CS 444 Operating Systems

Chapter 8 Multiple Processor Systems

J. Holly DeBlois

November 11, 2024

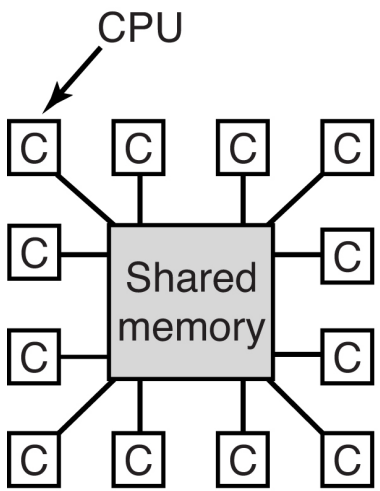
Physical Limits on Single Core

- Speed of light, electricity
- Heat dissipation
- From 1 MHz to 1 GHz is easy
- From 1 GHz to 1 THz is infeasible

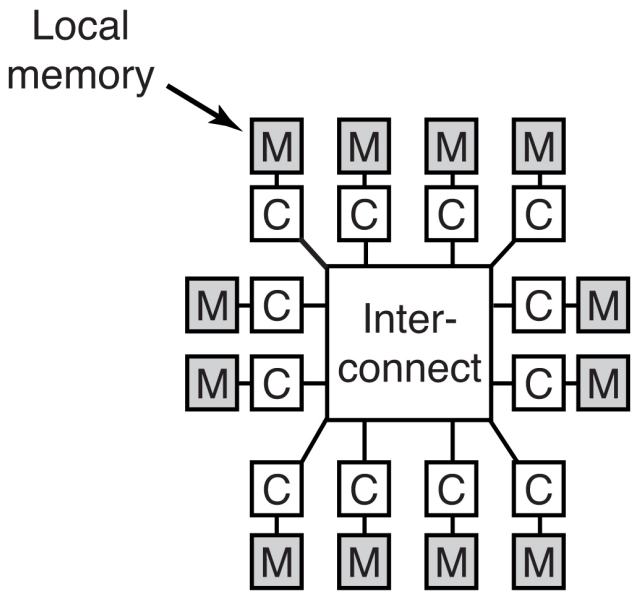
Three Scales of Communication Cost

- Shared-memory system
  - LOAD/STORE: 1-10 nano sec
- Message-passing multicomputer
  - High-speed interconnect: 10-50 micro sec
- Distributed system
  - Connected by a wide area network
- They differ by 3 orders of magnitude in delays

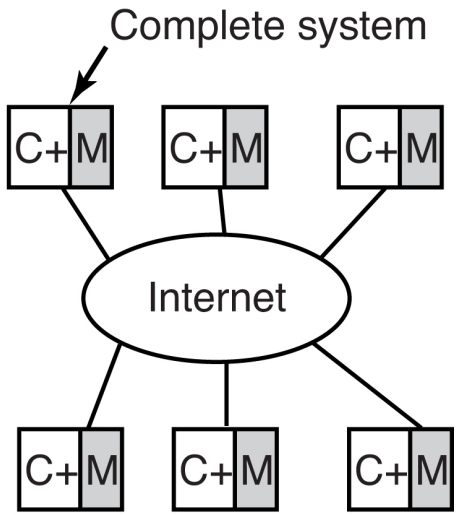
Three Scales of Multicomputers



(a)



(b)



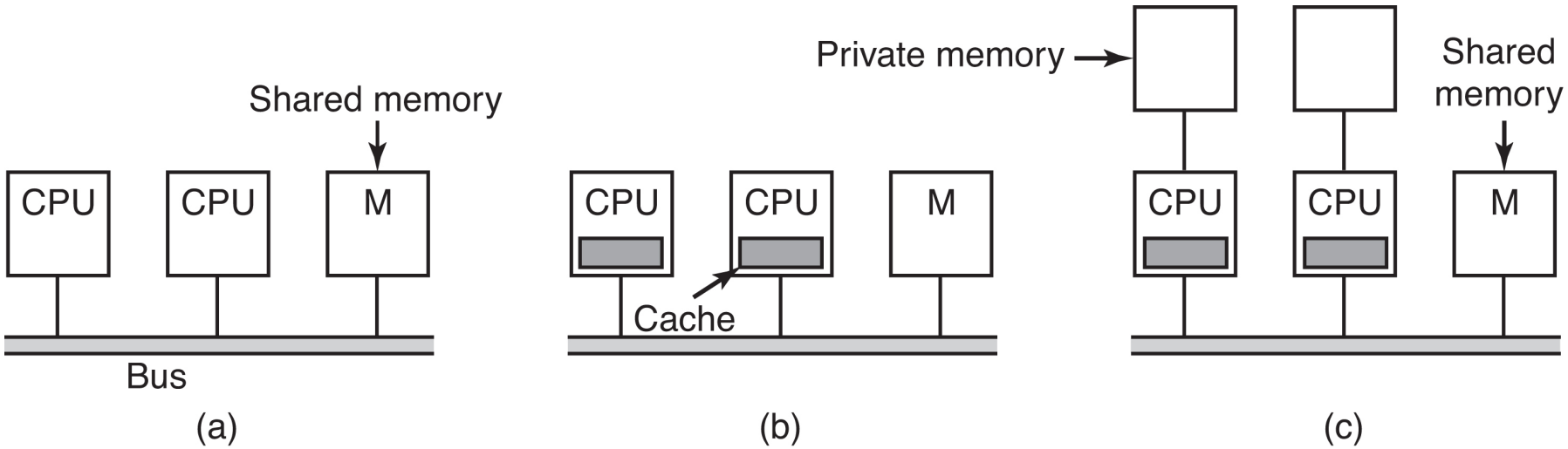
(c)

- Shared-memory computer
- Message-passing cluster
- Distributed system

Multiprocessor

- Shared-memory multiprocessor
- Shared-memory is the basis of inter-processor communication
- Uniform memory access, UMA
- Nonuniform memory access, NUMA

Shared-memory Architecture

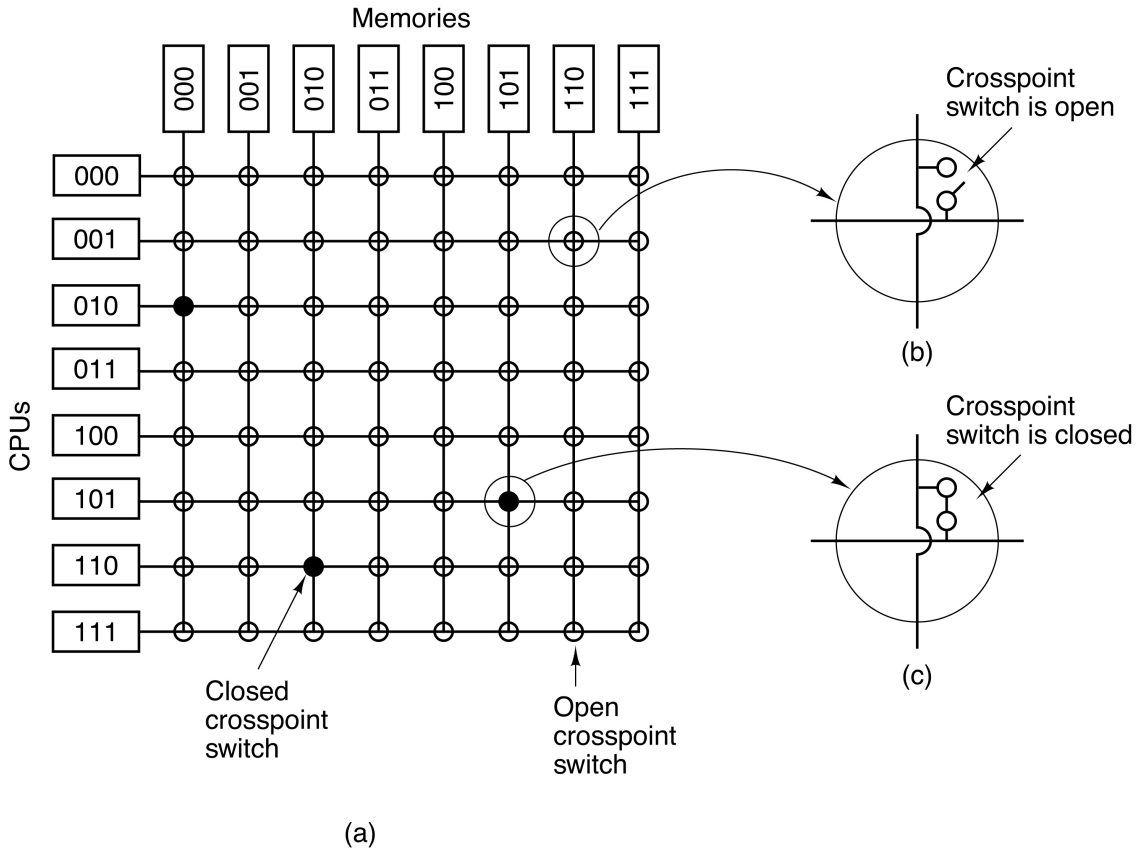


- Without caching
- Contention for bus
- Possible for 16-32 CPUs

- With caching
- Need to maintain cache coherence

- With caching and private memories

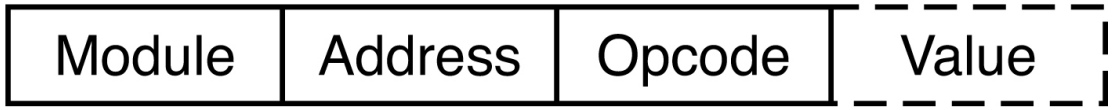
Uniform Memory Access Using Crossbar Switches



- Nonblocking
- n processors and n memory modules need n^2 crosspoints
- Does not scale up — scalability issues

Switching Circuits

(a)



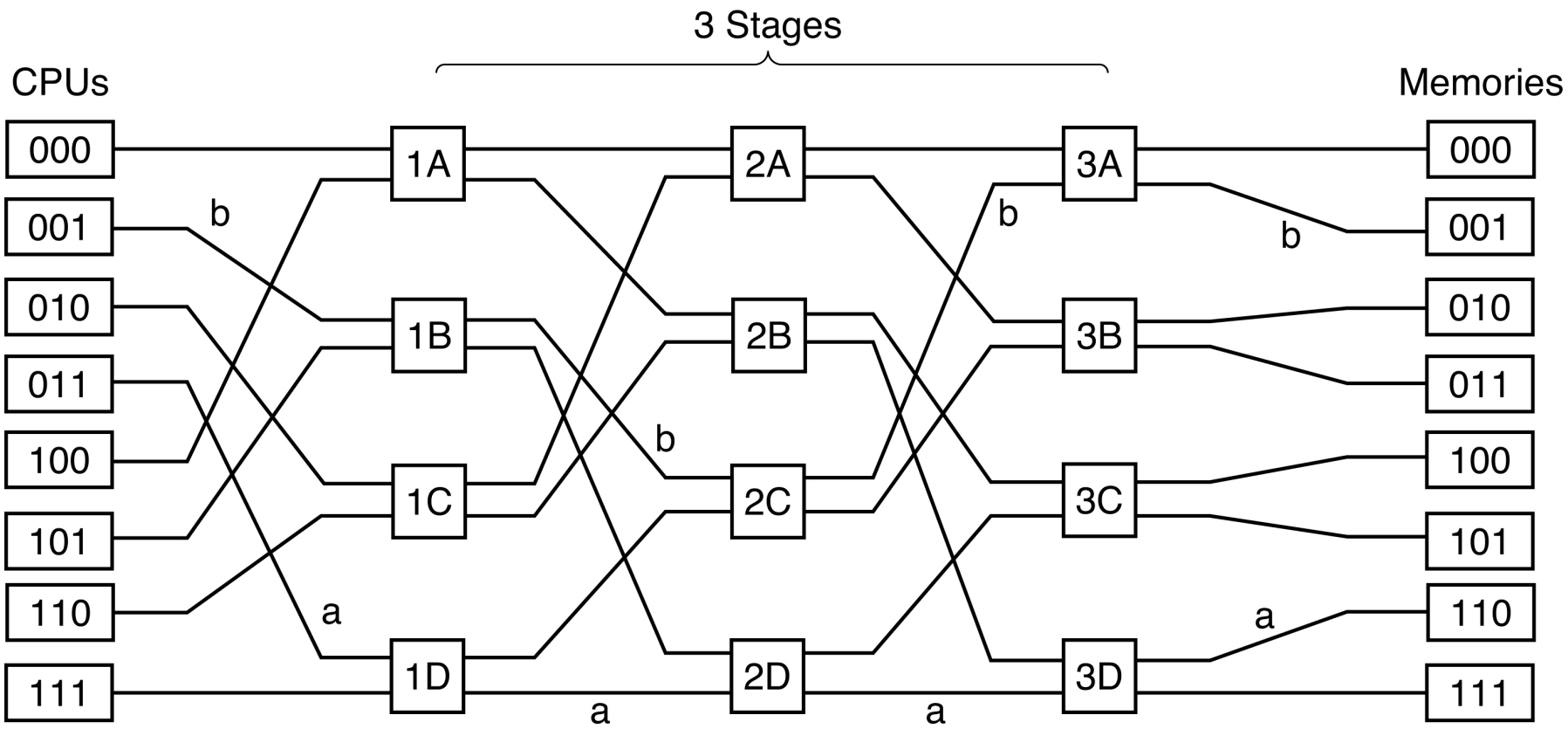
(b)

- A 2×2 switch with two input lines, A and B, and two output lines, X and Y
- A message format

UMA Multistage Switching

- Multistage switching network
- Omega network
  - n CPU
  - $\log_2 n$ stages
  - $n/2$ switches per stage
- Perfect shuffle from stage to stage
- It is a blocking network
- Uniform memory access

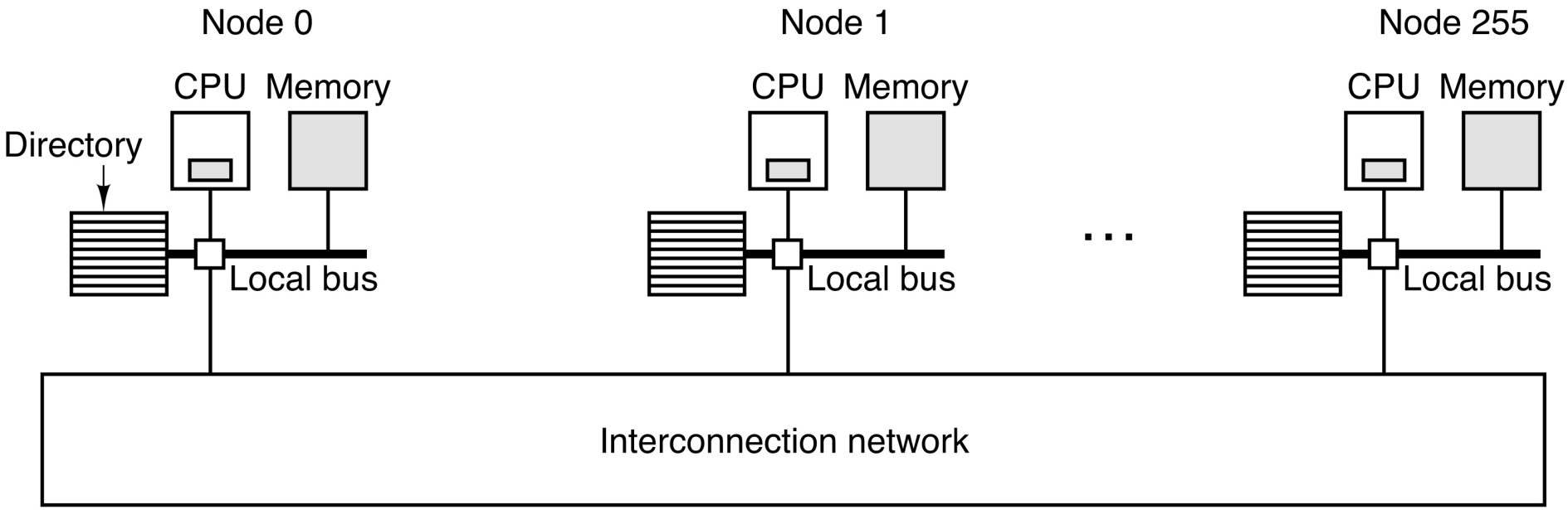
Omega Switching Network



Nonuniform Memory Access

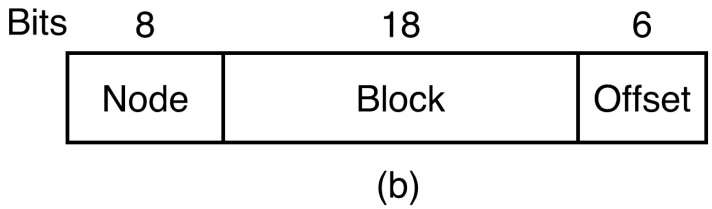
- NUMA
- For more than 100 CPUs
- Characteristics of NUMA machines which distinguish them from other multiprocessors
- There is a single address space visible to all CPUs
- Access to remote memory is via LOAD and STORE instructions
- Access to remote memory is slower than access to local memory

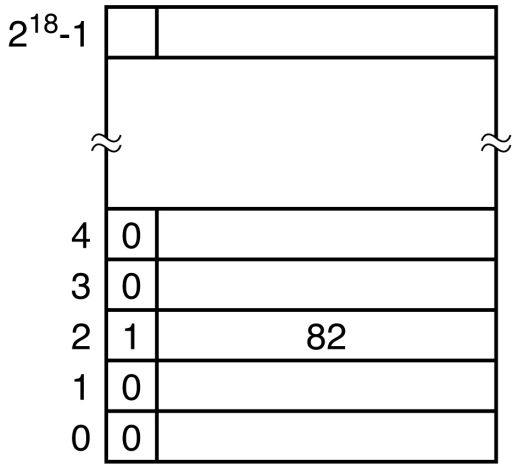
A 256-node NUMA Multiprocessor



- Cache-coherent, CC-NUMA

Cache Line Directory





- Memory address format
- 256 nodes
- 16 MB per node
- 4 GB in total
- 64 B per cache line
- 2^{26} cache lines

- Cache line directory
- 2^{18} rows
- 9 bits per row
- 1 bit: in use or not
- 8 bits: which node is using the cache line

Multicore Chips

- Dual-core, quad-core, octa-core, etc
- System on a chip, SoC
- Heterogeneous multicore
  - CPU
  - GPU
  - Video/audio decoders
  - Crypto-processors
  - Network interface

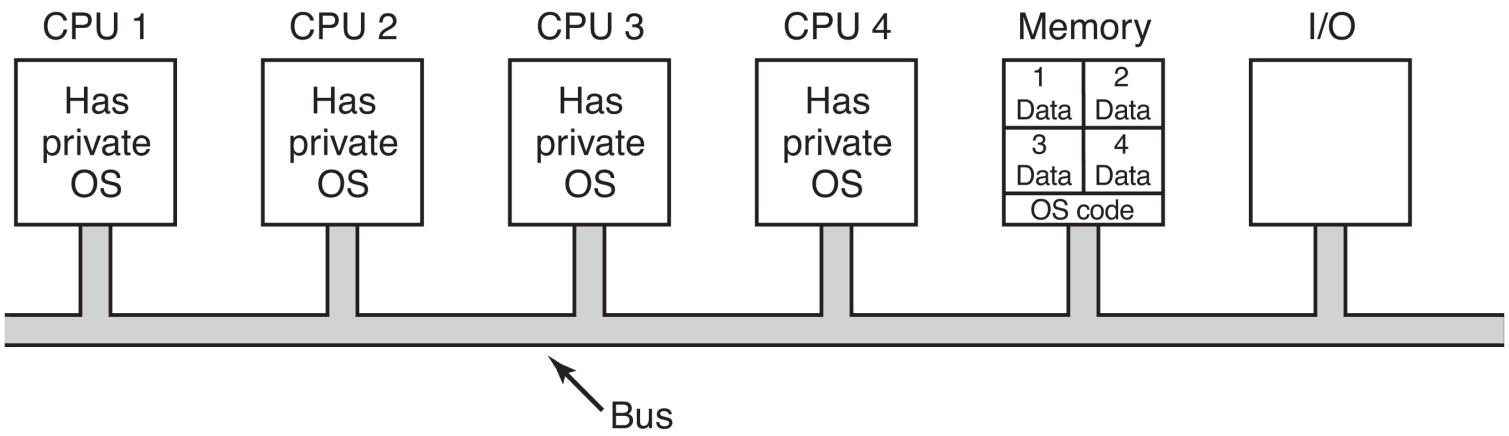
Manycore Chips

- How many is many? 50-100
- Cache coherence may not scale to many hundreds of cores
- This is called coherence wall
  - Slide 13 assumes only one node can hold a cache line — too restrictive if there are more than 100 cores
  - If a cache can be held by multiple cores, a 1024-core chip needs a directory of 1024 bits per cache line
  - 128 bytes of metadata per cache line
  - Size of cache line?
- GPU
  - GPU programming is difficult

Three Types of Multiprocessor Operating System

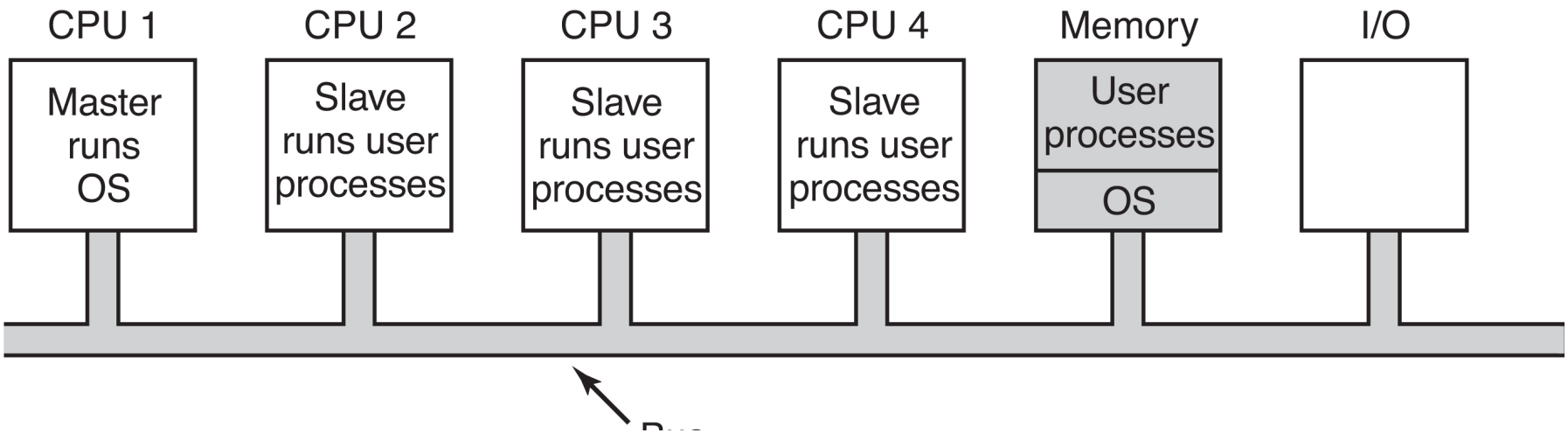
- Each CPU has its own OS
- Master-slave model
- Symmetric multiprocessor, SMP

Each CPU Has Its Own Operating System



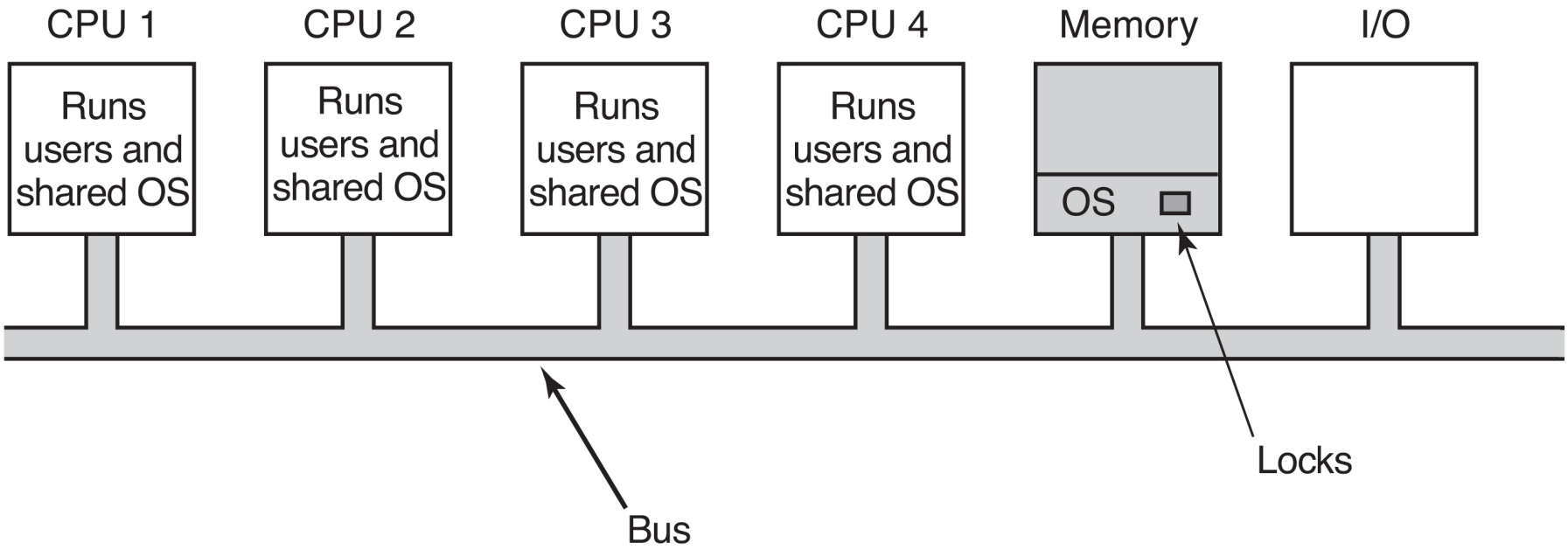
- Partition multiprocessor memory among 4 CPUs
- They share a single copy of the OS code
- The boxes marked Data are the OSs private data for each CPU
- System calls are handled by its own CPU
- No sharing of processes — unbalanced load
- No sharing of physical pages
- Need to eliminate disk buffer caches

The Master-Slave Model



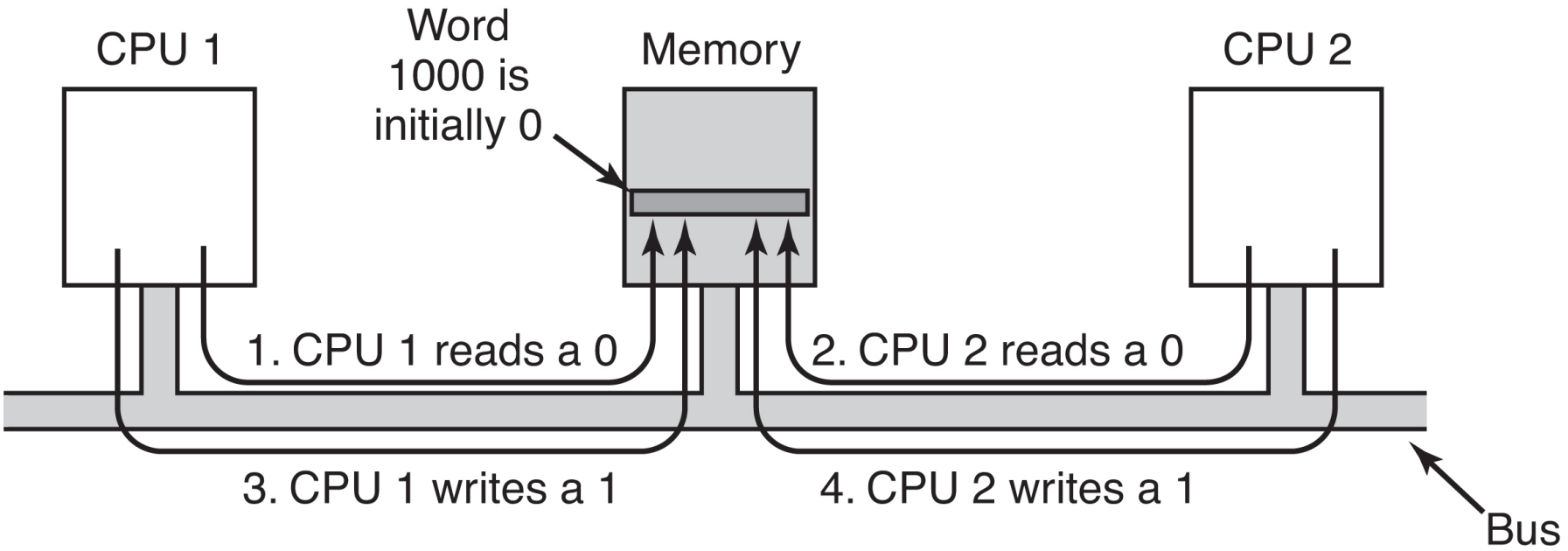
- Master node becomes the bottleneck

The SMP Model



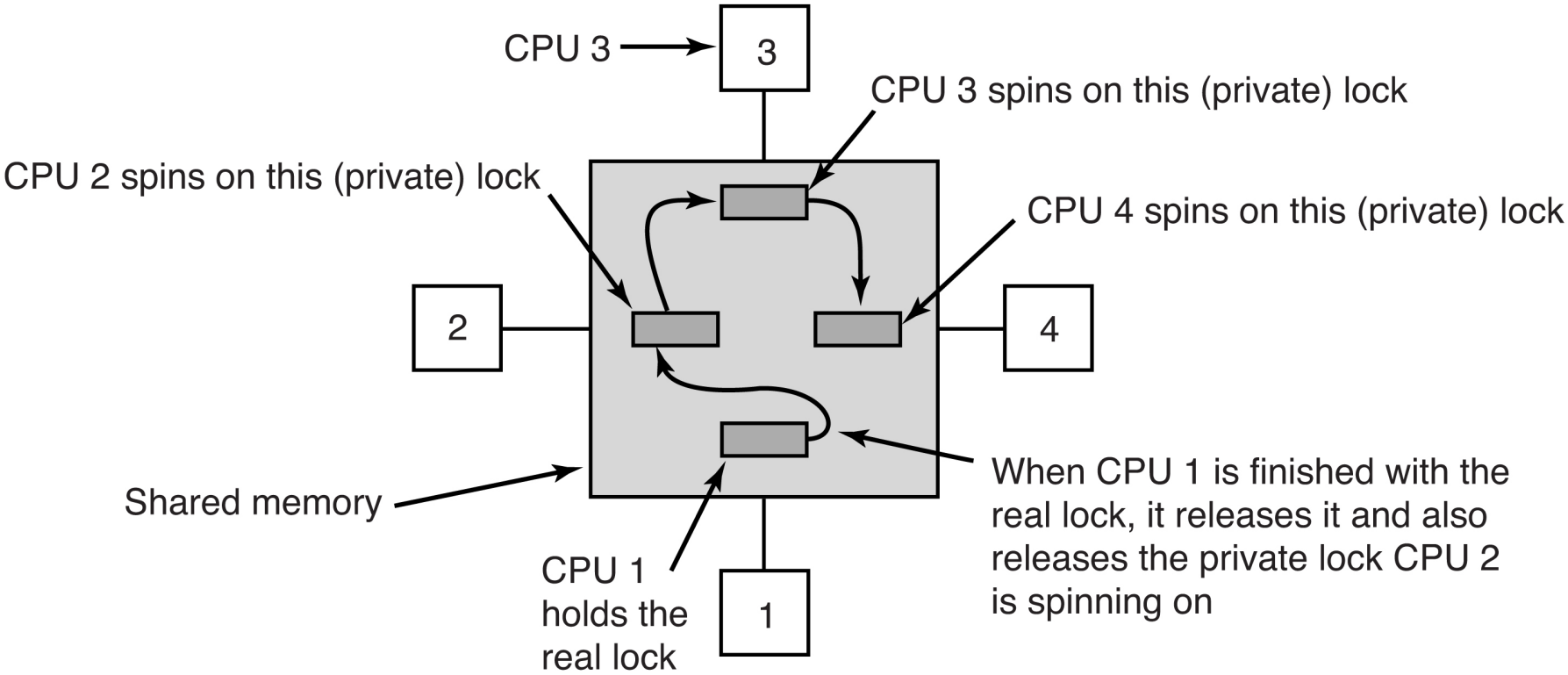
- A CPU acquires the kernel lock before taking up the kernel role
- Instead of one "big kernel lock", split the OS kernel into multiple independent critical regions — lock mutexes in a specific order
- Balance processes and memory dynamically

Synchronization



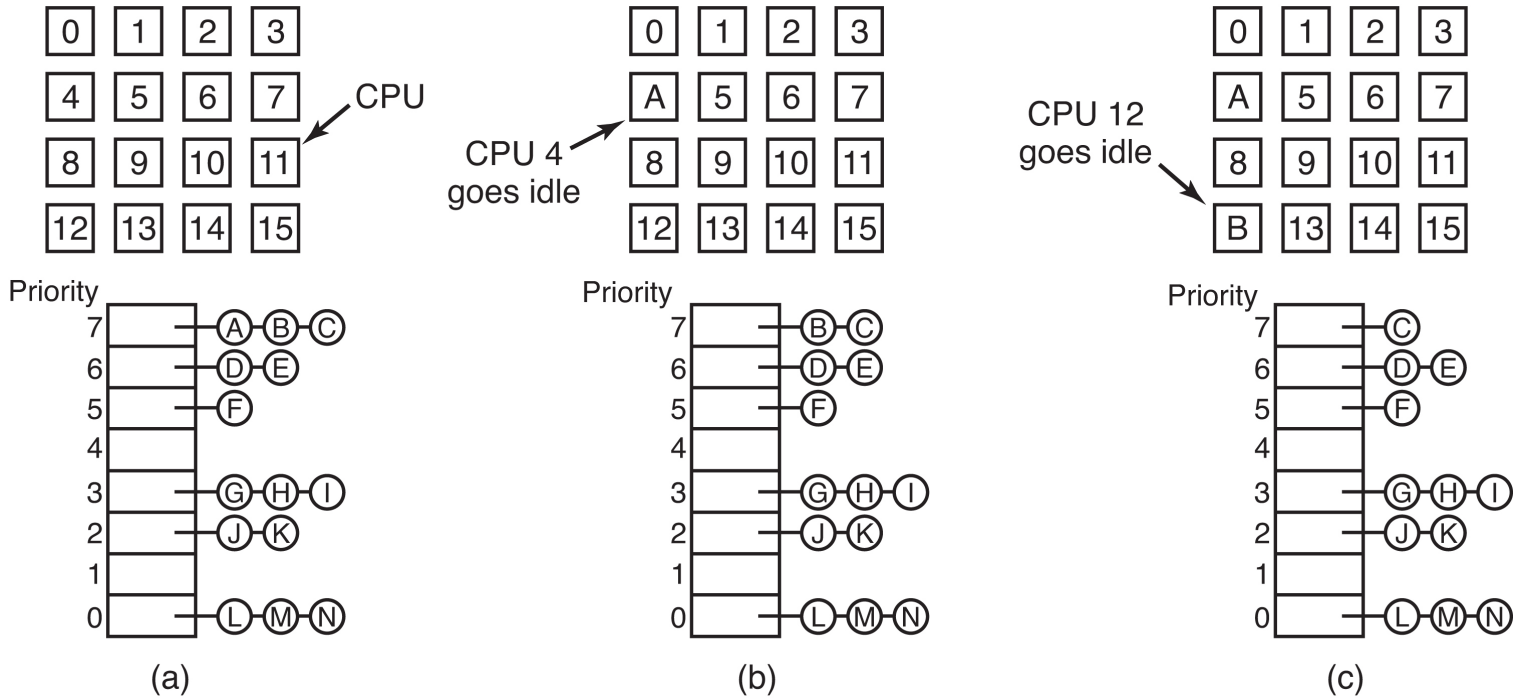
- The TSL instruction can fail if the bus cannot be locked
- This figure shows a sequence of events where the failure is demonstrated

Synchronization When Cache is Involved



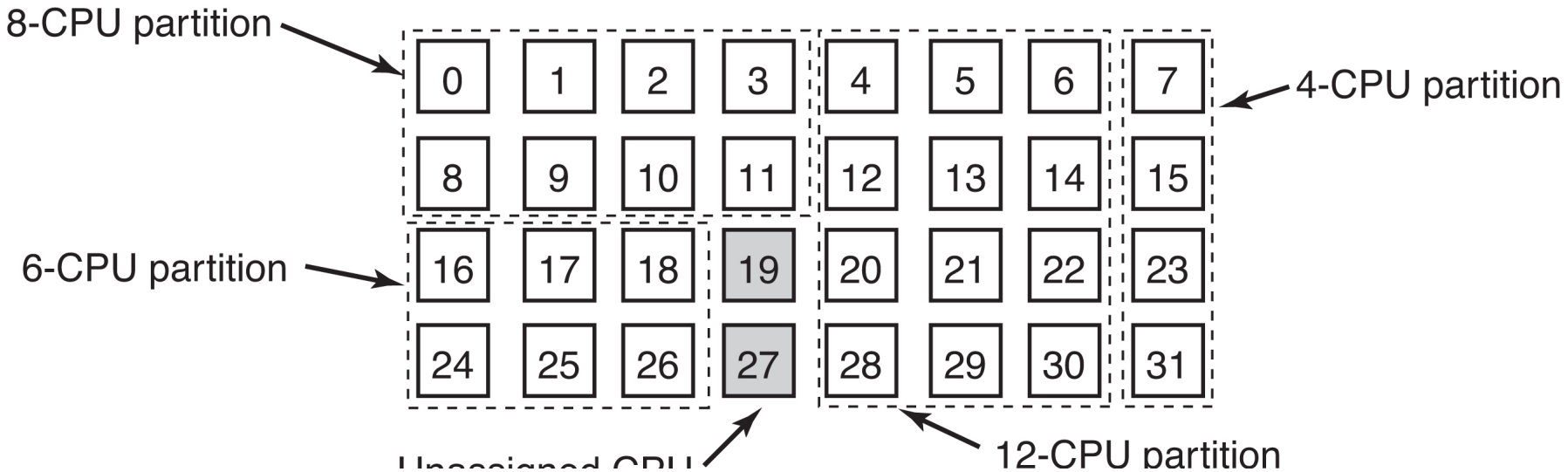
- A lock is stored in RAM as a part of a cache line
- Acquiring a lock moves the cache line to a core's local cache
- When multiple cores try to acquire the lock, the cache line is shuttled among the cores — cache thrashing
- Solution: use multiple locks and cores spinning on their private locks

Scheduling: Time Sharing



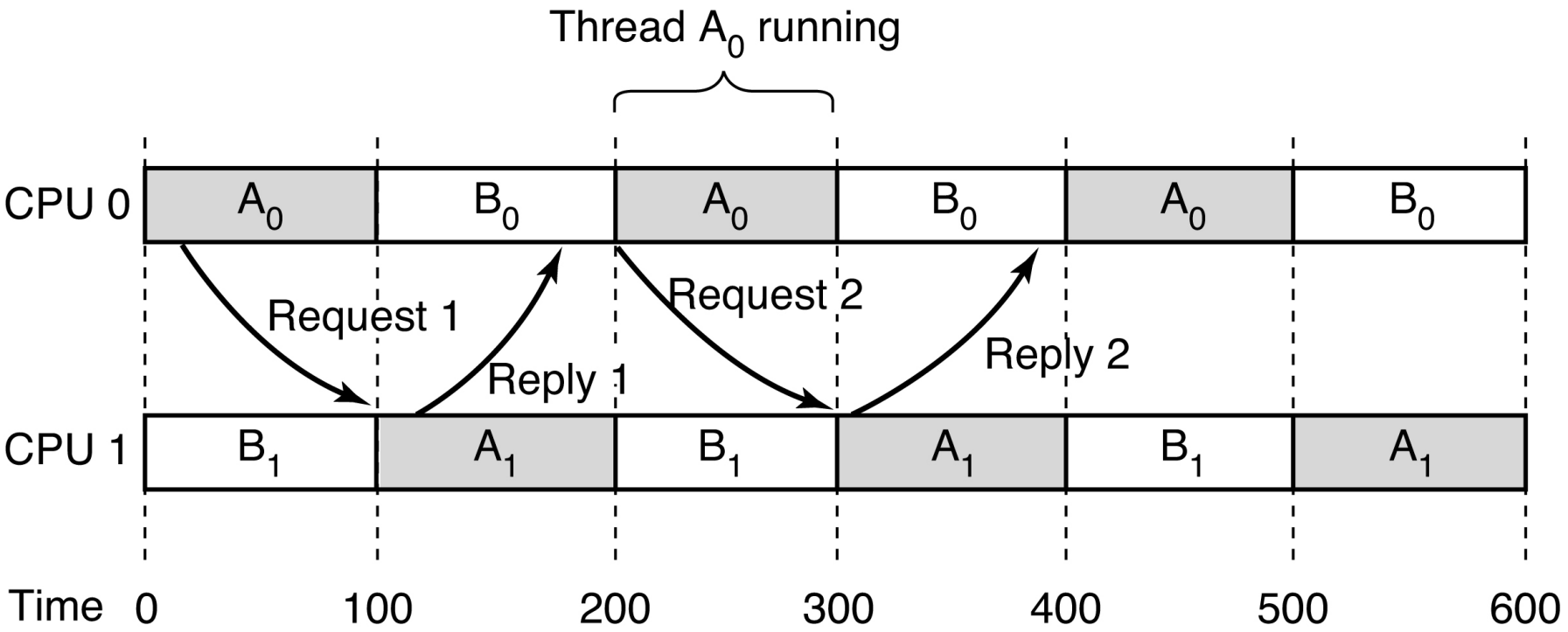
- Use a single data structure for scheduling a multiprocessor
- Considerations
  - Allow a thread holding a lock to run a little longer
  - Run a thread again on the same core as last time

Scheduling: Space Sharing



- Schedule multiple threads at the same time, if they need to work together
- A set of 32 CPUs is split into four partitions
- 2 CPUs are available

Threads Out of Sync



- Thread A_0 and A_1 need to communicate
- But they are scheduled out of sync

Gang Scheduling

- Groups of related threads are scheduled as a unit, a gang
- All members of a gang run at once on different timeshared CPUs
- All gang members start and end their time slices together

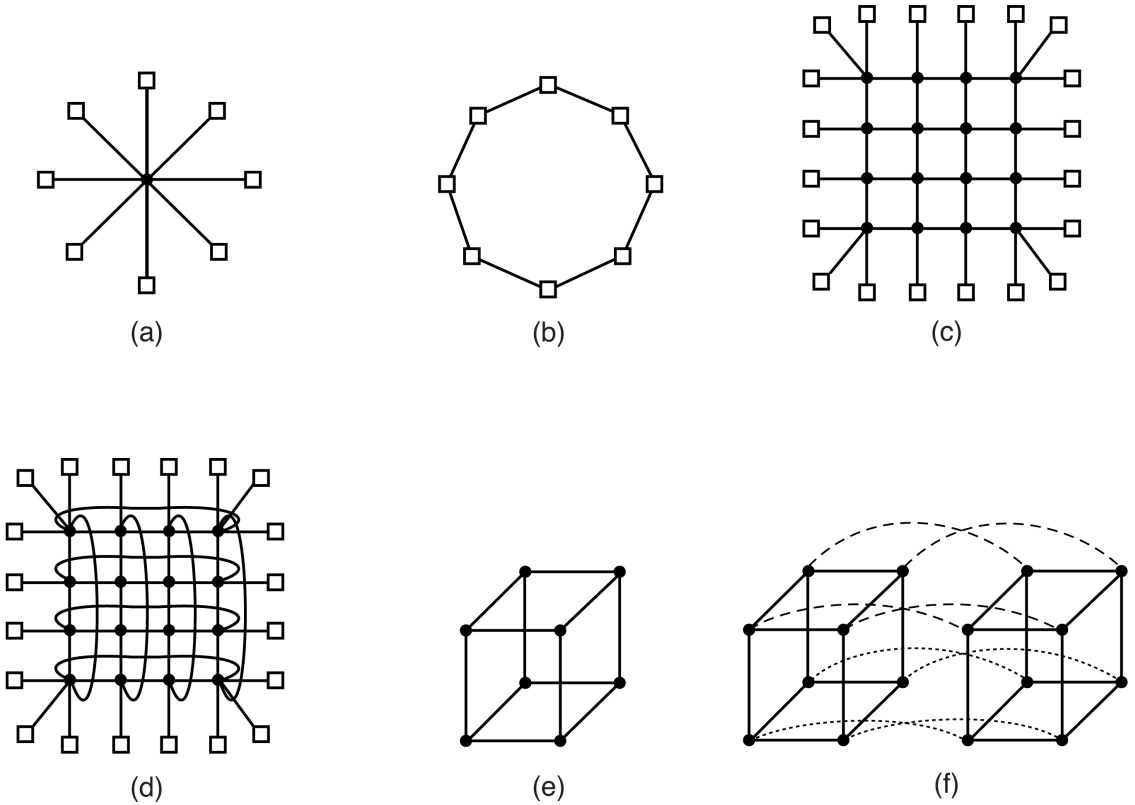
Multicomputers

- Cluster computers
- A group of stripped down PC, without keyboard, mouse, monitor, with a high performance network card
- Headless

Interconnection Topology

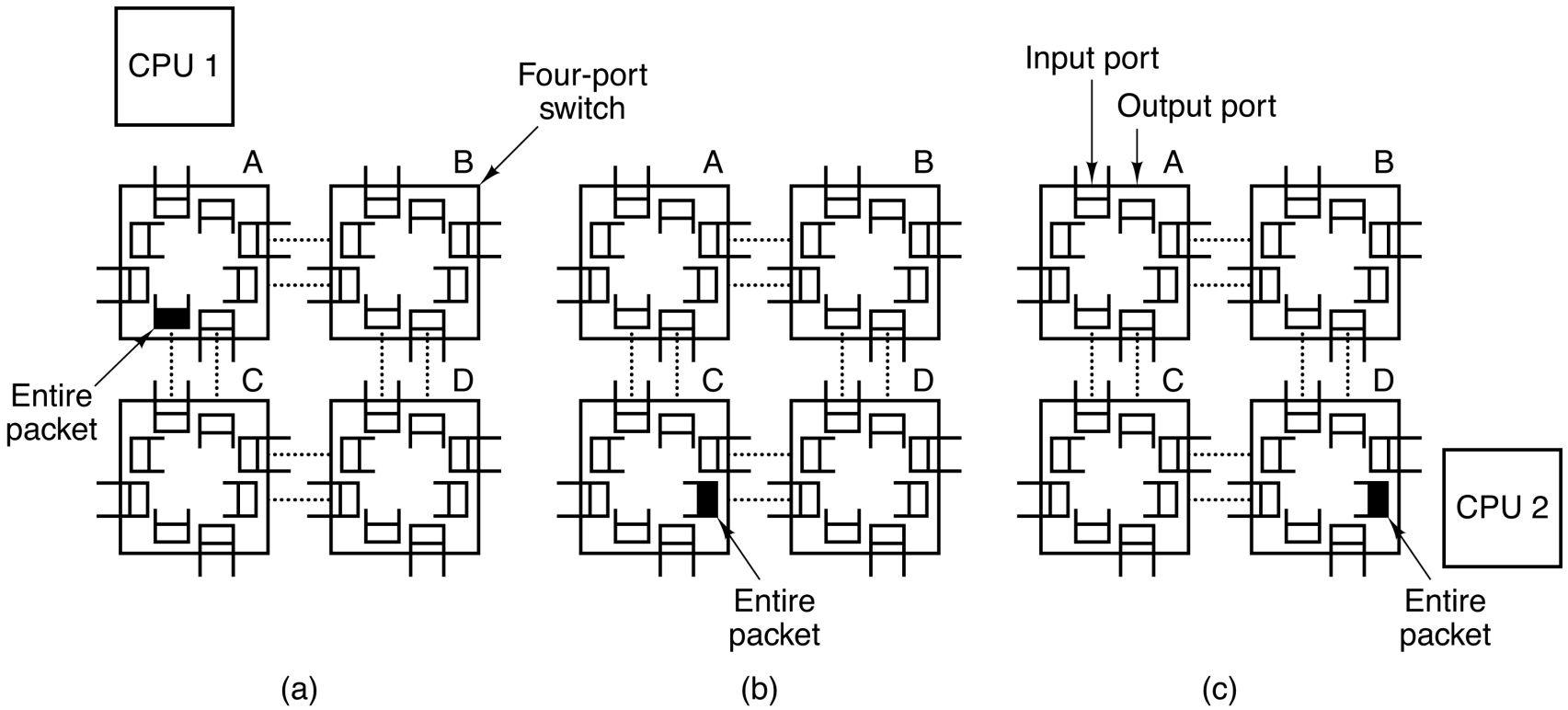
- Star
  - Switched ethernet
- Ring
- Mesh or grid
  - Torus
- Cube
- Hypercube

Various Topologies



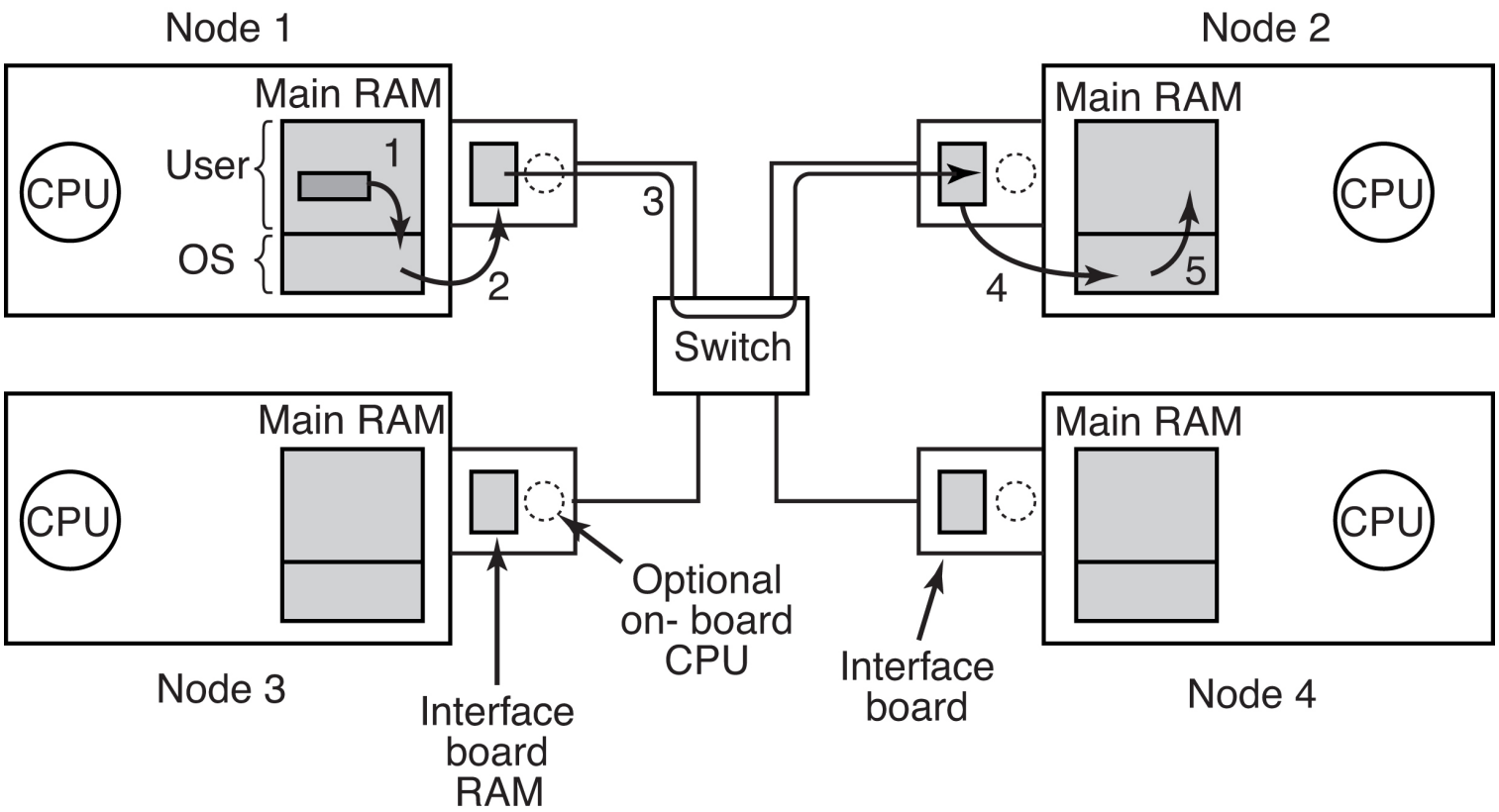
- A single switch, a ring, a mesh, a double torus, a cube, a 4-d hypercube
- Performance metrics: diameter, fault tolerance

Store & Forward Packet Switching



- A 4-port switch
- 2 switching schemes
- Store & forward packet switching incurs long latency
- Circuit stitching, wormhole routing

Network Interfaces



- Excessive copying causes long latency
- Solution: use 2 network cards
- One mapped to user space for applications
- The other mapped to kernel pace

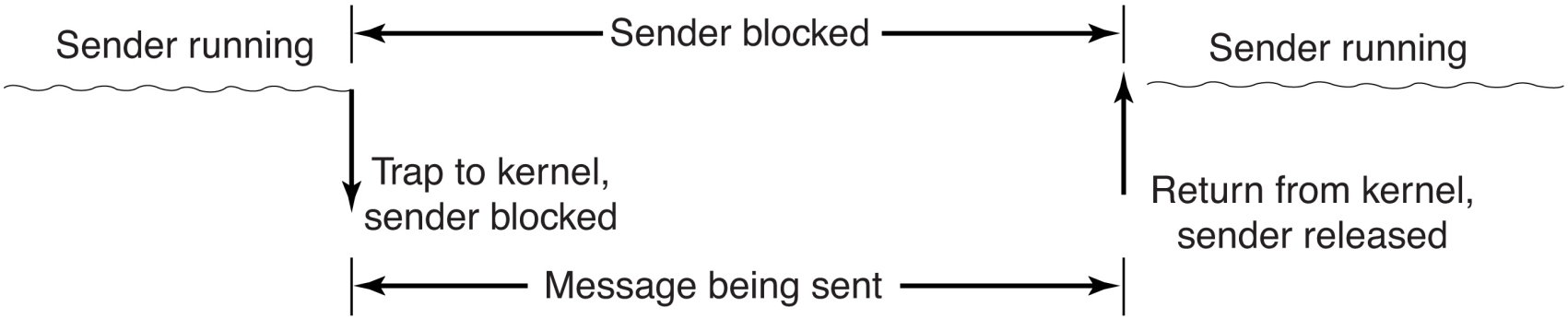
Node to Network Communication

- Network processor — multicasting, compression, encryption
- Network board RAM is mapped into user space
- Use DMA to copy from RAM to network board RAM
- Need I/O MMU
- Pin and unpin pages before and after DMA
- Remote DMA

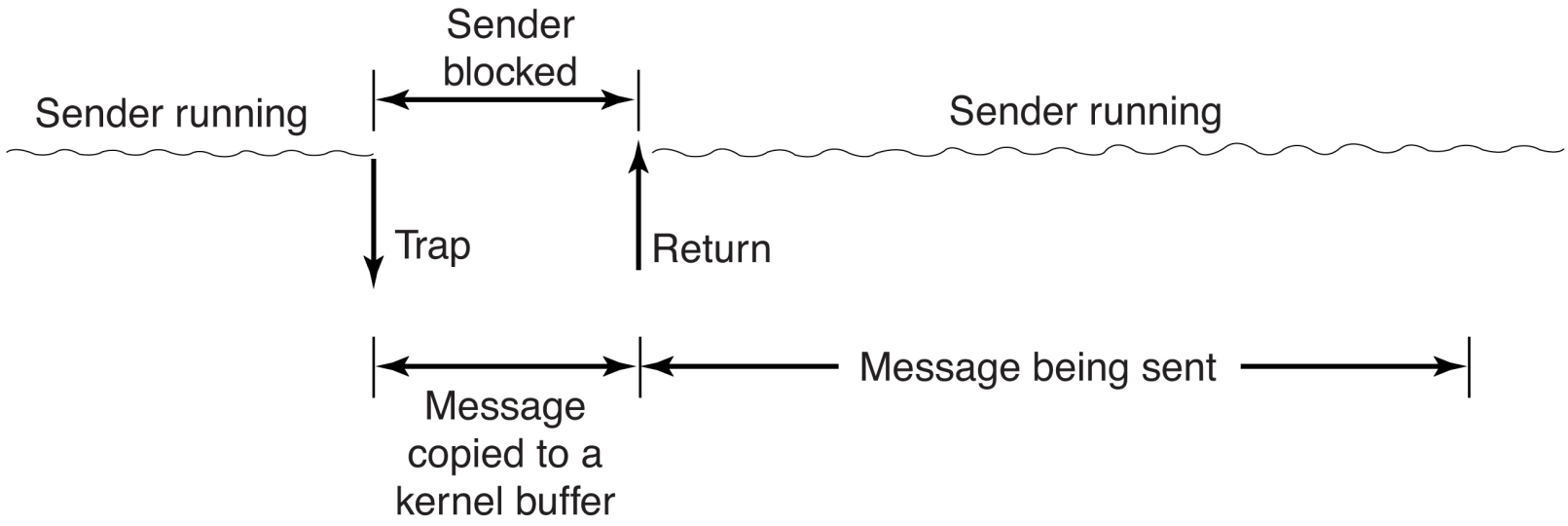
User-Level Communication

- Send and receive
  - `Send(dest, &mptr)`
  - `Receive(addr, &mptr)`
- Blocking calls: synchronous calls
- Nonblocking calls: asynchronous calls

Blocking versus Nonblocking Calls



(a)



(b)

Choices on the Sending Side

- Blocking send (CPU idle during transmission)
  - Most convenient
- Nonblocking send with copy (CPU time wasted for the extra copy)
- Nonblocking send with interrupt (makes programming difficult)
- Copy on write (extra copy probably needed eventually)

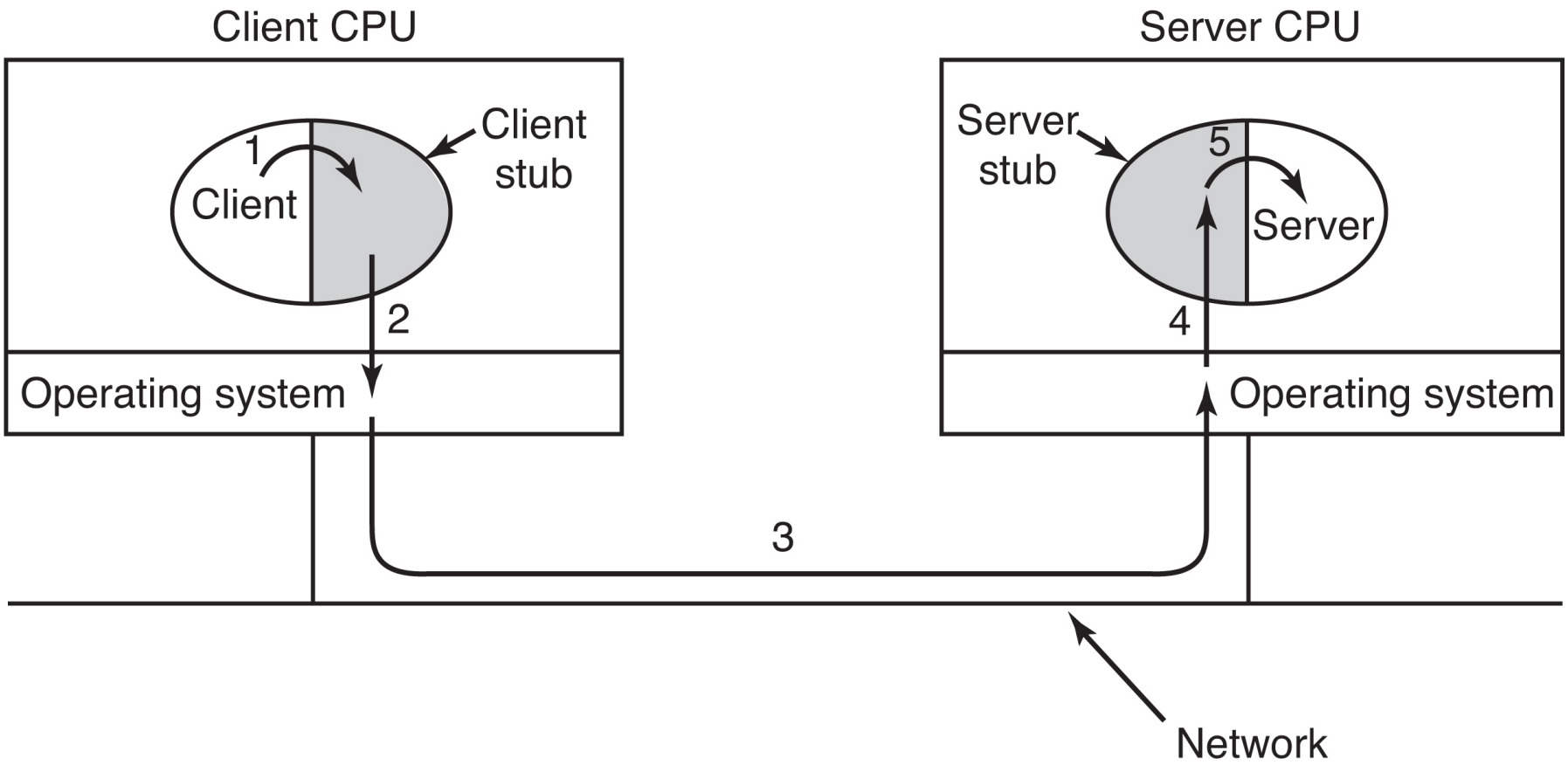
Choices on the Receiving Side

- Blocking
- Interrupt
- Use a poll procedure
- Pop-up thread
- Active messages: the message contains the address of the handler
  - When sender and receiver trust each other completely

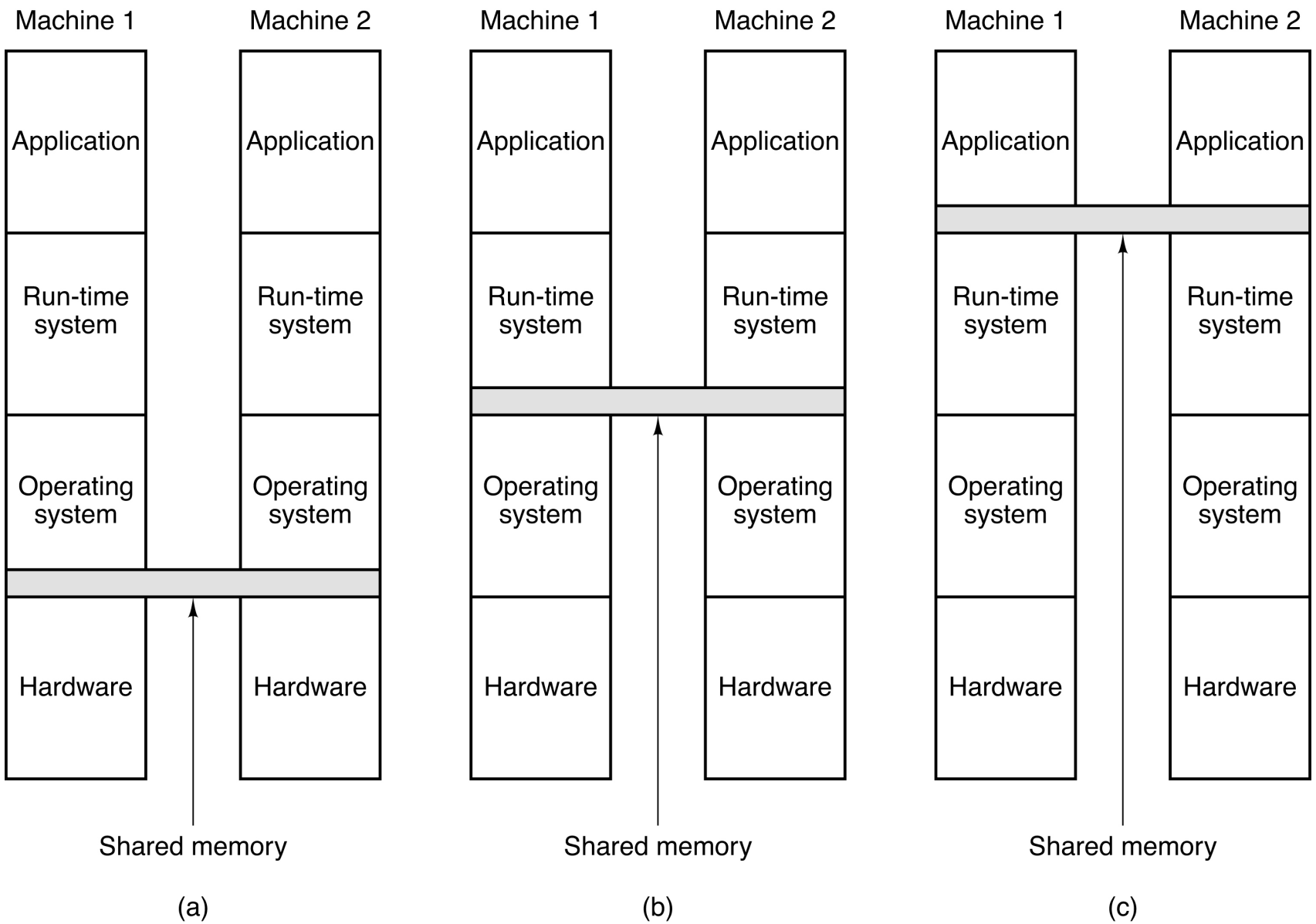
Remote Procedure Call

- Replace send/receive (I/O ops) by procedure calls
- The calling procedure is the client
  - Client stub
- The called procedure is the server
  - Server stub
- Packing the parameters is called marshalling

Steps in Making a Remote Procedure Call



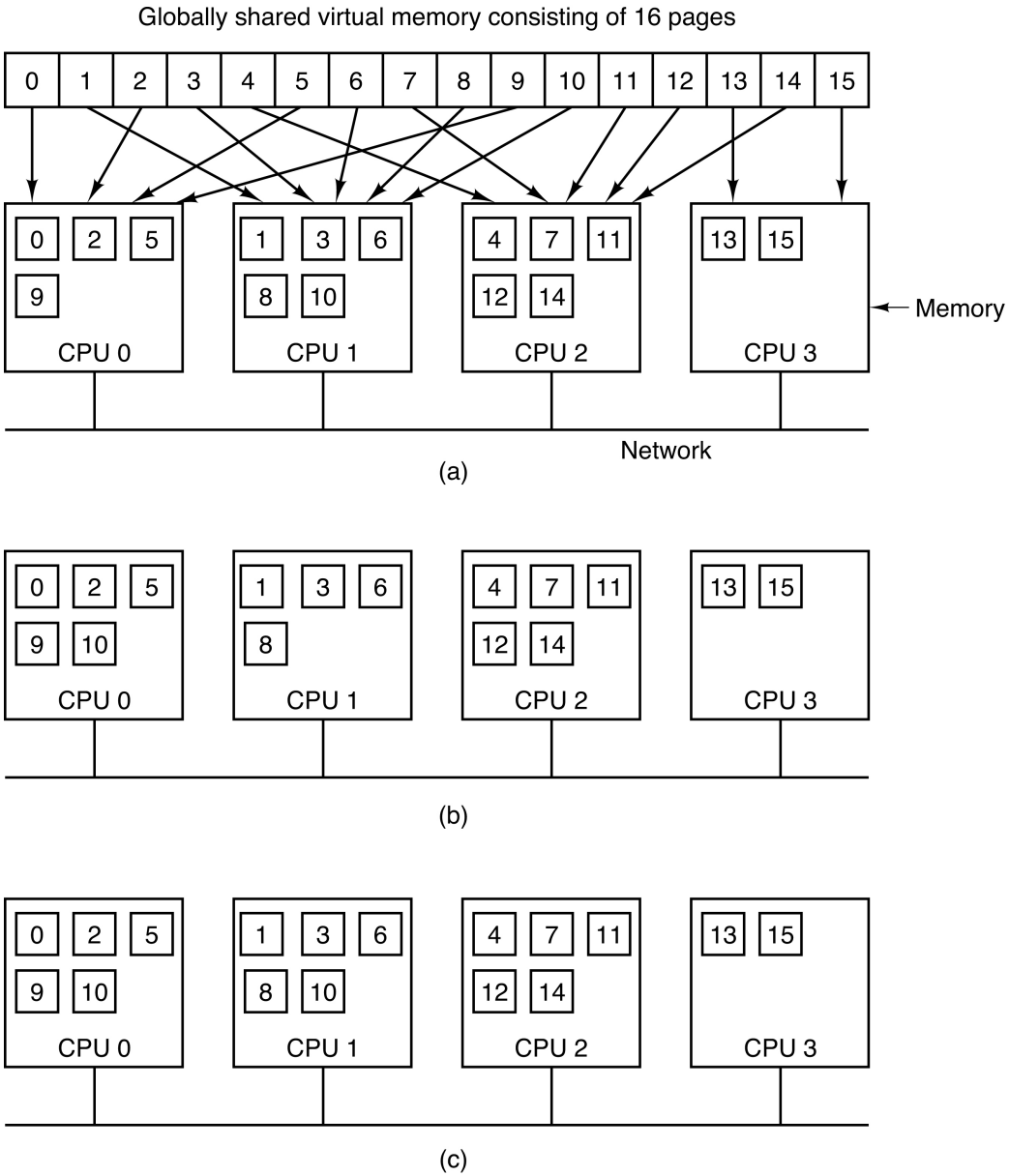
Implementing Distributed Shared Memory



Distributed Shared Memory

- The OS is satisfying page faults from remote RAM instead of from local disk
- Replication
  - Easy to achieve for read-only pages
  - For read-write pages: signal all nodes, get acknowledgment, then write
- False sharing

Distributed Shared Memory



- Page 10 is moved from CPU 1 to CPU 0
- Page 10 is replicated if read-only