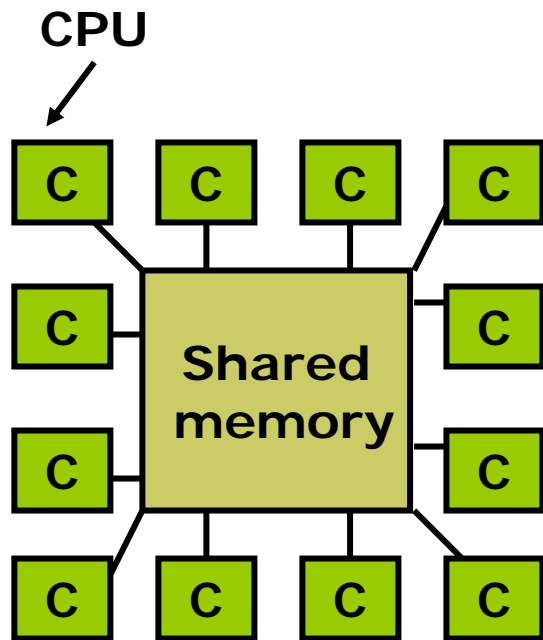


# Shared-memory Multiprocessors

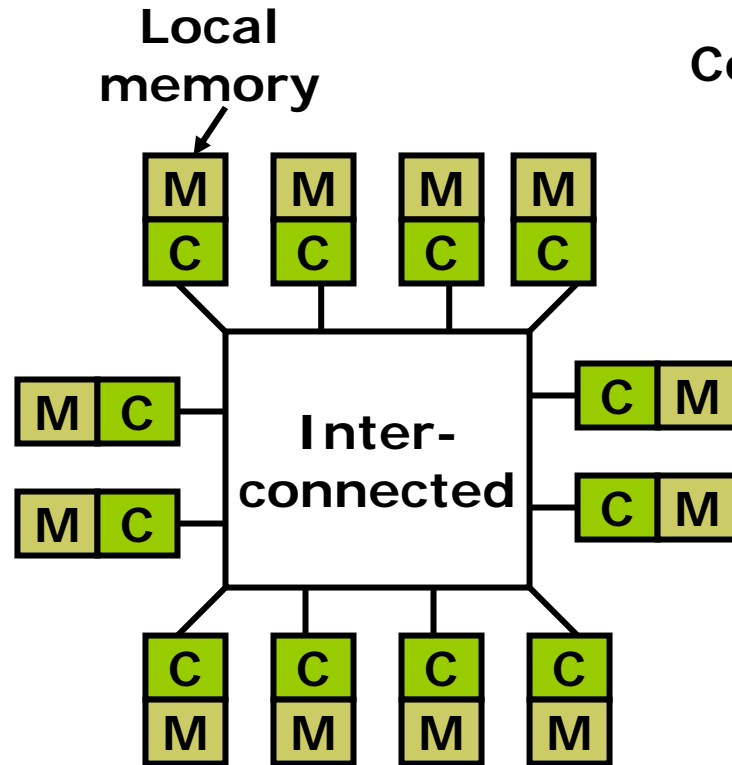


Presented by Jia Guo

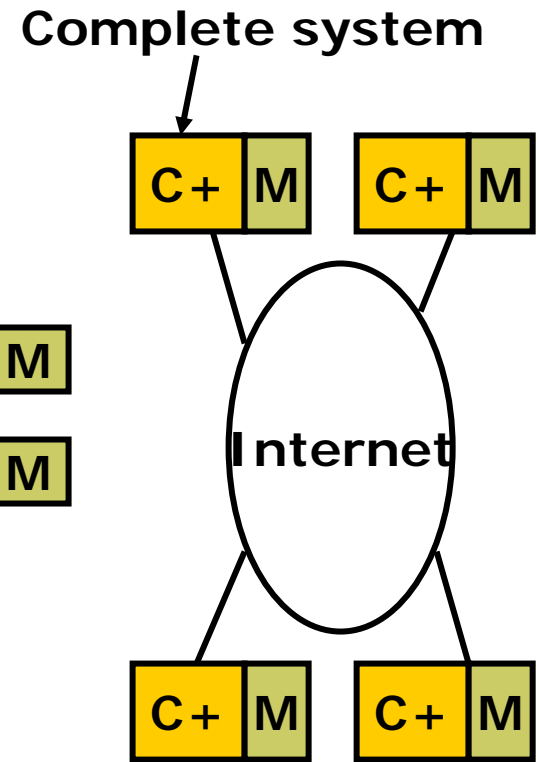
# Multiprocessor



A shared-memory multiprocessor



A message-passing multicomputer



A wide area distributed system

# Outline

---

- ❑ Multiprocessor types
- ❑ Multiprocessor operating system types
- ❑ Synchronization
- ❑ Scheduling
- ❑ Distributed shared memory

# Multiprocessor types

---

- UMA (Uniform Memory Access)
- NUMA (Nonuniform Memory Access)
  - Single address space
  - Access to remote memory slower than access to local memory
  - Access to remote memory: LOAD, STORE
  - NC-NUMA and CC-NUMA (Cache-Coherent NUMA)

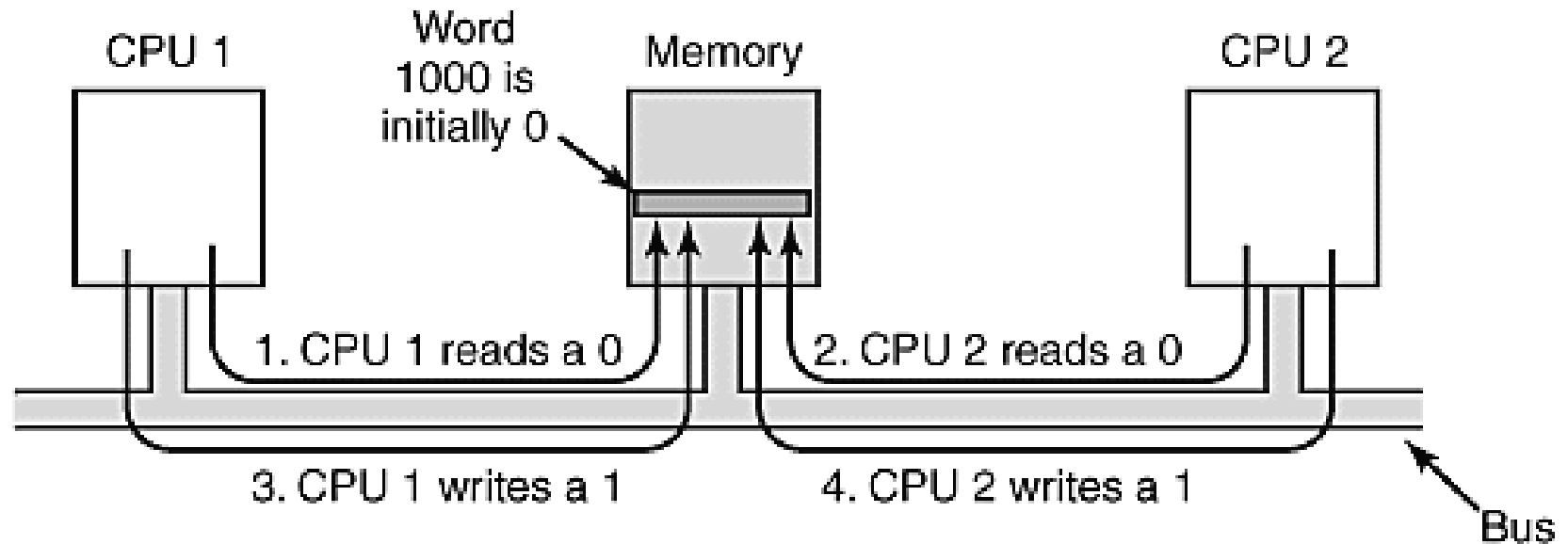
# Multiprocessor operating system types

---

- ❑ Each CPU has its own operating system
- ❑ Master-Slave multiprocessors
- ❑ Symmetric Multiprocessors (SMP)
  - Only one copy of the operating system in memory
  - Split OS into independent critical regions
  - Difficulties in designing such a system

# Synchronization

- ❑ The need for a proper mutex protocol for all CPUs.
- ❑ Example:



- ❑ TSL instruction first lock the bus

# Spin lock

---

- ❑ Massive load on bus or memory
- ❑ Caching?
  - Cache operates in cache lines (32/64 bytes)
  - TSL instruction needs exclusive access to the cache line containing the lock
  - The CPU holding the lock will access the data adjacent to the lock in the cache line
  - The cache line is shuttled constantly between the lock owner and the lock requestor.

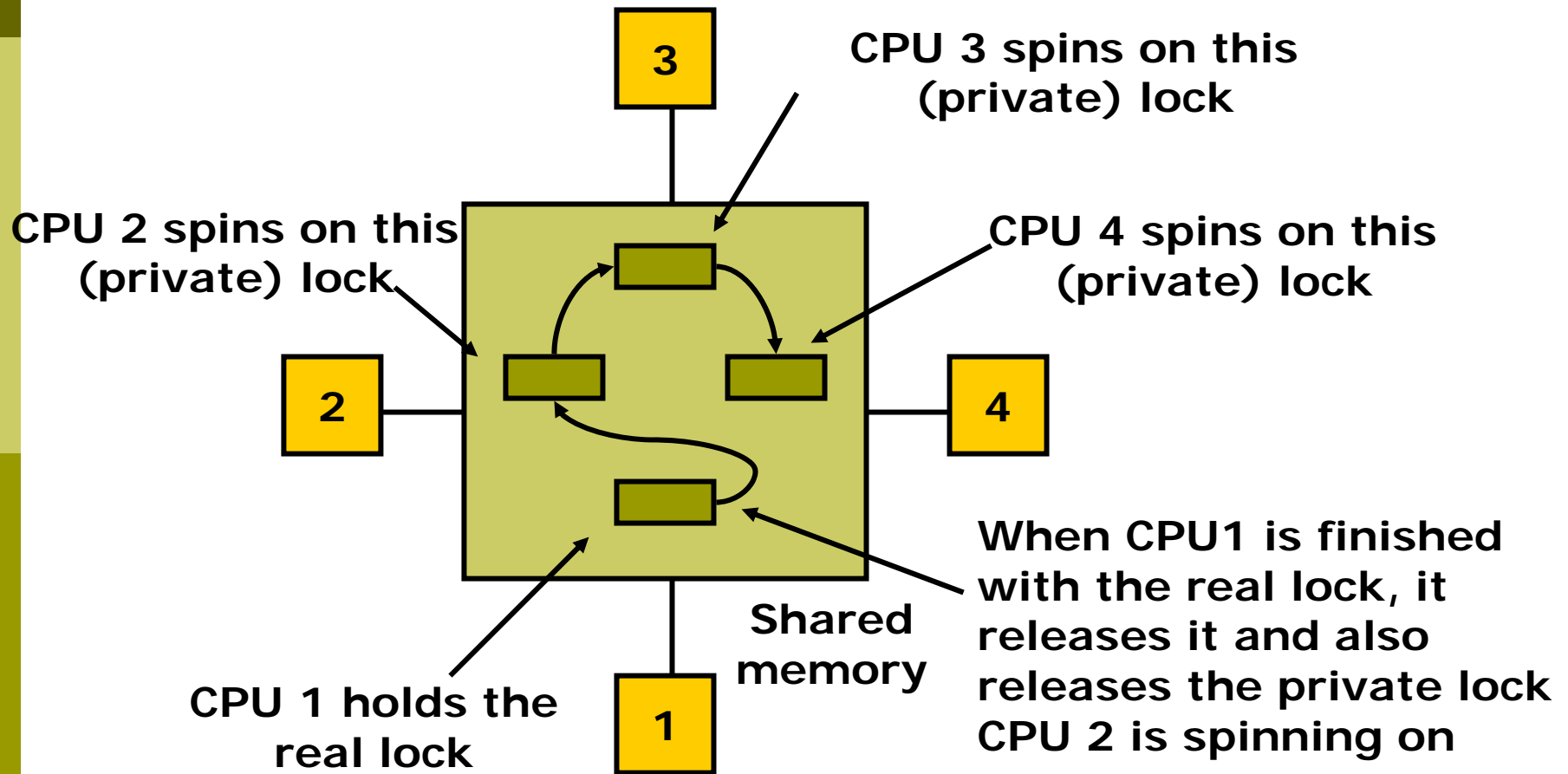
# Solutions

---

- ❑ Change writes to reads
  - Requesting CPUs do a poll of read first, then, TSL
  - Lock holder only reads the variables in the same cache line
- ❑ Use Ethernet binary exponential backoff algorithm (Anderson, 1990)
- ❑ Use of multiple locks (Mellor-Crummey and Scott, 1991)

# Multiple locks to avoid cache thrashing

---



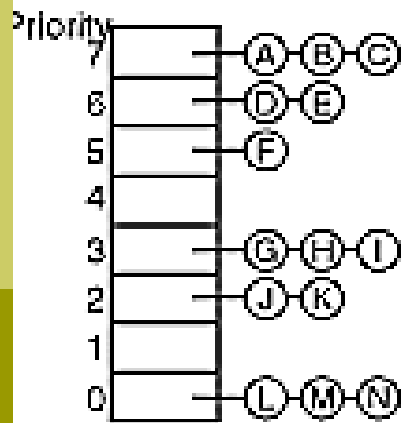
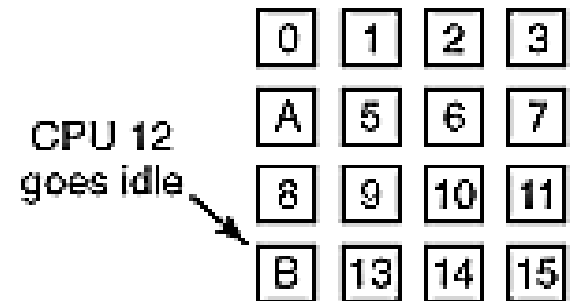
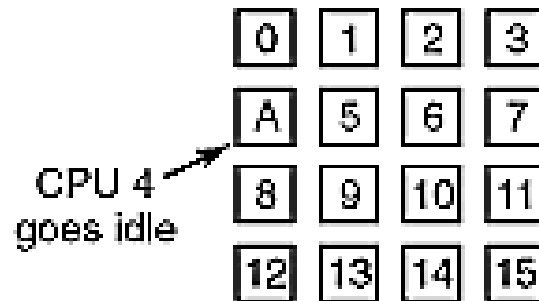
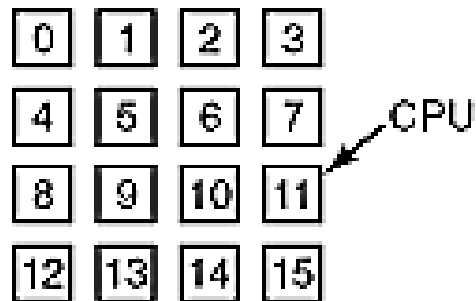
# Multiprocessor scheduling

---

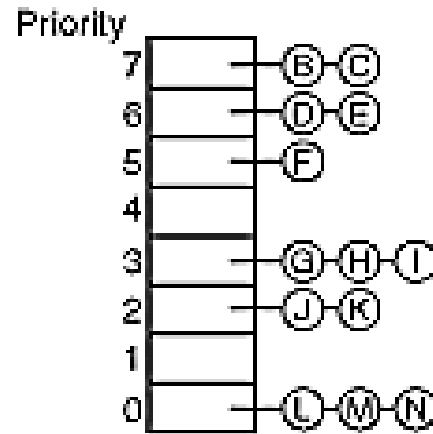
## □ Time sharing

- A set of priorities, each has a list of processes
- When a CPU is idle, it chooses the process from the list of highest priority queue.
- Load balancing (+), scalability (-), context switch overhead(-)
- Good for unrelated processes

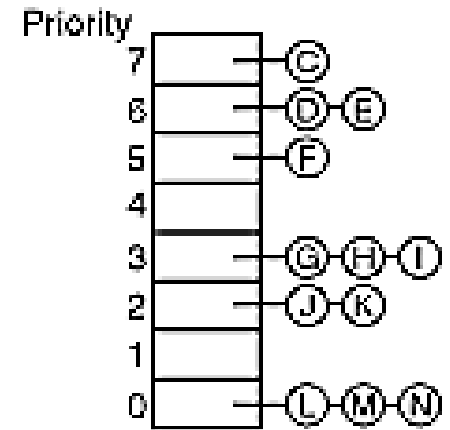
# Example



(a)



(b)



(c)

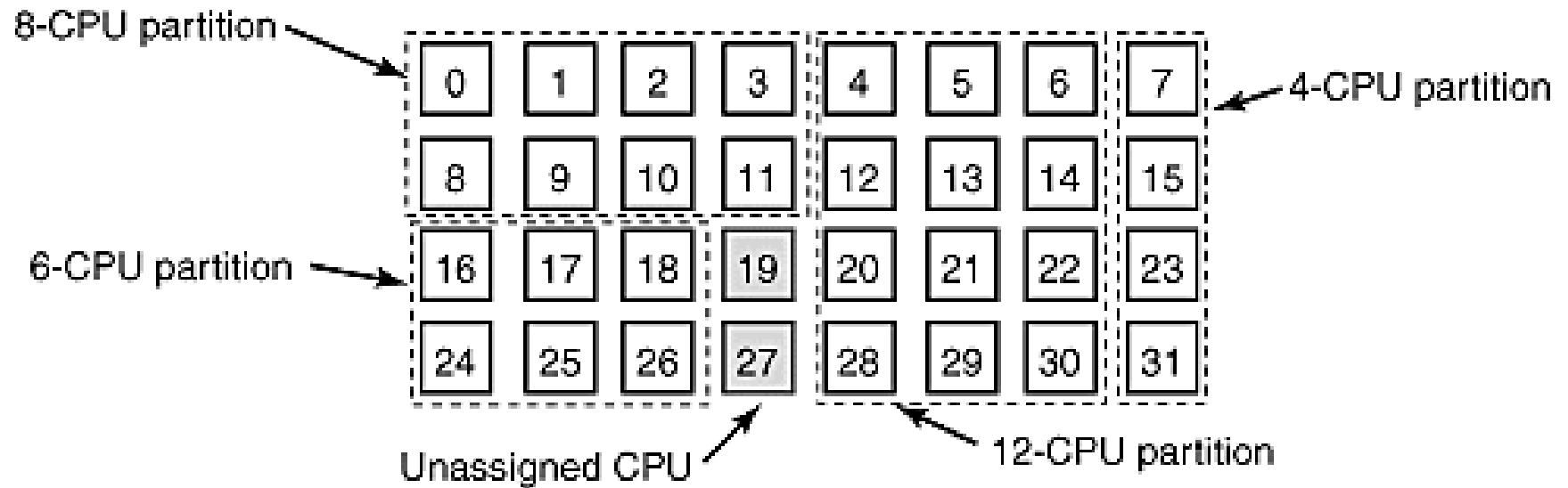
# Space sharing

---

- ❑ Used when processes are related
- ❑ Schedule multiple processes at the same time across multiple CPUs
- ❑ Each process holds its CPU until it finishes
- ❑ No context switch (+), time wasted when a CPU is blocked (-)

# Example

---

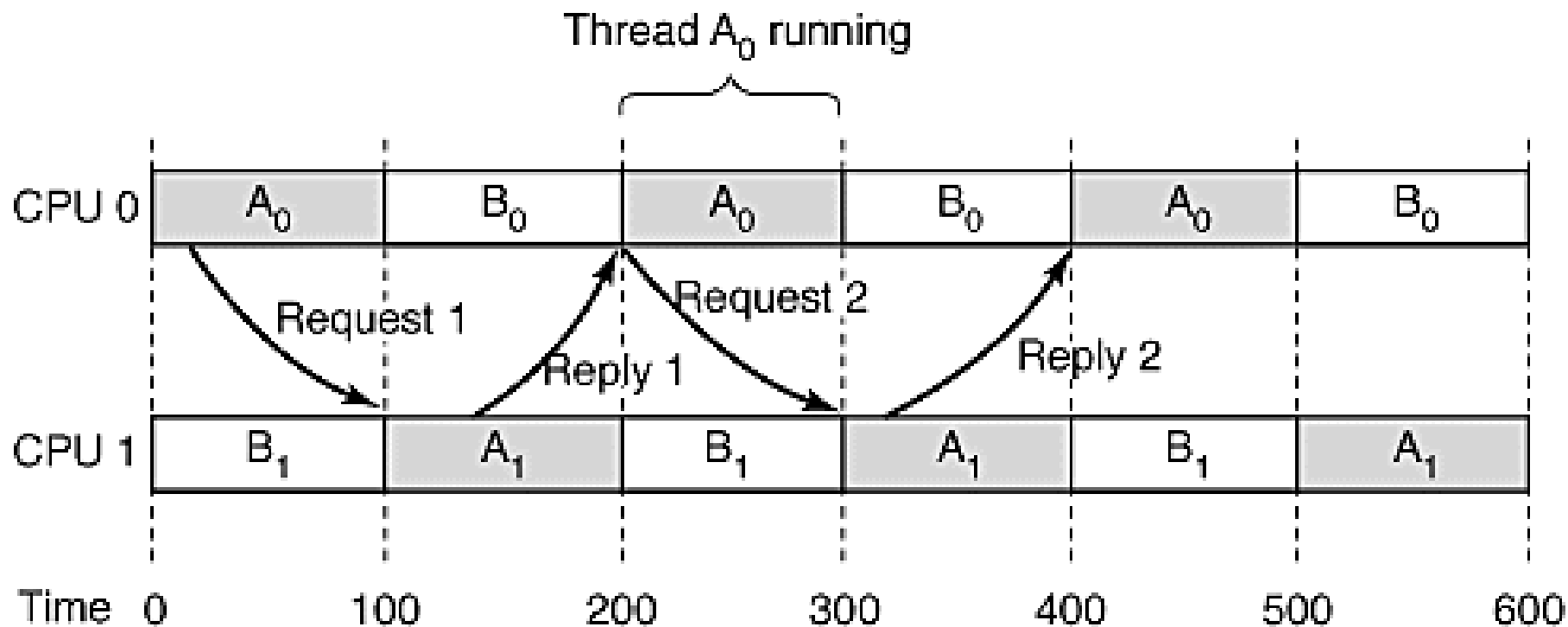


# Gang scheduling

---

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

# Problem in NOT gang scheduling



# Example

---

		CPU					
		0	1	2	3	4	5
Time slot	0	A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	A <sub>4</sub>	A <sub>5</sub>
	1	B <sub>0</sub>	B <sub>1</sub>	B <sub>2</sub>	C <sub>0</sub>	C <sub>1</sub>	C <sub>2</sub>
	2	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	E <sub>0</sub>
	3	E <sub>1</sub>	E <sub>2</sub>	E <sub>3</sub>	E <sub>4</sub>	E <sub>5</sub>	E <sub>6</sub>
	4	A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	A <sub>4</sub>	A <sub>5</sub>
	5	B <sub>0</sub>	B <sub>1</sub>	B <sub>2</sub>	C <sub>0</sub>	C <sub>1</sub>	C <sub>2</sub>
	6	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	E <sub>0</sub>
	7	E <sub>1</sub>	E <sub>2</sub>	E <sub>3</sub>	E <sub>4</sub>	E <sub>5</sub>	E <sub>6</sub>

# Distributed shared memory

---

- ❑ A collection of workstations connected by a LAN share a single paged, virtual address space
- ❑ Goal:
  - Run multiprocessor programs on multicomputer.
  - Easy to program (logically shared memory), easy to build( no physically shared memory).
- ❑ Method:
  - Add additional software layer to deal with remote data access

# Treadmarks

---

- It provides a global shared address space across the different machines on a cluster.
- Performance issue
  - Release consistency
  - Multiple writer protocols

# References

---

- Andrew S. Tanenbaum: *“Modern operating systems”*
- Mellor-Crummey, J.M., and Scott, M.L.: *“Algorithms for Scalable Synchronization on Shared-Memory Multiprocessors.”*
- [www.wikipedia.com](http://www.wikipedia.com)
- <http://www.cs.rice.edu/~willy/TreadMarks/overview.html>