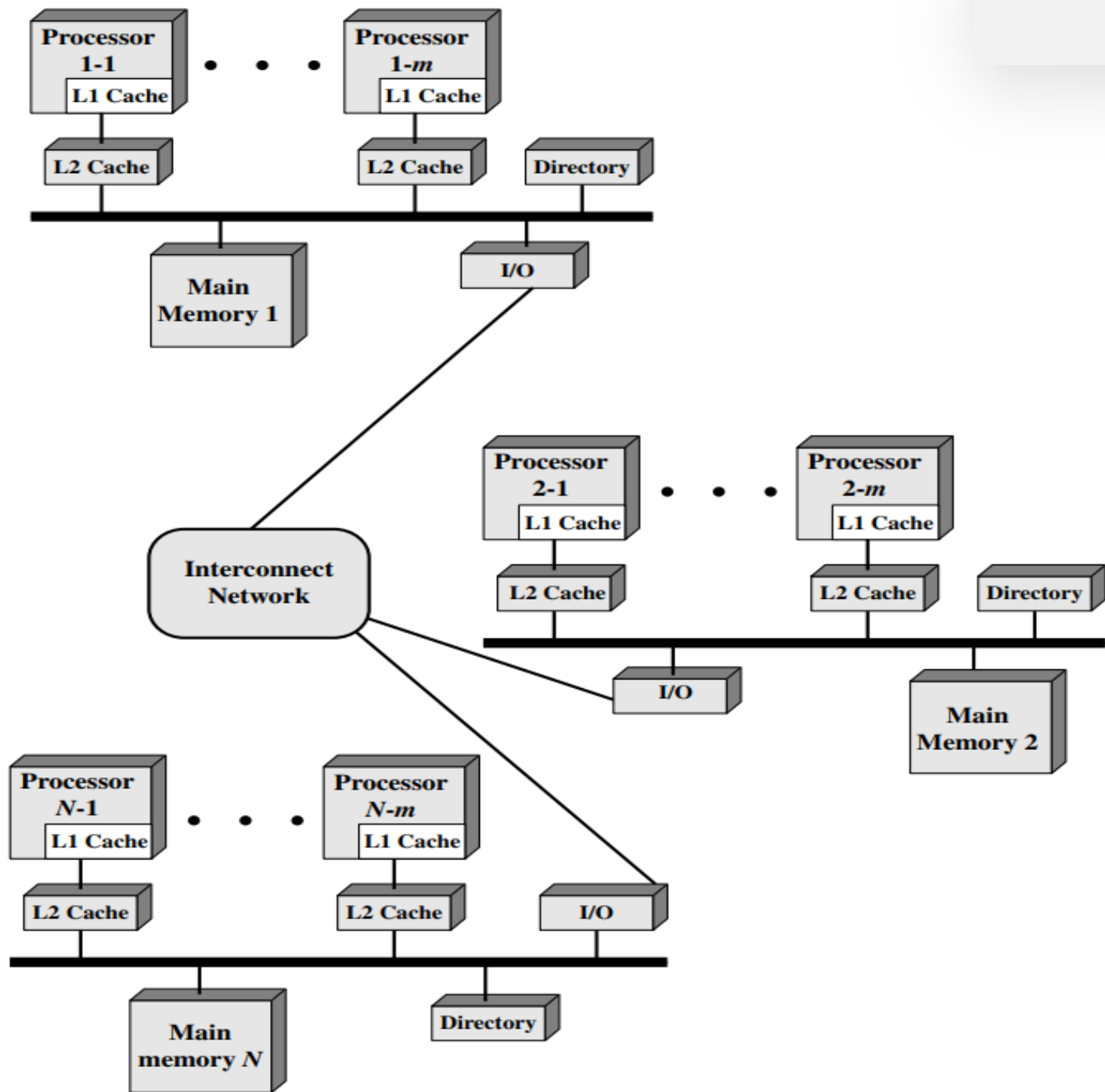
Non-uniform Memory Access (NUMA)

- Alternative to SMP & clustering.
- Uniform memory access: As used by SMP.
 - 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.
- Non-uniform 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 flavor while giving large scale multiprocessing.
- 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
- 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 recognizes 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 write-back.
- Local directory forces write-back 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.

Practice Problems

- **Book:** Computer Organization and Architecture: Designing for Performance (8th Edition) by William Stallings [CH-17].
- **Problems:** 17.1 to 17.4, 17.12, 17.14 to 17.16

Thank You 😊