CMPE 200
Computer Architecture & Design

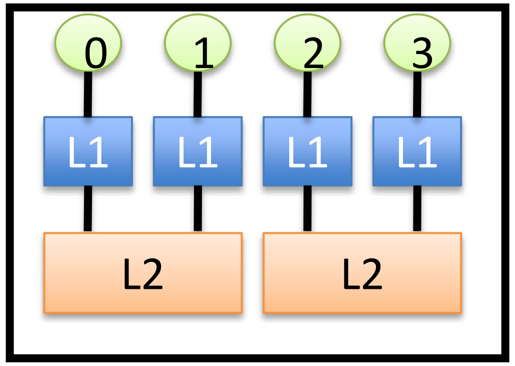
Lecture 5.
Parallel Processing (2)

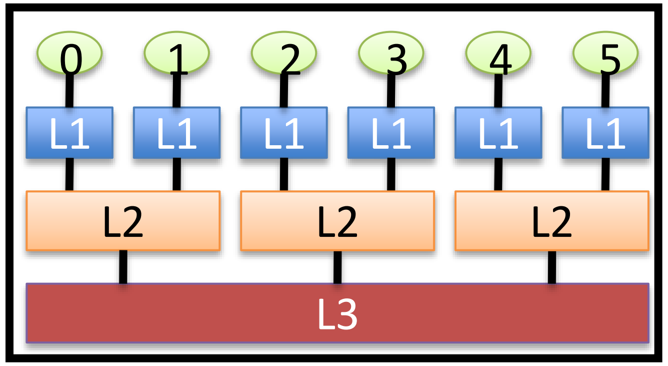
Haonan Wang

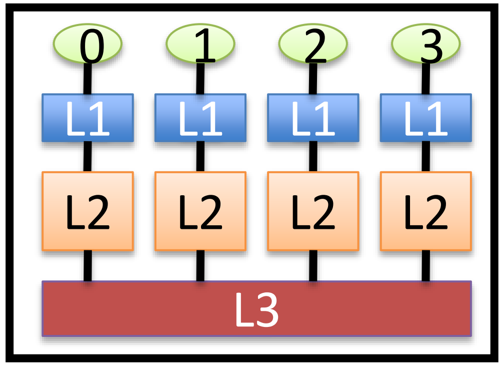
SJSU SAN JOSÉ STATE UNIVERSITY

Multiprocessor

- **Parallelism in processor unit → Multiprocessor**
  - Multiple CPUs or Multiple Cores within a CPU
- **Two representative types:**
  - Shared-memory
  - Message-passing
- **Platform variety: #cores, hierarchy, #memory controllers, interconnect, …**



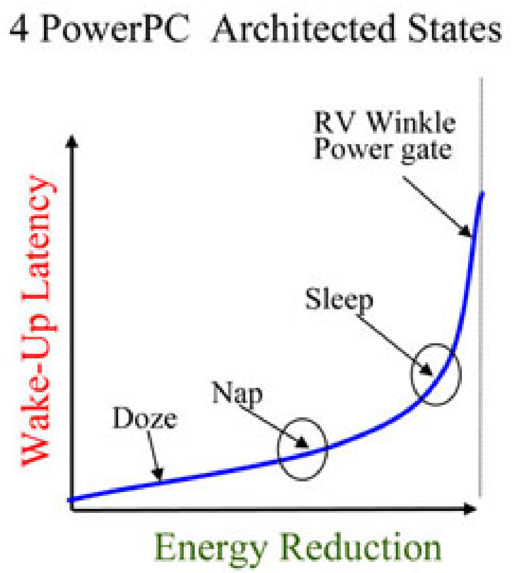




Multicore Power Management

- **For power management, you can turn off whole cores, vary the core frequency (lower or higher), and run fewer threads.**

- **Power reduction modes**
  - Nap: optimal wakeup latency relative to power reduction
    - Clocks turned off to the execution units, core frequency reduced, caches and TLB’s still active, remain coherent
  - Sleep: optimal power reduction relative to wakeup latency
    - Caches and TLB’s are purged, all clocks turned off, voltage lowered, some state info is retained



Multiprocessor Key Questions

- Q1 – How do they share data?
- Q2 – How do they coordinate?
- Q3 – How scalable is the architecture? How many processors can be supported?

Multiprocessor Key Questions

- **Q1 – How do they share data?**
- Q2 – How do they coordinate?
- Q3 – How scalable is the architecture? How many processors can be supported?

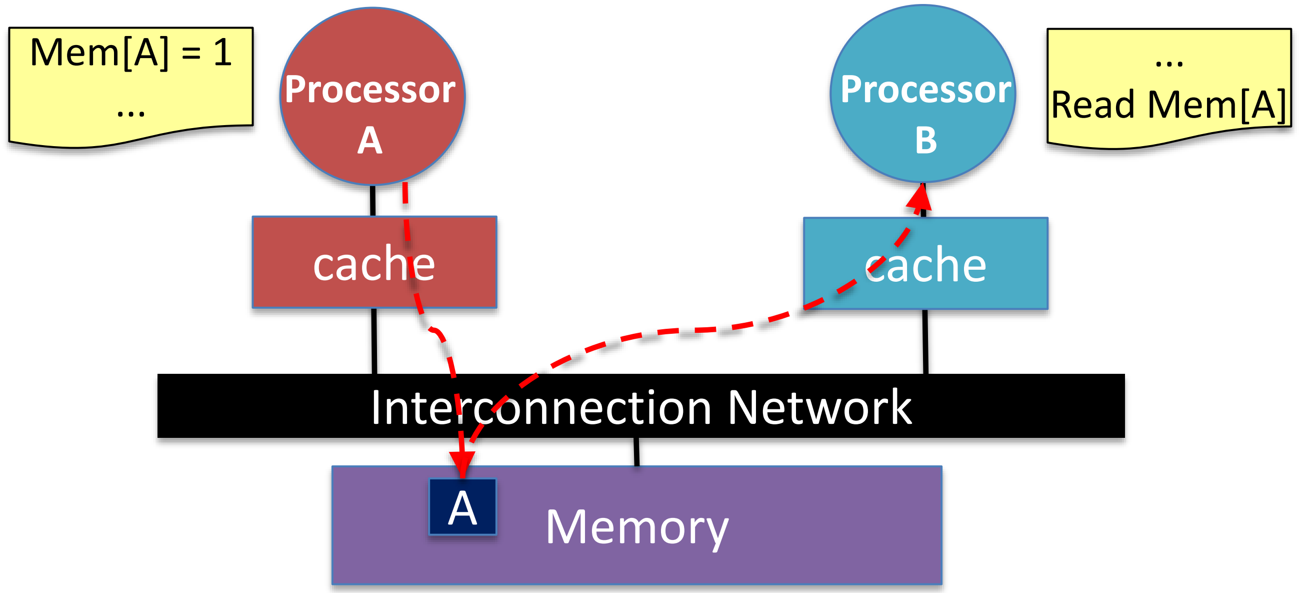
How to Share Data?

- **Shared Memory multi-Processor (SMP)**
  - Single address space shared by all cores
  - Cores coordinate/communicate through shared variables (via loads and stores)
    - Need synchronization primitives (e.g., locks)
  - Two styles
    - Uniform memory access (UMA): easier for programming
    - Non-uniform memory access (NUMA): more scalable & lower latency to local memory
- **Message Passing multi-Processor (MPP)**
  - Each core has its own private address space
    - Cores share data by *explicitly* sending and receiving information (message passing)
  - Coordination is built into MP primitives (message send & message receive)

Shared Memory Model

- **Parallel programs communicate through shared memory**
  - E.g., processor A writes to an address, then processor B reads from the same address

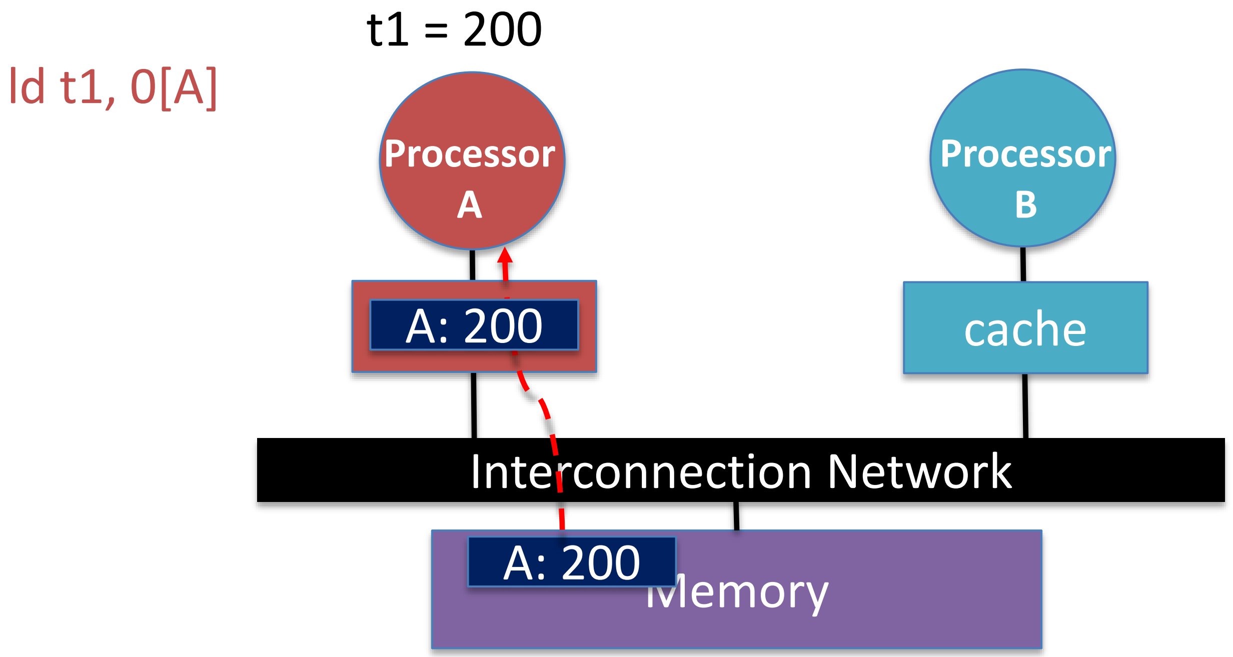
- **Each read should receive the most up-to-date value**



SJSU SAN JOSÉ STATE UNIVERSITY

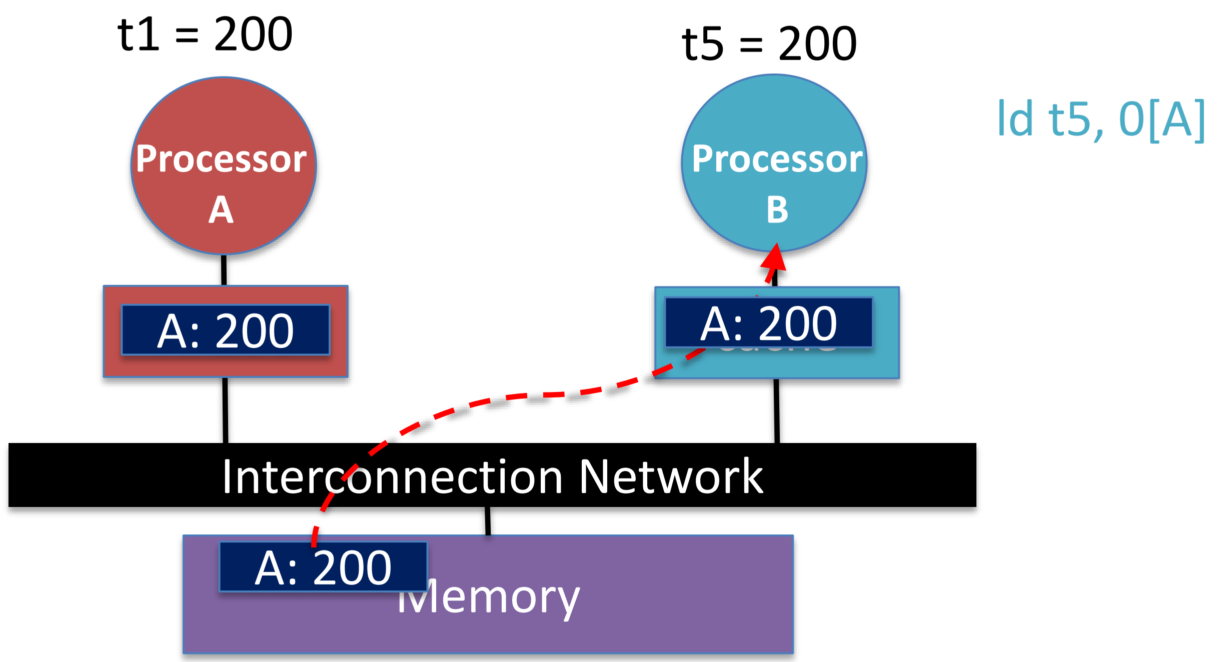
Shared Data Management

- **If multiple processors cache the same block in their private cache, how do they ensure they all see a consistent state?**



Shared Data Management

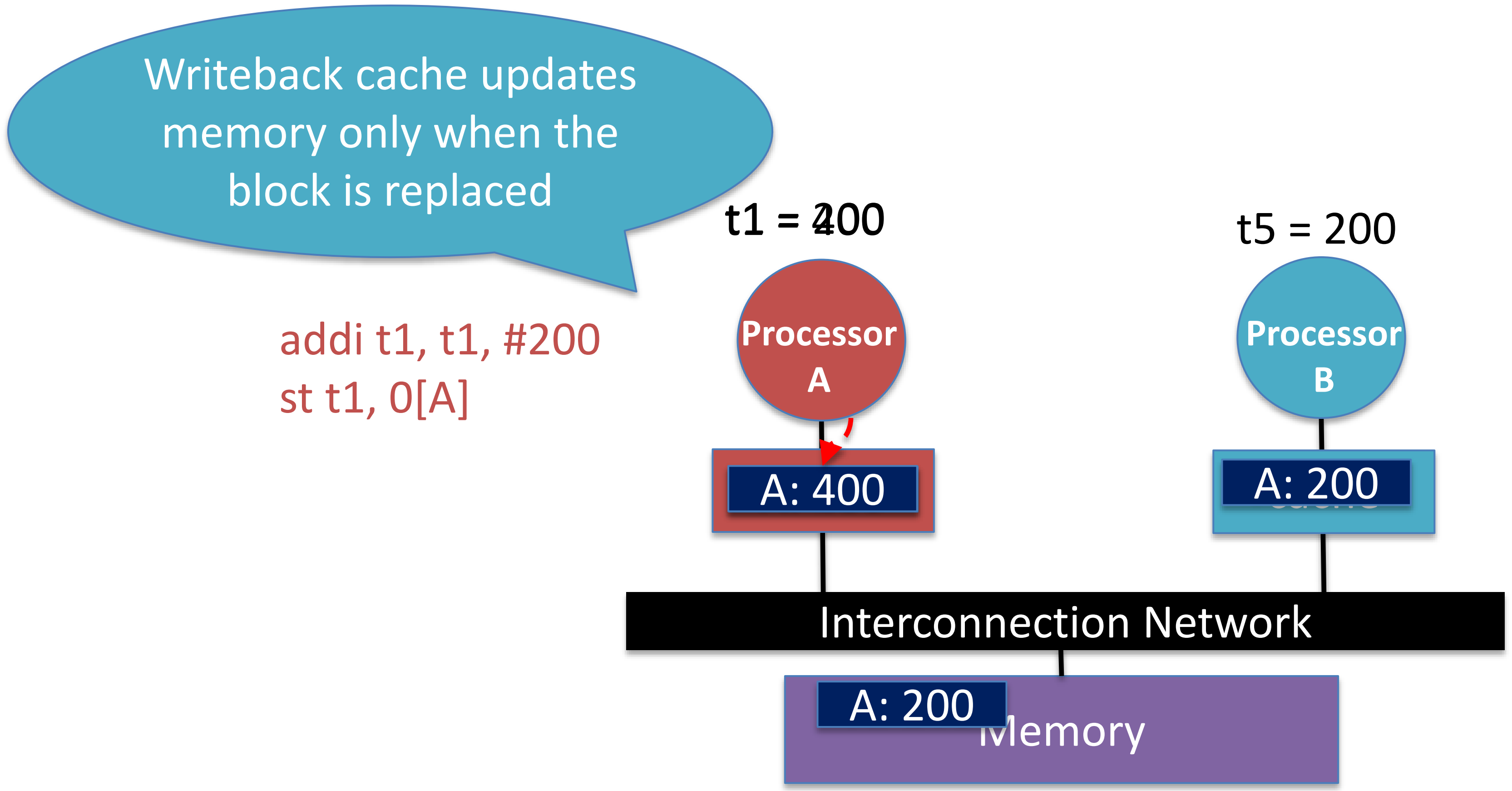
- **If multiple processors cache the same block in their private cache, how do they ensure they all see a consistent state?**



SJSU SAN JOSÉ STATE UNIVERSITY

Shared Data Management

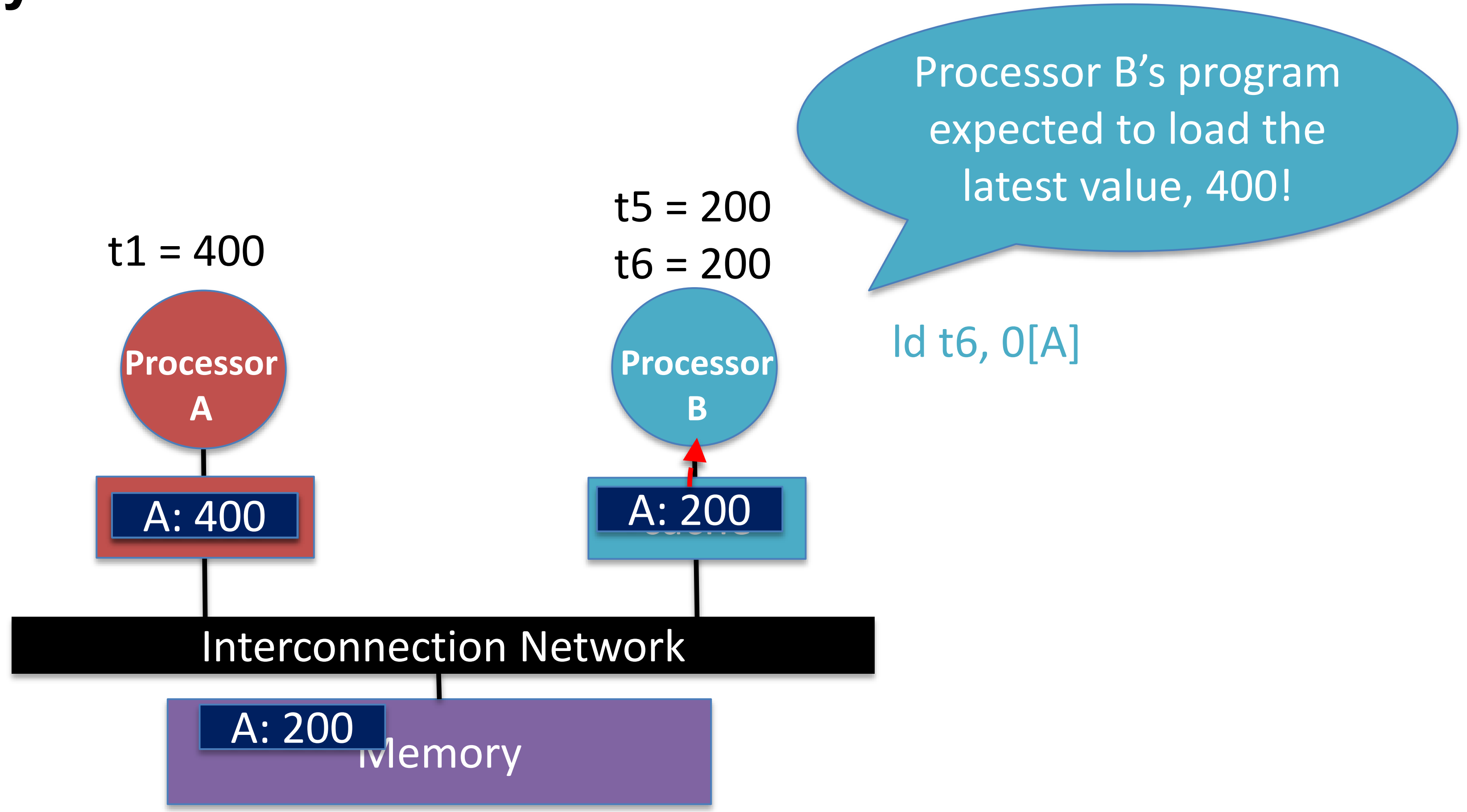
- **If multiple processors cache the same block in their private cache, how do they ensure they all see a consistent state?**



SJSU SAN JOSÉ STATE UNIVERSITY

Shared Data Management

- **If multiple processors cache the same block in their private cache, how do they ensure they all see a consistent state?**



SJSU SAN JOSÉ STATE UNIVERSITY

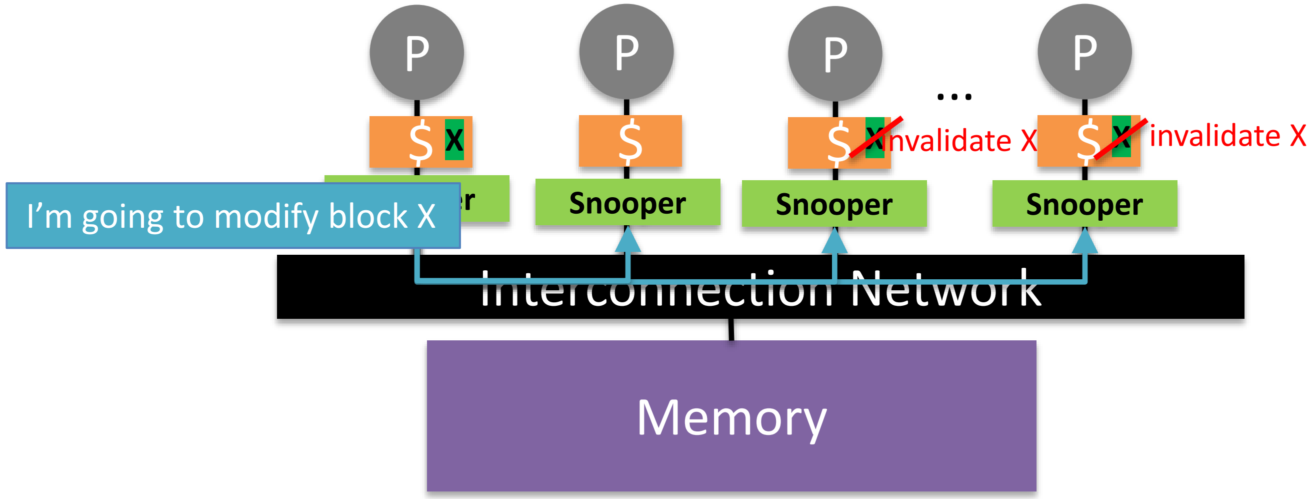
A Coherent Memory System

- **Coherence: writes to the same location are seen in the same order by all cores (serialized)**

- **Cache coherent multiprocessor: provide migration & replication**
  - Migration: data items are moved to local caches and used in a transparent fashion
    - Reduces latency and the bandwidth demand on the shared memory
  - Replication: simultaneous reads make copies of the data item in local caches
    - Reduces both latency and contention for a read shared data item

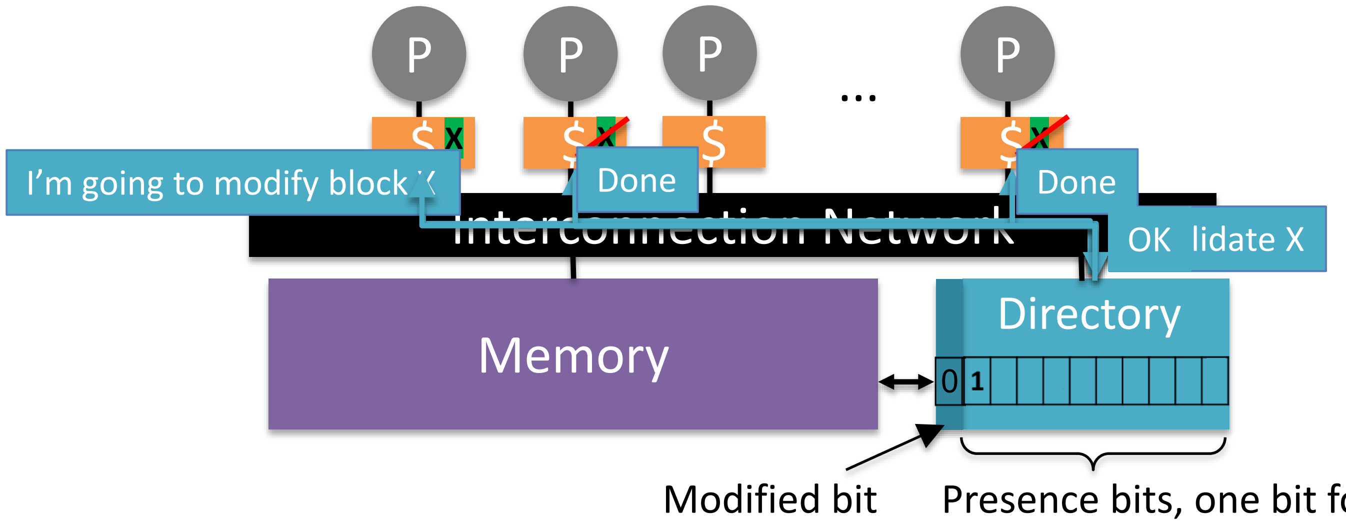
Cache Coherence Methods

- **Snooping-based protocol**
  - Each processor's cache controller constantly snoops on the bus
  - Processors observe other processors' actions
    - E.g., processor A makes "read-exclusive" request for A on bus, processor B sees this and invalidates its own copy of A



Cache Coherence Methods

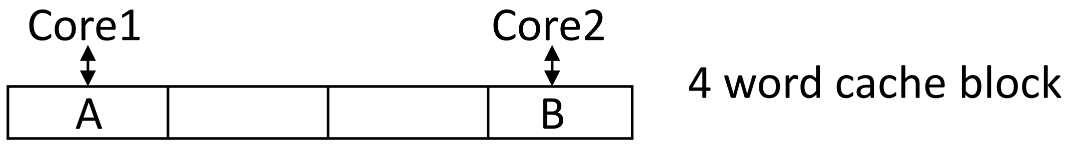
- **Directory-based protocol**
  - Directory tracks ownership (sharer set) for each block (who has what)
    - A *modified* bit and multiple *presence* bits per cache block
  - Directory coordinates invalidation appropriately
    - E.g., processor A asks directory for “read-exclusive” copy, directory asks processor B to invalidate the requested copy, waits for ACK from processor B, then responds to processor A



SJSU SAN JOSÉ STATE UNIVERSITY

Write Invalidate (All Other Copies)

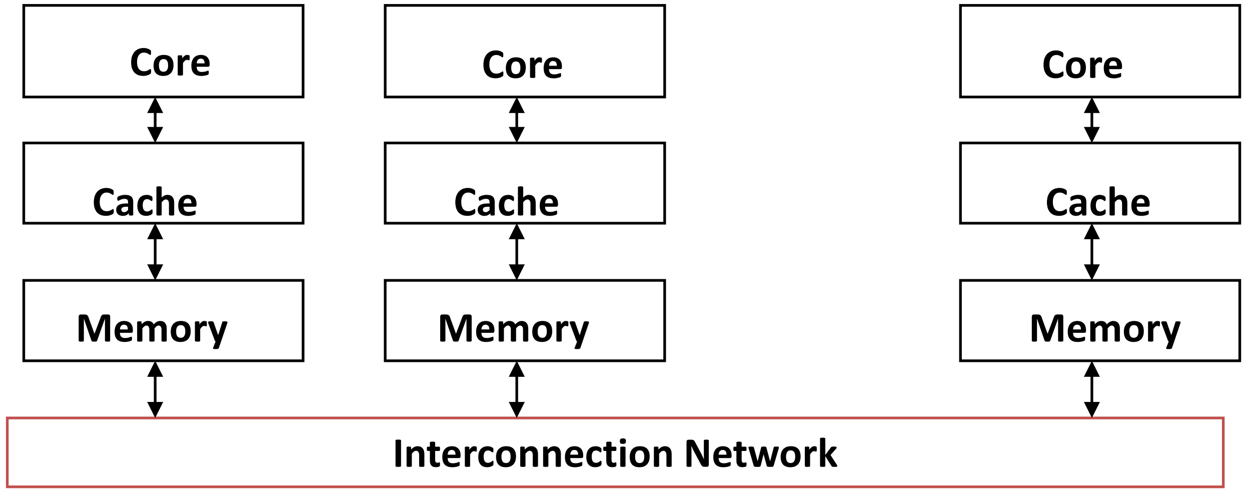
- **Can enforce write serialization**
  - Two processors attempt to write the same data at the same time → one of them wins the race → the other core’s copy is invalidated
  - The other core must obtain a new copy of the data → must now contain the updated value



- **False sharing problem**
  - Two cores writes to two different variables in the same cache block → increases miss rate
  - Compilers can help reduce this by allocating correlated data to the same cache block

Message Passing multi-Processors (MPP)

- **Each core has its own private address space**
  - Cores share data by explicitly sending and receiving information (message passing)
- **Coordination is built into message passing primitives (message send and message receive)**



SJSU SAN JOSÉ STATE UNIVERSITY

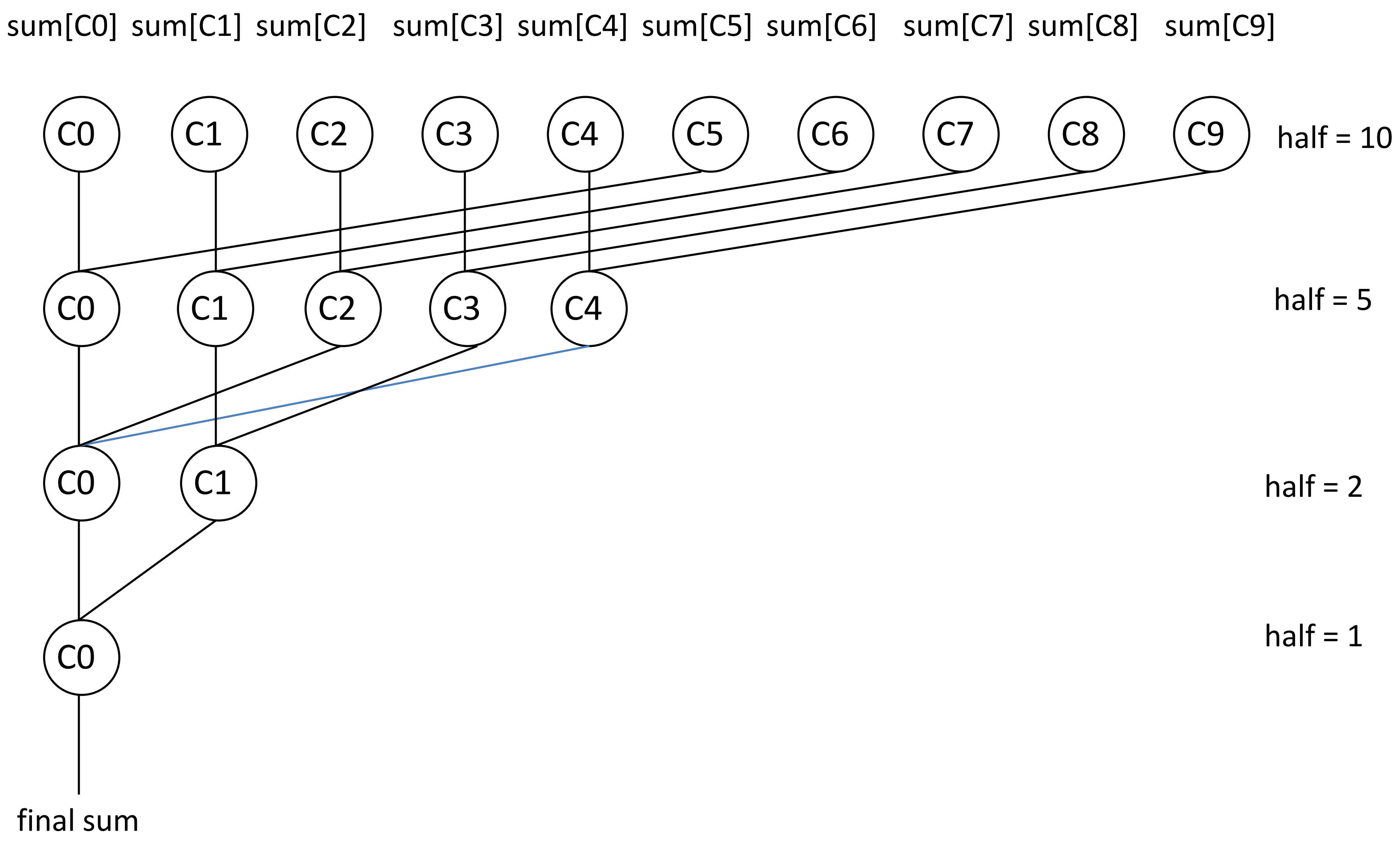
Pros and Cons of Message Passing

- **Cons:**
  - Performance
  - Harder to port a sequential program to an MPP
    - Every communication must be identified in advance
- **Pros:**
  - Easier for hardware design
  - Communication is explicit, fewer “performance surprises”
    - Message passing standard: MPI

Multiprocessor Key Questions

- Q1 – How do they share data?
- Q2 – How do they coordinate?
- Q3 – How scalable is the architecture? How many processors can be supported?

SMP Example with 10 Cores


sum[C0] sum[C1] sum[C2] sum[C3] sum[C4] sum[C5] sum[C6] sum[C7] sum[C8] sum[C9]
C0
C1
C2
C3
C4
C5
C6
C7
C8
C9
half = 10
C0
C1
C2
C3
C4
half = 5
C0
C1
half = 2
C0
half = 1
final sum

Process Synchronization

- **We need to coordinate processes working on a common task**
  - Locks (e.g., spin locks, semaphores)
    - Mutual exclusion: restrict data access to one core at a time
    - Sequential ordering: must complete the first operation before starting the second

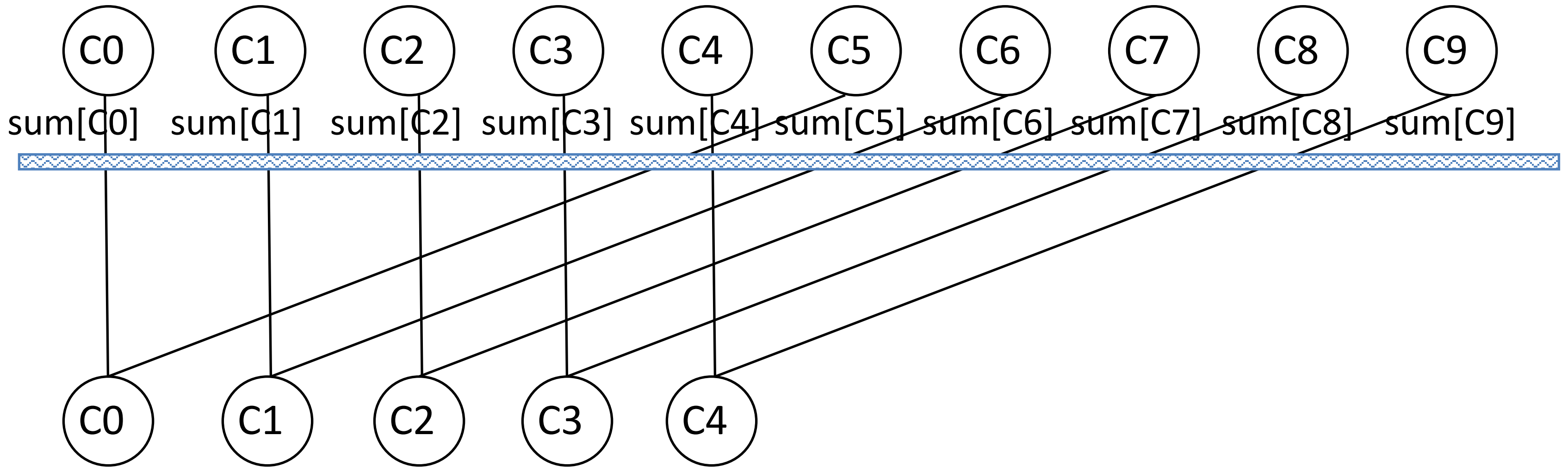
- **Need architectural support to implement the lock**

1. Arbitration mechanism
    - Single bus provides an arbitration mechanism, since the bus is the only path to memory → the core that gets the bus wins

2. Architecture-supported operation
    - Atomic swap operation (e.g., `ll` and `sc` on MIPS) allows a core to both read a location and set it to the locked state (test-and-set) in the same bus operation

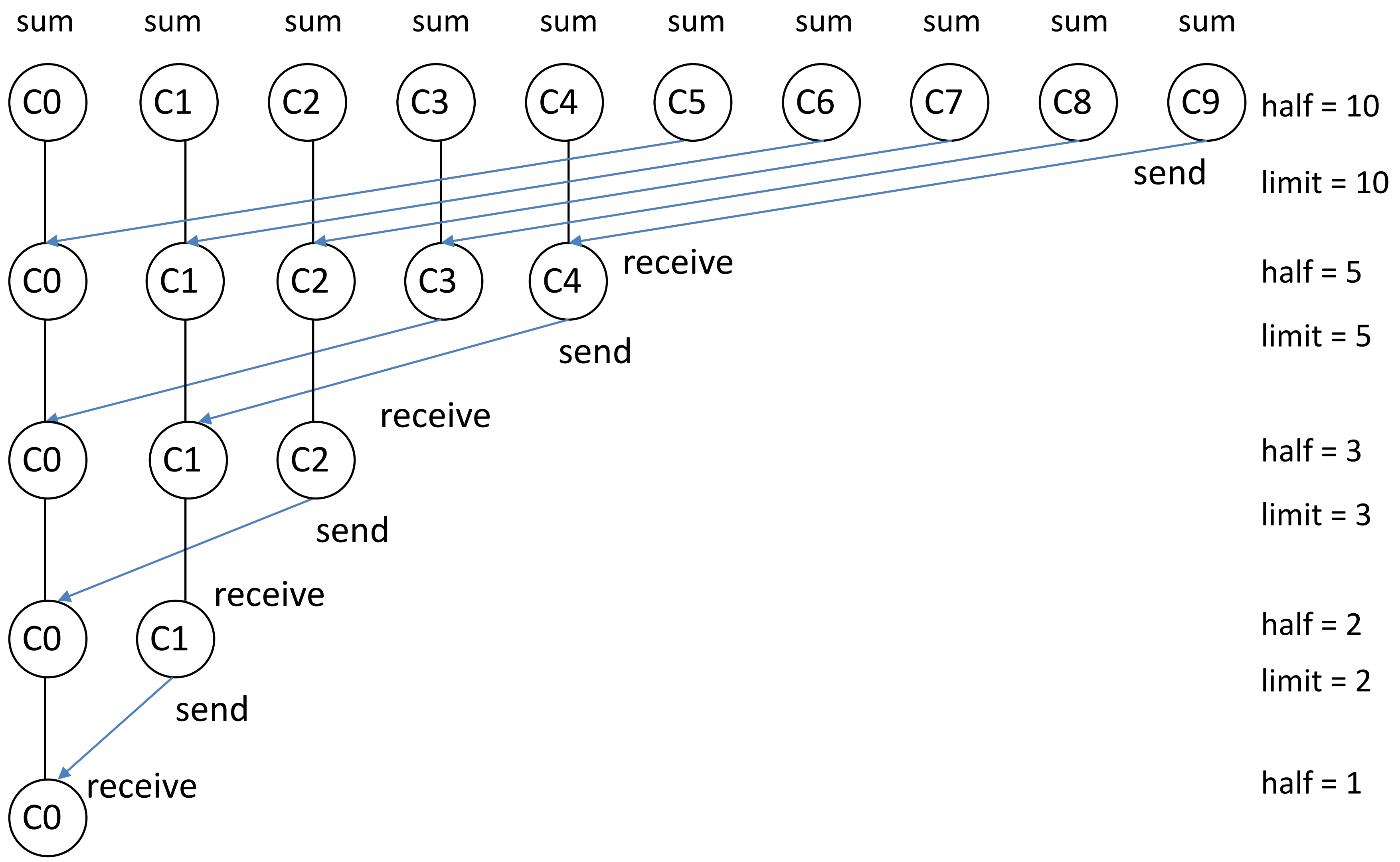
Process Synchronization

- **synch(): Cores must synchronize before the "consumer" core tries to read the results from the memory location written by the "producer" core**
  - Barrier synchronization: a synchronization scheme where cores wait at the barrier, not proceeding until every core has reached it



SJSU SAN JOSÉ STATE UNIVERSITY

MPP Example with 10 Cores



Load Balancing

- **Load balancing is important for multiprocessor performance**
  - E.g., a single core with twice the load of the others cuts the speedup almost in half

- **S = serial part, W = parallel part, N cores**
  - Time on 1 processor is S + W

- **Suppose the workload is equally distributed and there is no extra overhead to accumulate partial results**
  - Time on N processors is S + W/N

- **Suppose the workload is almost equally distributed (and still no overhead)**
  - One core does 2(W/N), one does nothing, most do W/N

  → Time is S + max(2(W/N), 0, W/N) = S + 2(W/N) = S + W/(N/2)

Multiprocessor Key Questions

- Q1 – How do they share data?
- Q2 – How do they coordinate?
- **Q3 – How scalable is the architecture? How many processors can be supported?**

			# of Cores
Communication model	Message passing		8 to 2048
	SMP	NUMA	8 to 256
		UMA	2 to 64
Physical connection	Network		8 to 256
	Bus		2 to 36

Conclusion Time

What is the difference between concurrency and parallelism?

Software implementation vs. hardware capability

What is the main difference between UMA and NUMA?

Memory access latency

SAN JOSÉ STATE UNIVERSITY *powering* SILICON VALLEY