+

William Stallings Computer Organization and Architecture 10th Edition

+

Chapter 17

Parallel Processing

Multiple Processor Organization

- Single instruction, single data **(SISD)** stream
  - Single processor executes a single instruction stream to operate on data stored in a single memory
  - Uniprocessors fall into this category
- Single instruction, multiple data **(SIMD)** stream
  - A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis
  - Vector and array processors fall into this category
- Multiple instruction, single data **(MISD)** stream
  - A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence
  - Not commercially implemented
- Multiple instruction, multiple data **(MIMD)** stream
  - A set of processors simultaneously execute different instruction sequences on different data sets
  - SMPs, clusters and NUMA systems fit this category

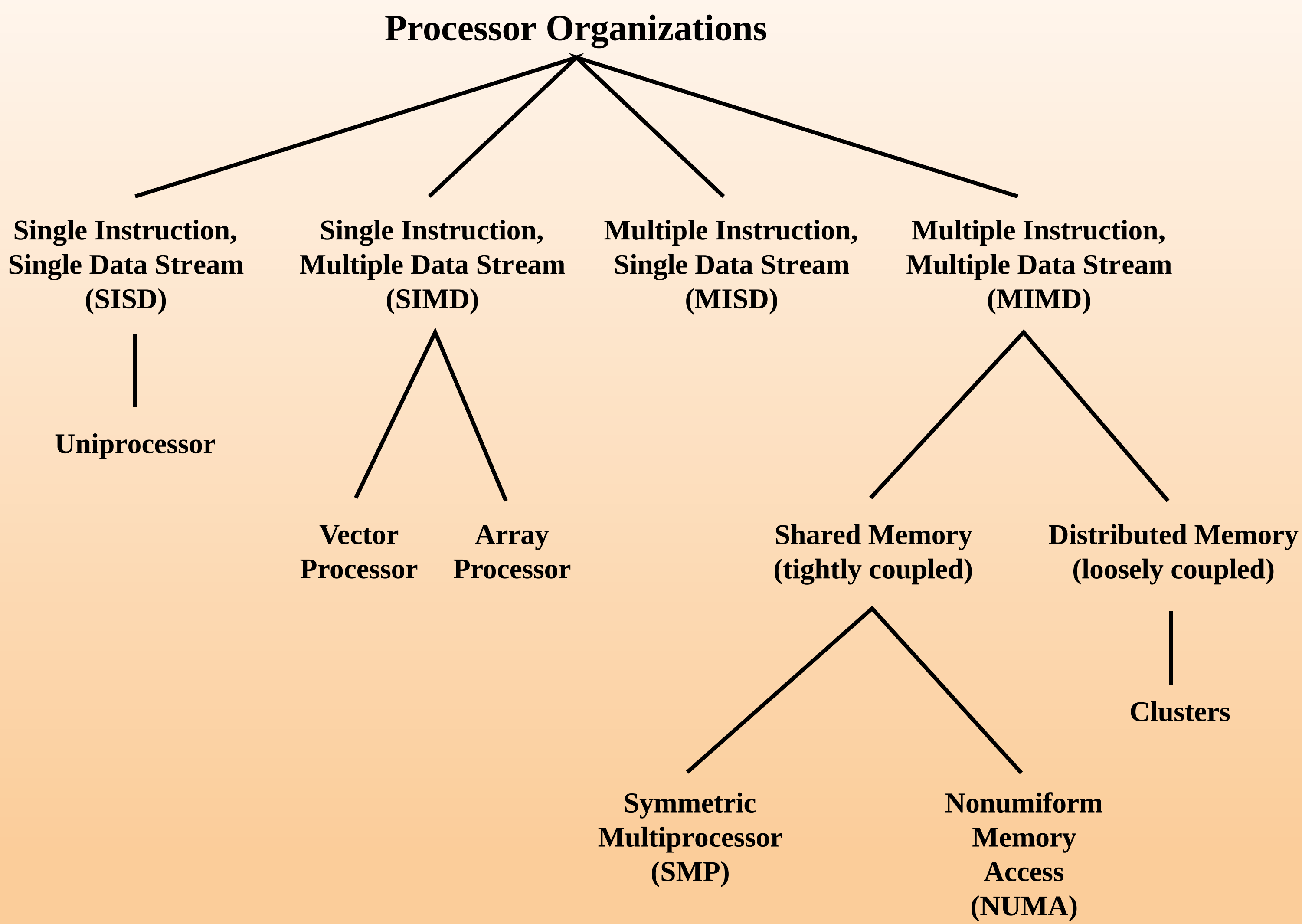


Figure 17.1 A Taxonomy of Parallel Processor Architectures

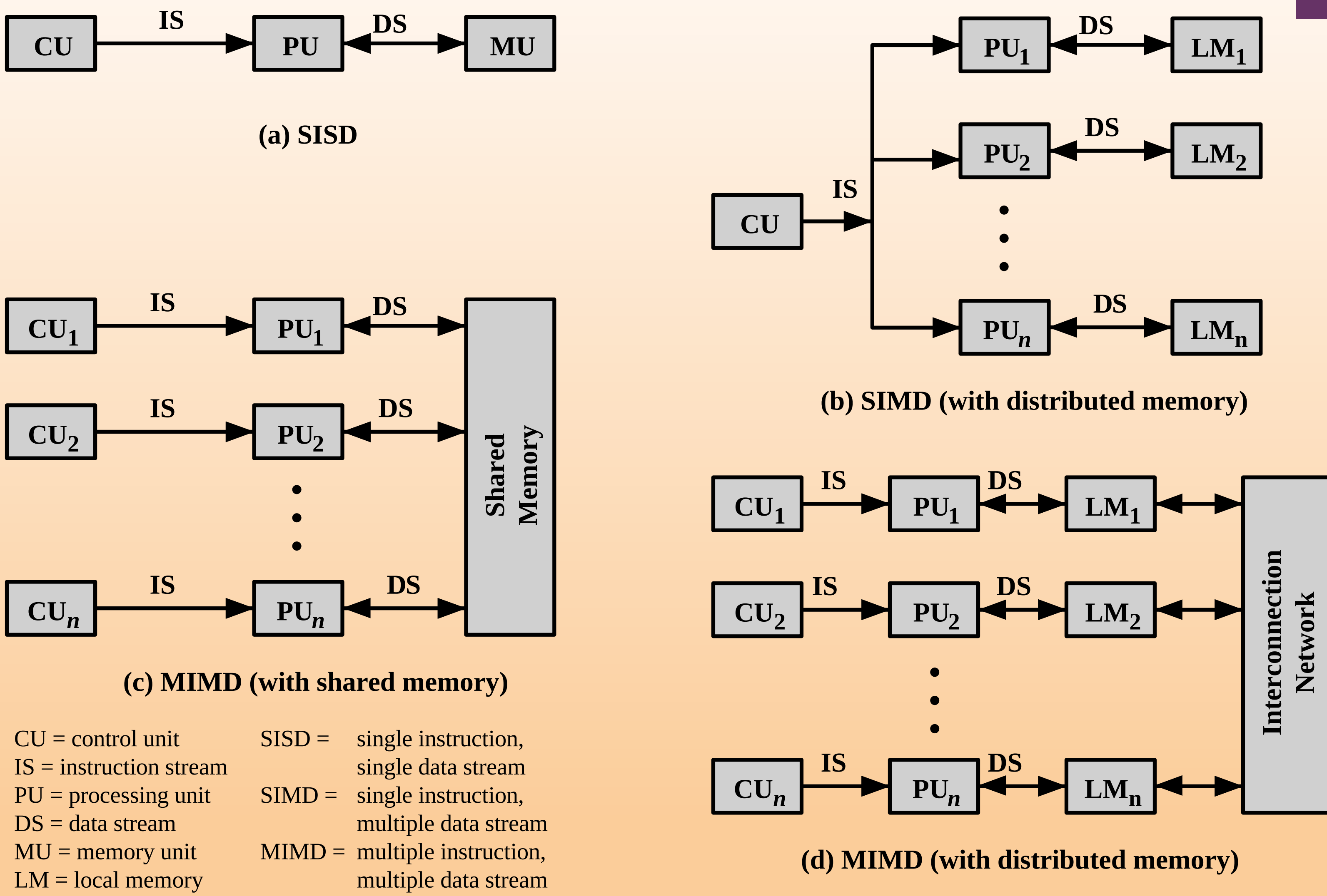


Figure 17.2 Alternative Computer Organizations

Symmetric Multiprocessor (SMP)

A stand alone computer with the following characteristics:

- Two or more similar processors of comparable capacity
- Processors share same memory and I/O facilities
  - 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 devices
  - 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
  - Provides interaction between processors and their programs at job, task, file and data element levels

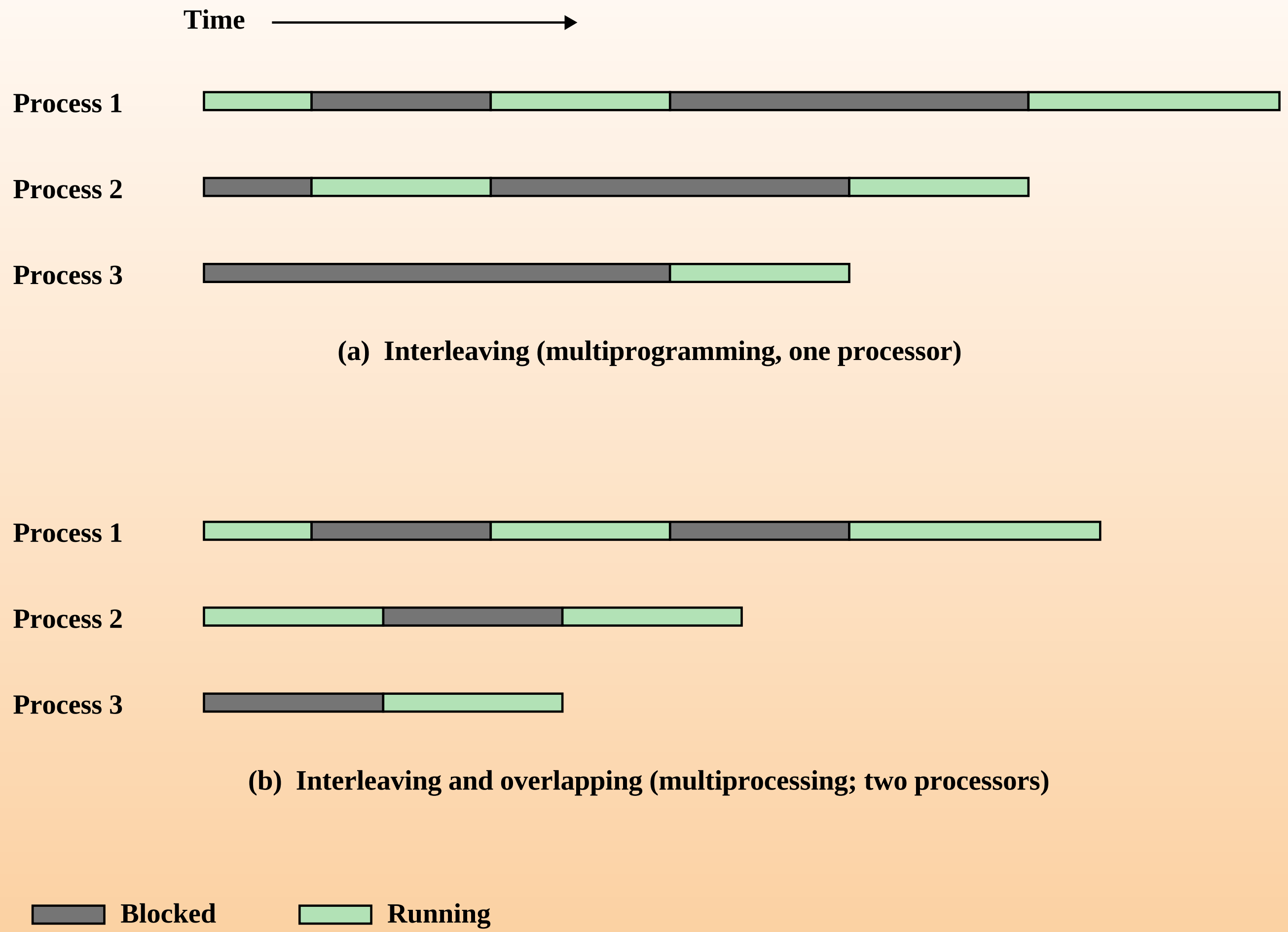


Figure 17.3 Multiprogramming and Multiprocessing

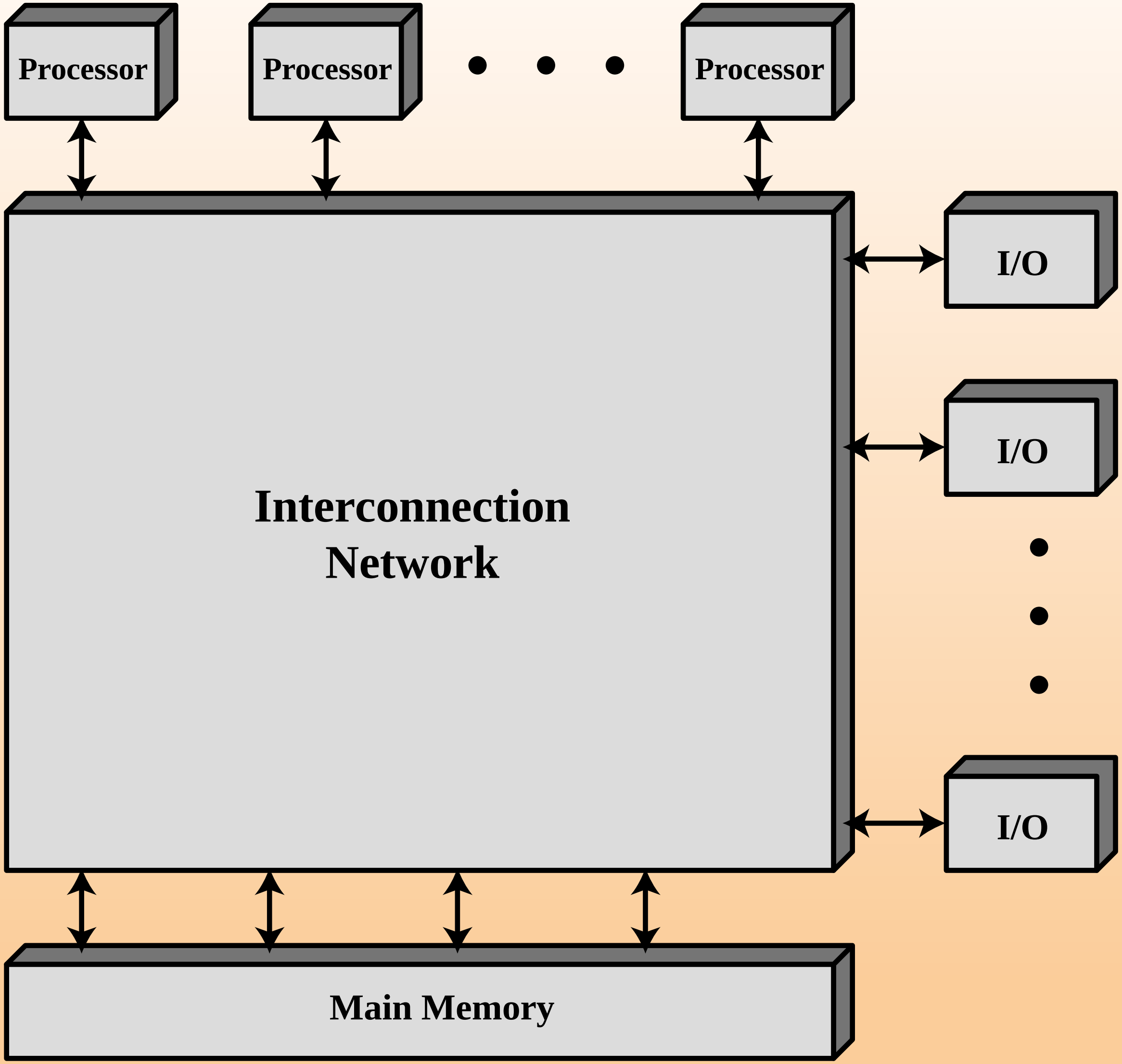


Figure 17.4 Generic Block Diagram of a Tightly Coupled Multiprocessor

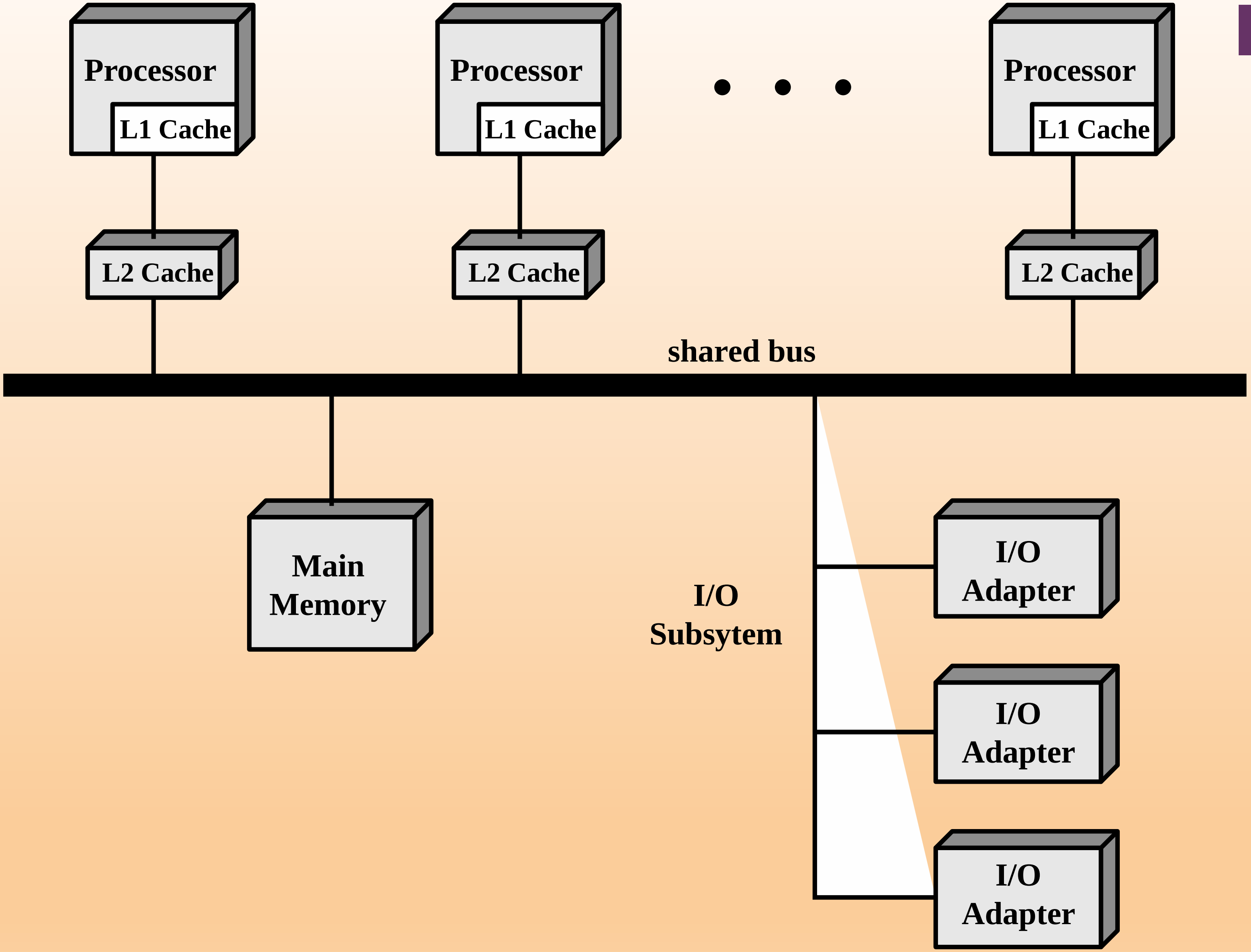
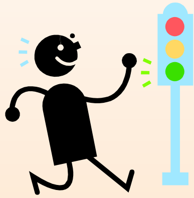


Figure 17.5 Symmetric Multiprocessor Organization

+

The bus organization has several attractive features:



- Simplicity
  - Simplest approach to multiprocessor organization
- Flexibility
  - Generally easy to expand the system by attaching more processors to the bus
- Reliability
  - The bus is essentially a passive medium and the failure of any attached device should not cause failure of the whole system

+

Disadvantages of the bus organization:

- Main drawback is performance
  - All memory references pass through the common bus
  - Performance is limited by bus cycle time
- Each processor should have cache memory
  - Reduces the number of bus accesses
- Leads to problems with *cache coherence*
  - If a word is altered in one cache it could conceivably invalidate a word in another cache
    - To prevent this the other processors must be alerted that an update has taken place
  - Typically addressed in hardware rather than the operating system

+

Multiprocessor Operating System Design Considerations

- **Simultaneous concurrent processes**
  - OS routines need to be reentrant to allow several processors to execute the same IS code simultaneously
  - OS tables and management structures must be managed properly to avoid deadlock or invalid operations
- **Scheduling**
  - Any processor may perform scheduling so conflicts must be avoided
  - Scheduler must assign ready processes to available processors
- **Synchronization**
  - With multiple active processes having potential access to shared address spaces or I/O resources, care must be taken to provide effective synchronization
  - Synchronization is a facility that enforces mutual exclusion and event ordering
- **Memory management**
  - In addition to dealing with all of the issues found on uniprocessor machines, the OS needs to exploit the available hardware parallelism to achieve the best performance
  - Paging mechanisms on different processors must be coordinated to enforce consistency when several processors share a page or segment and to decide on page replacement
- **Reliability and fault tolerance**
  - OS should provide graceful degradation in the face of processor failure
  - Scheduler and other portions of the operating system must recognize the loss of a processor and restructure accordingly

+

Cache Coherence

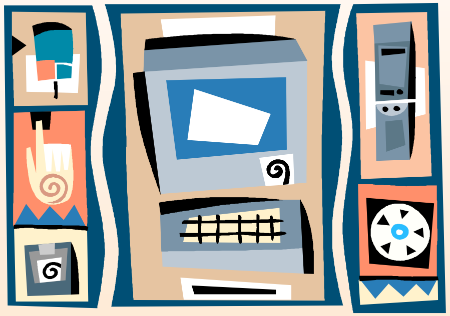
Software Solutions

- Attempt to avoid the need for additional hardware circuitry and logic by relying on the compiler and operating system to deal with the problem
- Attractive because the overhead of detecting potential problems is transferred from run time to compile time, and the design complexity is transferred from hardware to software
  - However, compile-time software approaches generally must make conservative decisions, leading to inefficient cache utilization

+

Cache Coherence

Hardware-Based Solutions



- Generally referred to as cache coherence protocols
- These solutions provide dynamic recognition at run time of potential inconsistency conditions
- Because the problem is only dealt with when it actually arises there is more effective use of caches, leading to improved performance over a software approach
- Approaches are transparent to the programmer and the compiler, reducing the software development burden
- Can be divided into two categories:
  - 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 the responsibility for maintaining cache coherence among all of the cache controllers in a multiprocessor
  - A cache must recognize when a line that it holds is shared with other caches
  - When updates are performed on a shared cache line, it must be announced to other caches by a broadcast mechanism
  - Each cache controller is able to "snoop" on the network to observe these broadcast notifications and react accordingly
- Suited to bus-based multiprocessor because the shared bus provides a simple means for broadcasting and snooping
  - Care must be taken that the increased bus traffic required for broadcasting and snooping does not cancel out the gains from the use of local caches
- Two basic approaches have been explored:
  - Write invalidate
  - Write update (or write broadcast)

Write Invalidate

- Multiple readers, but only one writer at a time
- When a write is required, all other caches of the line are invalidated
- Writing processor then has exclusive (cheap) access until line is required by another processor
- Most widely used in commercial multiprocessor systems such as the x86 architecture
- State of every line is marked as modified, exclusive, shared or invalid
  - For this reason the write-invalidate protocol is called *MESI*

+

Write Update

Can be multiple readers and writers

When a processor wishes to update a shared line the word to be updated is distributed to all others and caches containing that line can update it

Some systems use an adaptive mixture of both write-invalidate and write-update mechanisms

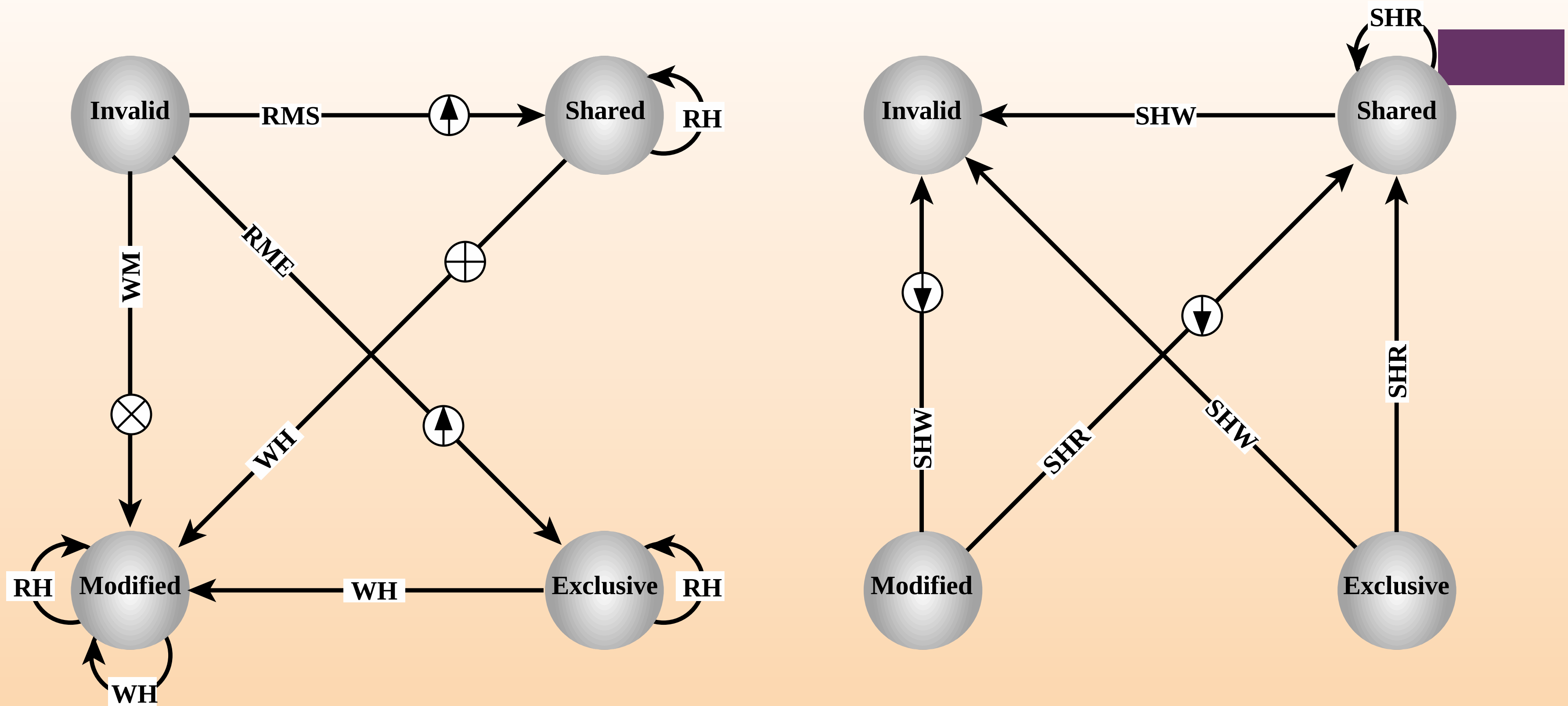
MESI Protocol

To provide cache consistency on an SMP the data cache supports a protocol known as MESI:

- Modified
  - The line in the cache has been modified 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

Table 17.1
MESI Cache Line States

	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



(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

Figure 17.6 MESI State Transition Diagram

Multithreading and Chip Multiprocessors

- Processor performance can be measured by the rate at which it executes instructions
- MIPS rate = f * IPC
  - f = processor clock frequency, in MHz
  - IPC = average instructions per cycle
- Increase performance by increasing clock frequency and increasing instructions that complete during cycle
- Multithreading
  - Allows for a high degree of instruction-level parallelism without increasing circuit complexity or power consumption
  - Instruction stream is divided into several smaller streams, known as threads, that can be executed in parallel

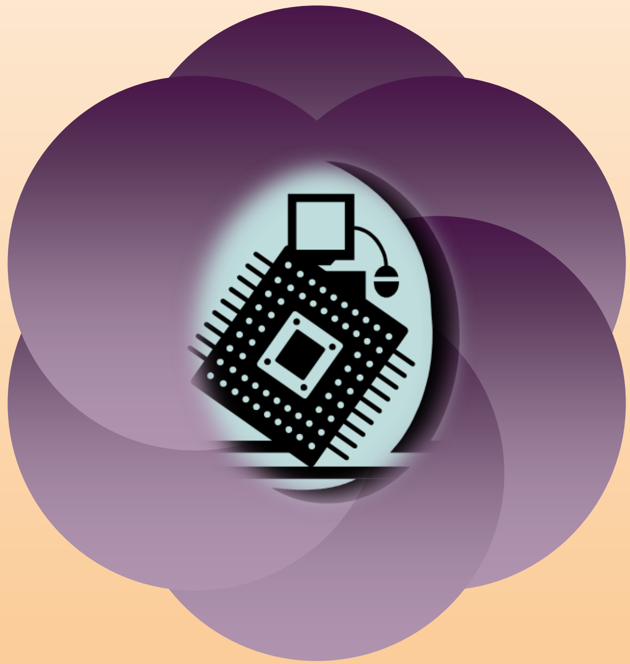
Definitions of Threads and Processes

Thread in multithreaded processors may or may not be the same as the concept of software threads in a multiprogrammed operating system

Thread switch

- The act of switching processor control between threads within the same process
- Typically less costly than process switch

Thread is concerned with scheduling and execution, whereas a process is concerned with both scheduling/execution and resource and resource ownership



Thread:

- Dispatchable unit of work within a process
- Includes processor context (which includes the program counter and stack pointer) and data area for stack
- Executes sequentially and is interruptible so that the processor can turn to another thread

Process:

- An instance of program running on computer
- Two key characteristics:
  - Resource ownership
  - Scheduling/execution

Process switch

- Operation that switches the processor from one process to another by saving all the process control data, registers, and other information for the first and replacing them with the process information for the second

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
- Simultaneous (SMT)
  - Instructions are simultaneously issued from multiple threads to execution units of superscalar processor
- Blocked
  - Coarse-grained
  - Thread executed until event causes delay
  - Effective on in-order processor
  - Avoids pipeline stall
- Chip multiprocessing
  - Processor is replicated on a single chip
  - Each processor handles separate threads
  - Advantage is that the available logic area on a chip is used effectively

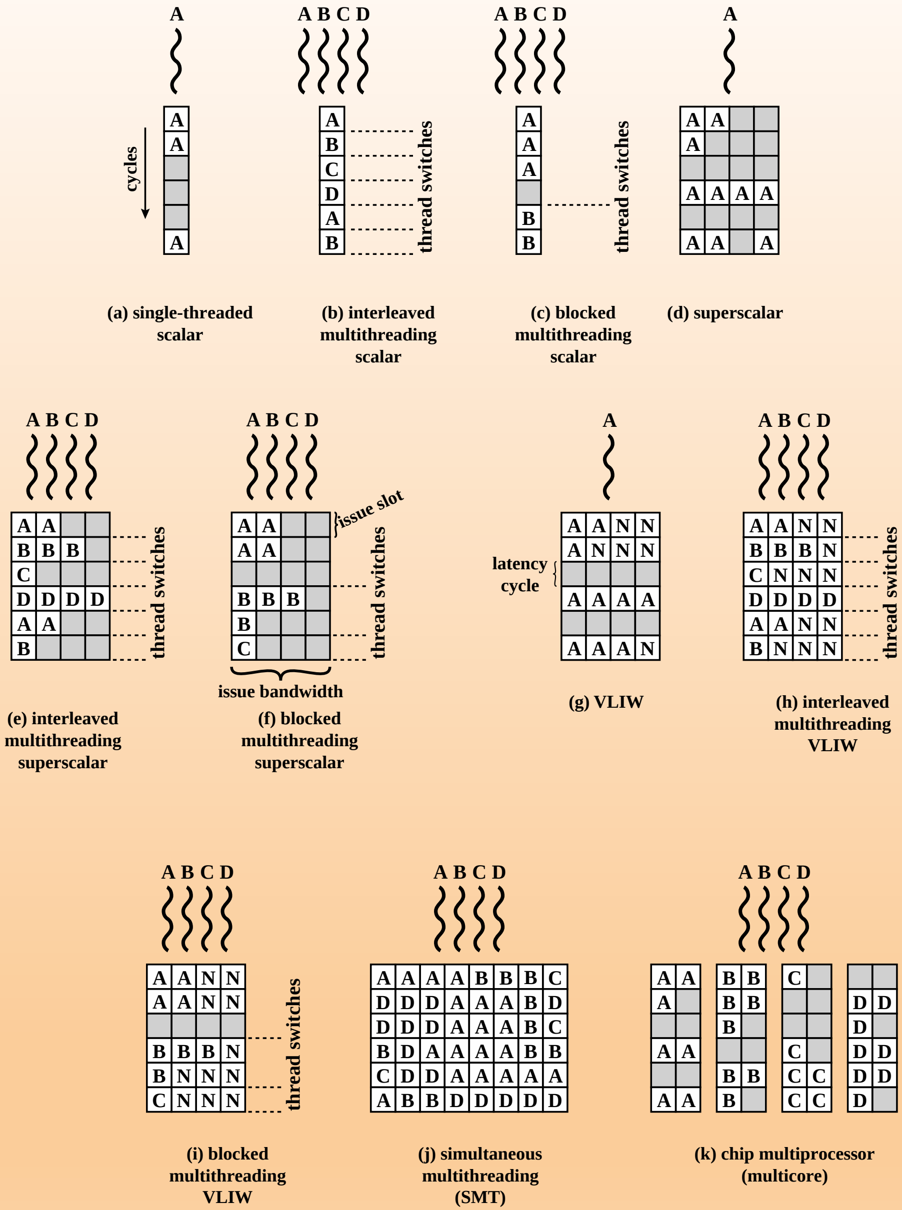
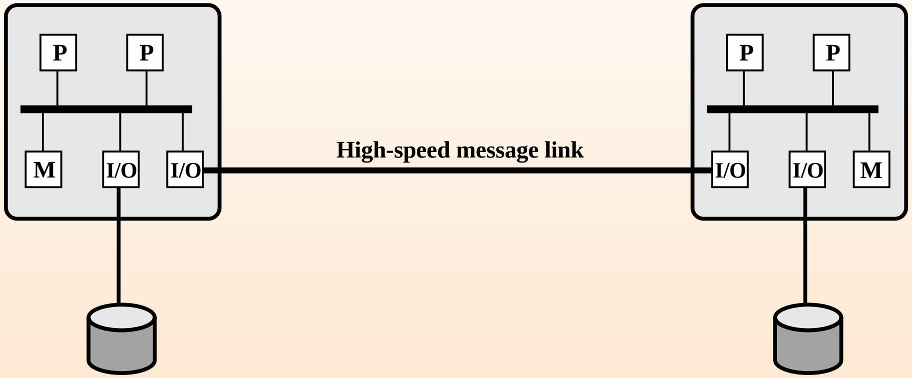


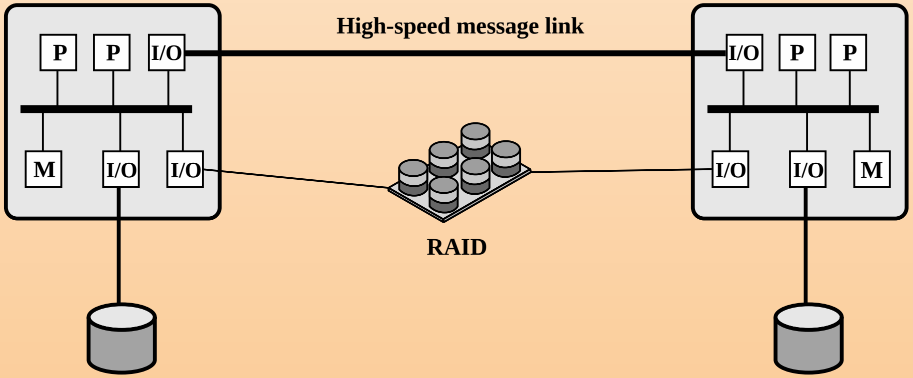
Figure 17.7 Approaches to Executing Multiple Threads

Clusters

- Alternative to SMP as an approach to providing high performance and high availability
- Particularly attractive for server applications
- Defined as:
  - A group of interconnected whole computers working together as a unified computing resource that can create the illusion of being one machine
  - (The term *whole computer* means a system that can run on its own, apart from the cluster)
- Each computer in a cluster is called a node
- Benefits:
  - Absolute scalability
  - Incremental scalability
  - High availability
  - Superior price/performance



(a) Standby server with no shared disk



(b) Shared disk

Figure 17.8 Cluster Configurations

Table 17.2
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 System Design Issues

- How failures are managed depends on the clustering method used
- Two approaches:
  - Highly available clusters
  - Fault tolerant clusters
- Failover
  - The function of switching applications and data resources over from a failed system to an alternative system in the cluster
- Failback
  - Restoration of applications and data resources to the original system once it has been fixed
- Load balancing
  - Incremental scalability
  - Automatically include new computers in scheduling
  - Middleware needs to recognize that processes may switch between machines

Parallelizing Computation

Effective use of a cluster requires executing software from a single application in parallel

Three approaches are:

Parallelizing complier

- Determines at compile time which parts of an application can be executed in parallel
- These are then split off to be assigned to different computers in the cluster

Parallelized application

- Application written from the outset to run on a cluster and uses message passing to move data between cluster nodes

Parametric computing

- Can be used if the essence of the application is an algorithm or program that must be executed a large number of times, each time with a different set of starting conditions or parameters

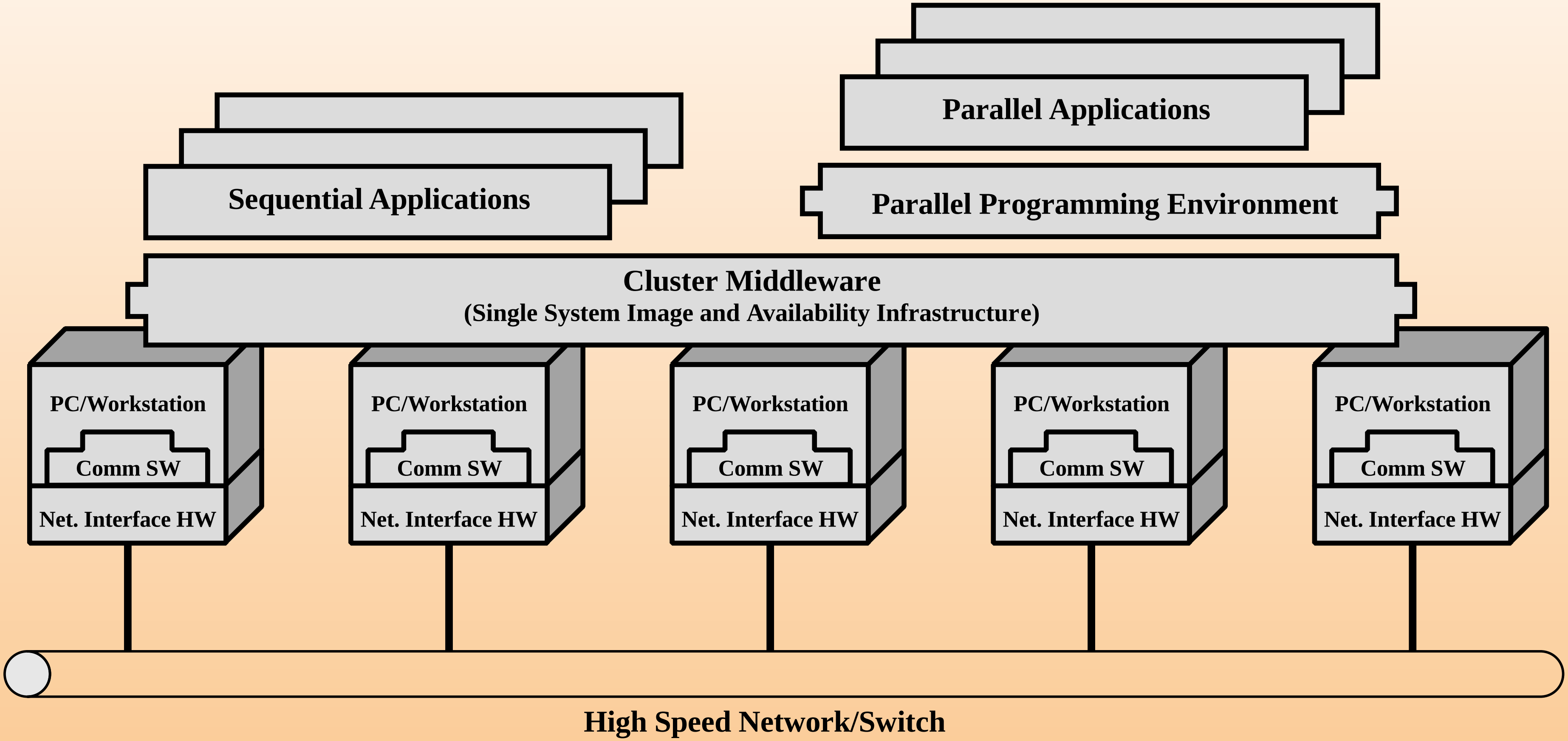


Figure 17.9 Cluster Computer Architecture

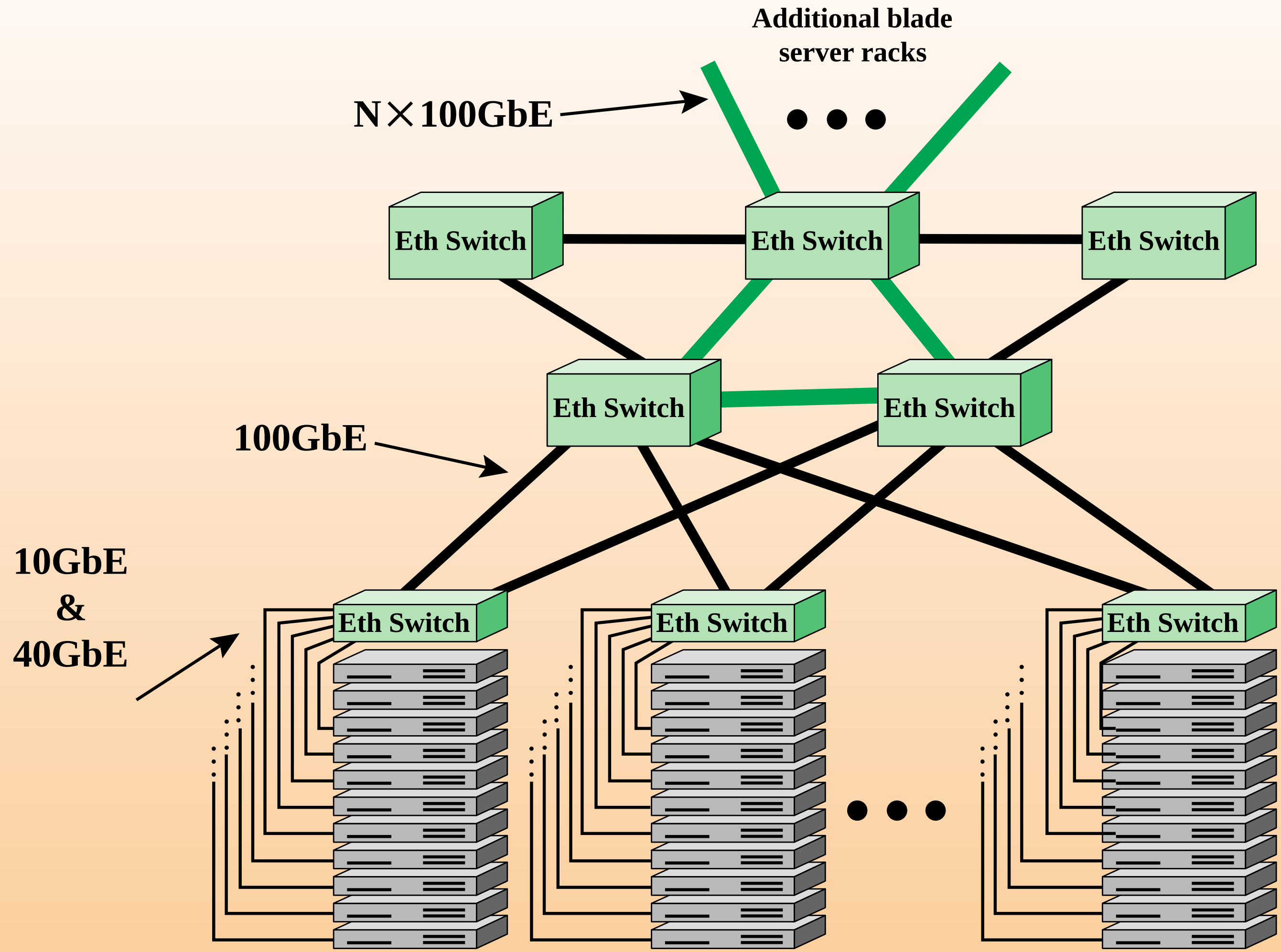


Figure 17.10 Example 100-Gbps Ethernet Configuration for Massive Blade Server Cloud Site

+

Clusters Compared to SMP

- Both provide a configuration with multiple processors to support high demand applications
- Both solutions are available commercially

SMP	Clustering
Easier to manage and configure	Far superior in terms of incremental and absolute scalability
Much closer to the original single processor model for which nearly all applications are written	Superior in terms of availability
Less physical space and lower power consumption	All components of the system can readily be made highly redundant
Well established and stable	

+ Nonuniform Memory Access (NUMA)

- Alternative to SMP and clustering
- Uniform memory access (UMA)
  - All processors have access to all parts of main memory using loads and stores
  - Access time to all regions of memory is the same
  - Access time to memory for different processors is the same
- Nonuniform memory access (NUMA)
  - All processors have access to all parts of main memory using loads and stores
  - Access time of processor differs depending on which region of main memory is being accessed
  - Different processors access different regions of memory at different speeds
- Cache-coherent NUMA (CC-NUMA)
  - A NUMA system in which cache coherence is maintained among the caches of the various processors

Motivation

SMP has practical limit to number of processors that can be used

- Bus traffic limits to between 16 and 64 processors

In clusters each node has its own private main memory

- Applications do not see a large global memory
- Coherency is maintained by software rather than hardware

NUMA retains SMP flavor while giving large scale multiprocessing

Objective with NUMA is to maintain a transparent system wide memory while permitting multiple multiprocessor nodes, each with its own bus or internal interconnect system

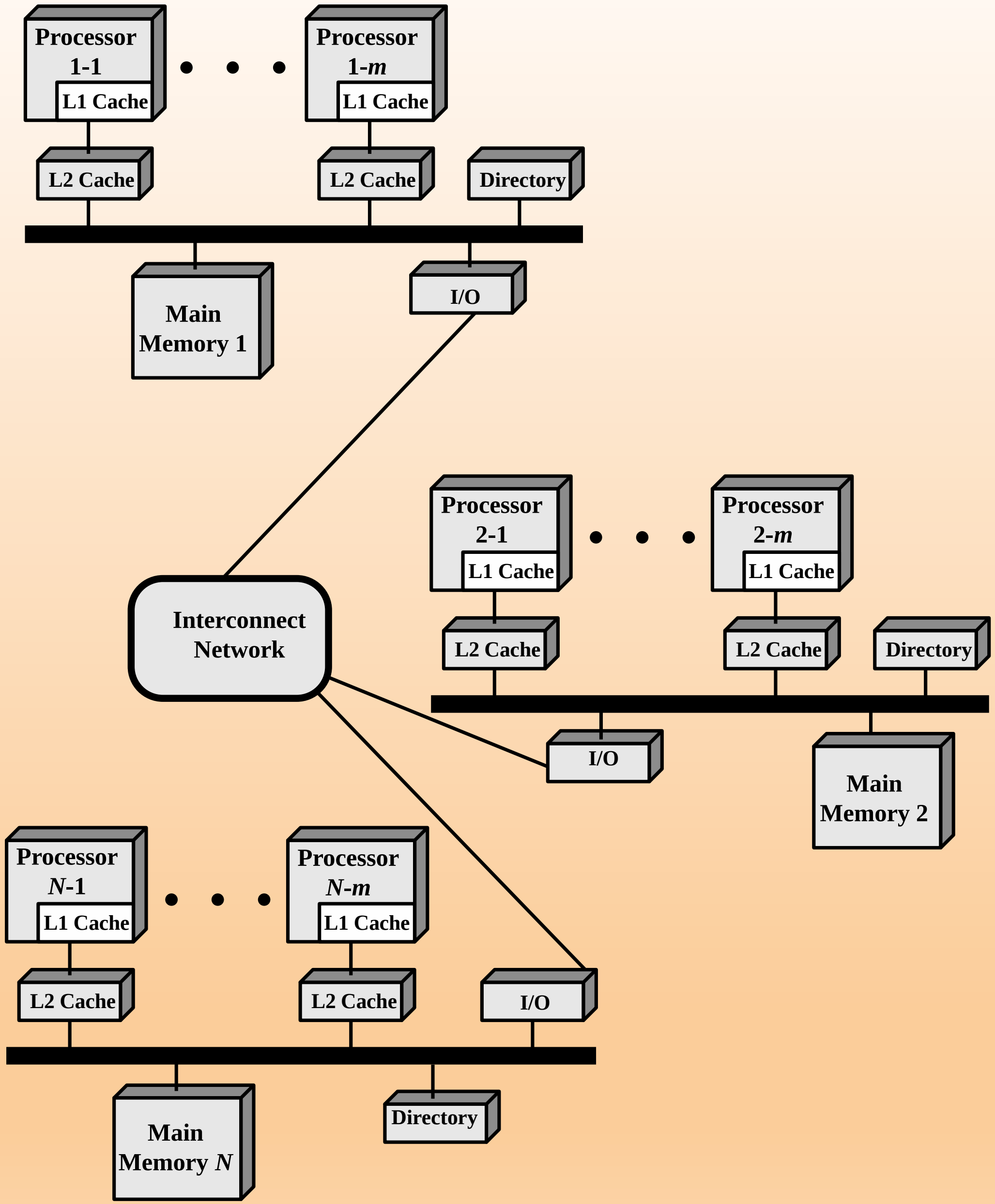


Figure 17.11 CC-NUMA Organization

+

NUMA Pros and Cons

- Main advantage of a CC-NUMA system is that it can deliver effective performance at higher levels of parallelism than SMP without requiring major software changes
- Bus traffic on any individual node is limited to a demand that the bus can handle
- If many of the memory accesses are to remote nodes, performance begins to break down

- Does not transparently look like an SMP
- Software changes will be required to move an operating system and applications from an SMP to a CC-NUMA system
- Concern with availability

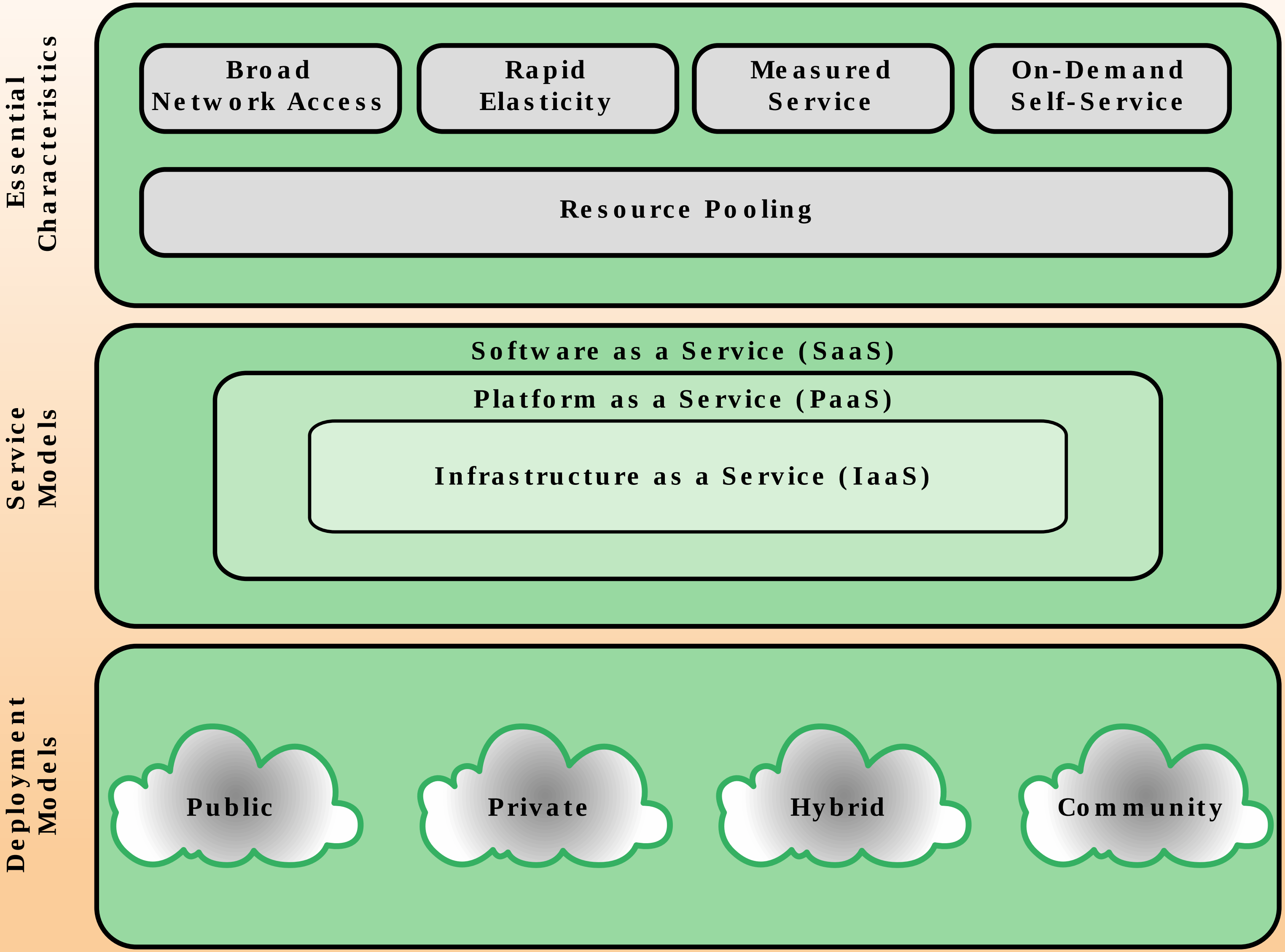


Figure 17.12 Cloud Computing Elements

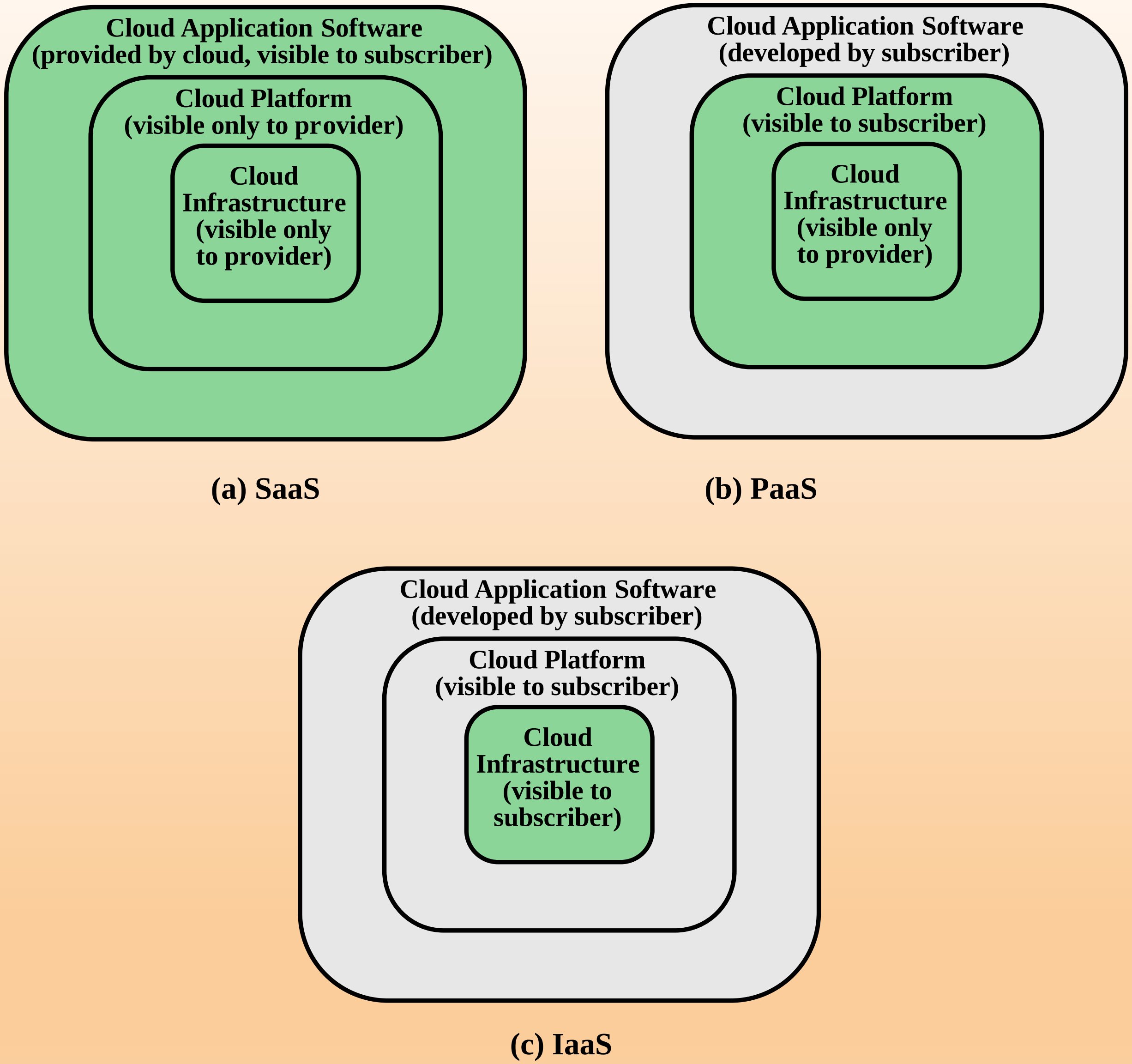


Figure 17.13 Cloud Service Models

Deployment Models

- Public cloud
  - The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services
  - Major advantage is cost
- Private cloud
  - A cloud infrastructure implemented within the internal IT environment of the organization
  - A key motivation for opting for a private cloud is security
- Community cloud
  - Like a private cloud it is not open to any subscriber
  - Like a public cloud the resources are shared among a number of independent orgaizations
- Hybrid cloud
  - The cloud infrastructure is a composition of two or more clouds that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability
  - Sensitive information can be placed in a private area of the cloud and less sensitive data can take advantage of the cost benefits of the public cloud

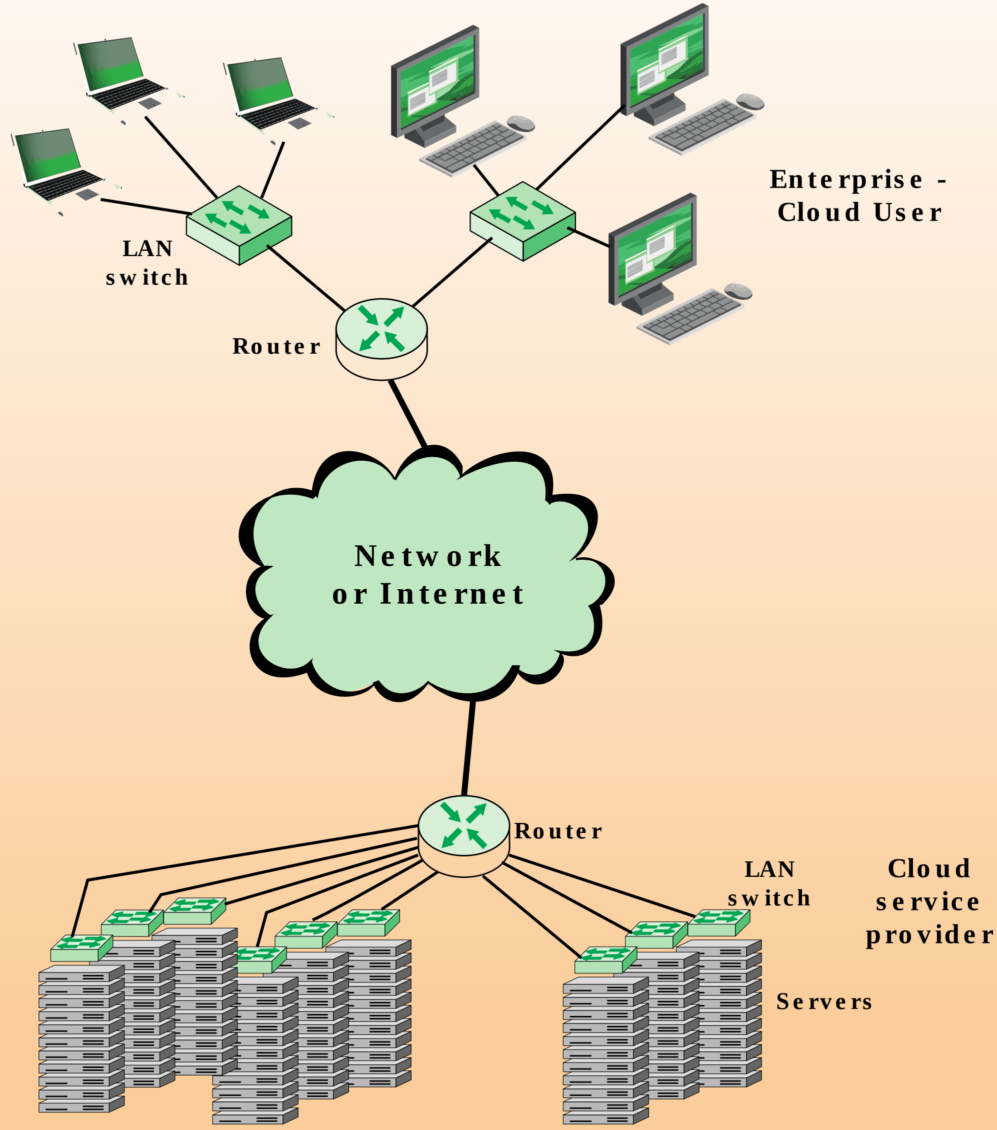


Figure 17.14 Cloud Computing Context

Cloud Computing Reference Architecture

- NIST SP 500-292 establishes a reference architecture, described as:

"The NIST cloud computing reference architecture focuses on the requirements of "what" cloud services provide, not a "how to" design solution and implementation. The reference architecture is intended to facilitate the understanding of the operational intricacies in cloud computing. It does not represent the system architecture of a specific cloud computing system; instead it is a tool for describing, discussing, and developing a system-specific architecture using a common framework of reference."

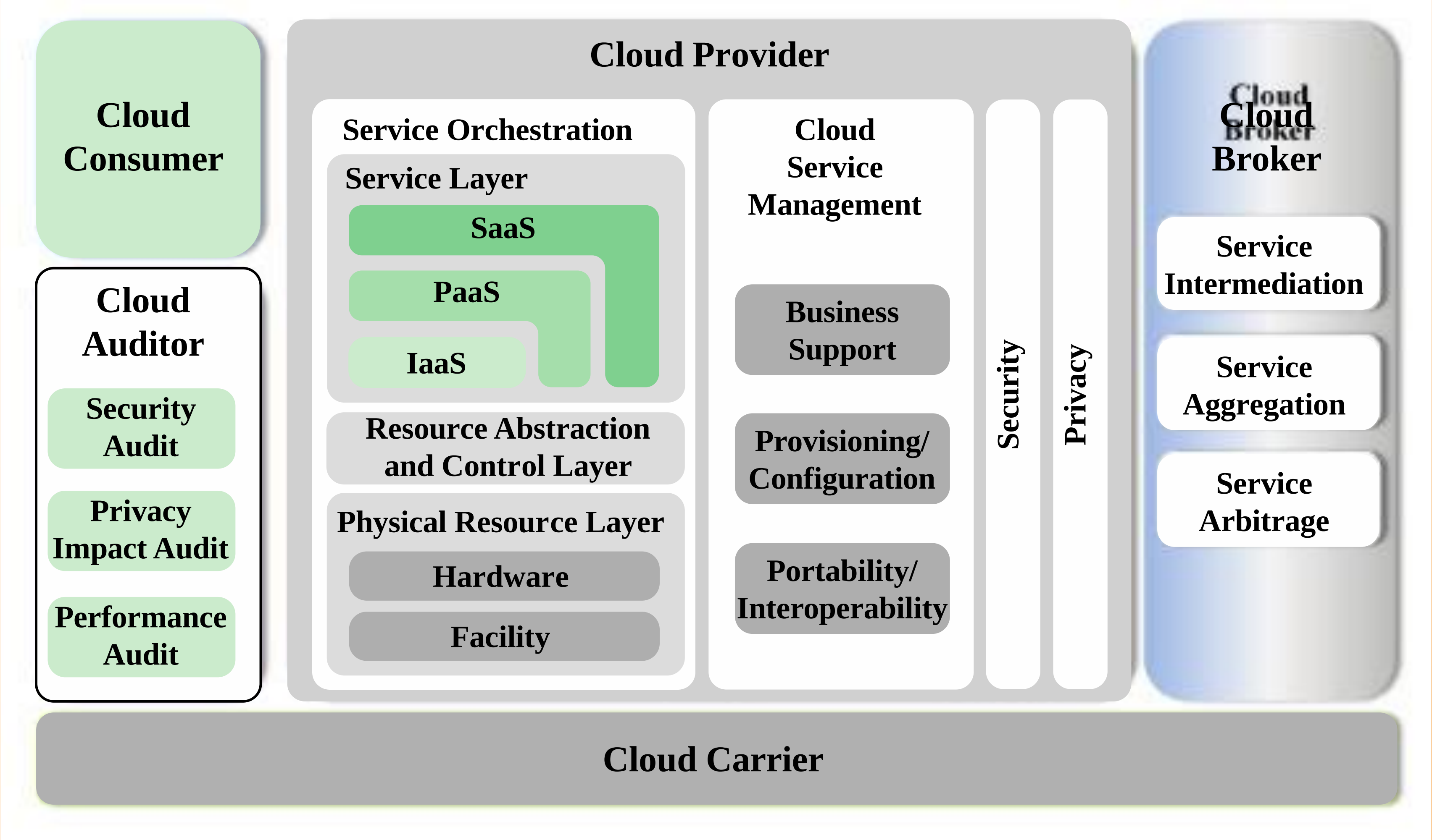


Figure 17.15 NIST Cloud Computing Reference Architecture

+ # Summary

Chapter 17

Parallel Processing

- Multiple processor organizations
  - Types of parallel processor systems
  - Parallel organizations
- Symmetric multiprocessors
  - Organization
  - Multiprocessor operating system design considerations
- Cache coherence and the MESI protocol
  - Software solutions
  - Hardware solutions
  - The MESI protocol
- Multithreading and chip multiprocessors
  - Implicit and explicit multithreading
  - Approaches to explicit multithreading
- Clusters
  - Cluster configurations
  - Operating system design issues
  - Cluster computer architecture
  - Blade servers
  - Clusters compared to SMP
- Nonuniform memory access
  - Motivation
  - Organization
  - NUMA Pros and cons
- Cloud computing
  - Cloud computing elements
  - Cloud computing reference architecture