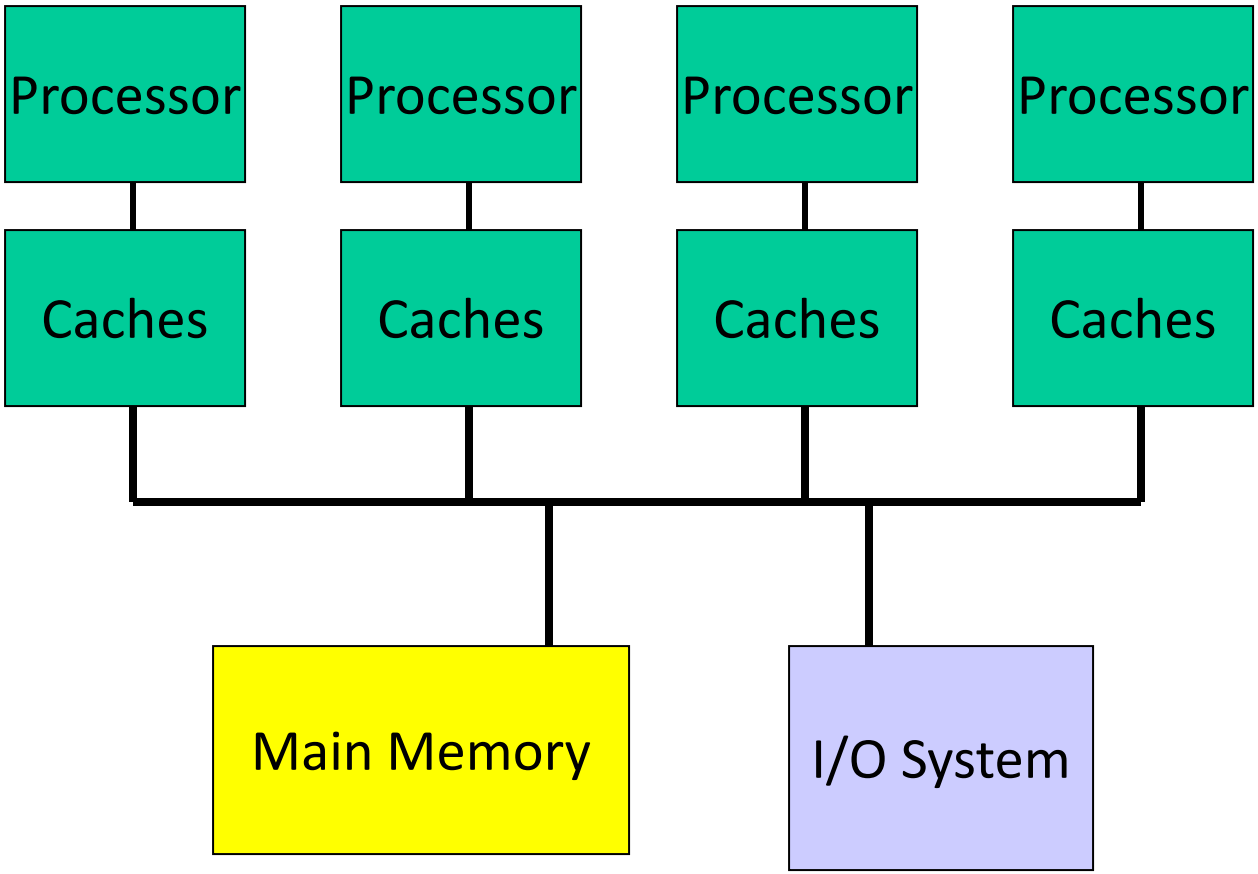
Lecture: Coherence Protocols

- Topics: multi-thread programming models,
  snooping-based protocols, directory-based protocols

Multiprocs -- Memory Organization - I

- Centralized shared-memory multiprocessor or Symmetric shared-memory multiprocessor (SMP)
- Multiple processors connected to a single centralized memory – since all processors see the same memory organization → uniform memory access (UMA)
- Shared-memory because all processors can access the entire memory address space
- Can centralized memory emerge as a bandwidth bottleneck? – not if you have large caches and employ fewer than a dozen processors

SMPs or Centralized Shared-Memory



Multiprocs -- Memory Organization - II

- For higher scalability, memory is distributed among processors → distributed memory multiprocessors
- If one processor can directly address the memory local to another processor, the address space is shared → distributed shared-memory (DSM) multiprocessor
- If memories are strictly local, we need messages to communicate data → cluster of computers or multicomputers
- Non-uniform memory architecture (NUMA) since local memory has lower latency than remote memory

SMPs

- Centralized main memory and many caches → many copies of the same data

- A system is cache coherent if a read returns the most recently written value for that word

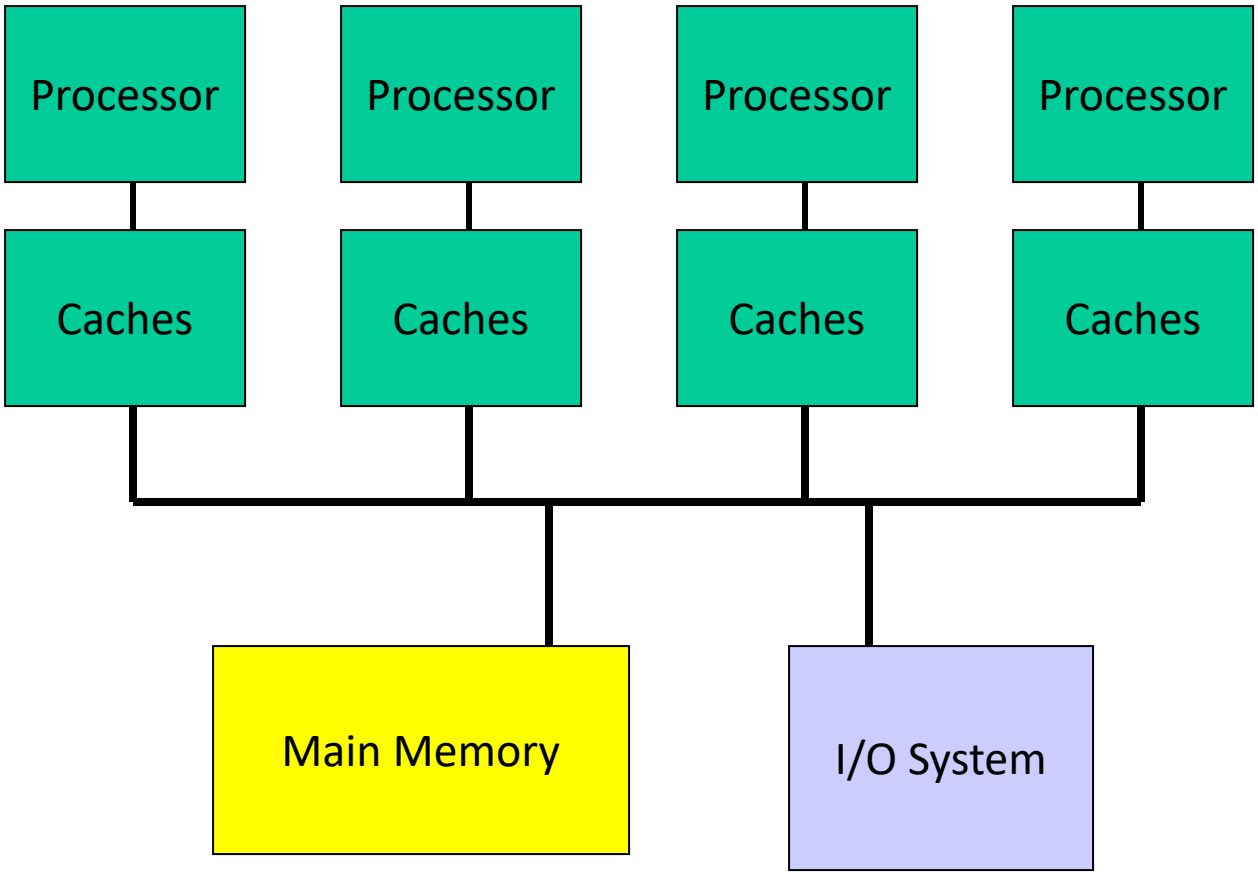
Time	Event	Value of X in	Cache-A	Cache-B	Memory
0			-	-	1
1	CPU-A reads X		1	-	1
2	CPU-B reads X		1	1	1
3	CPU-A stores 0 in X		0	1	0

Cache Coherence

A memory system is coherent if:

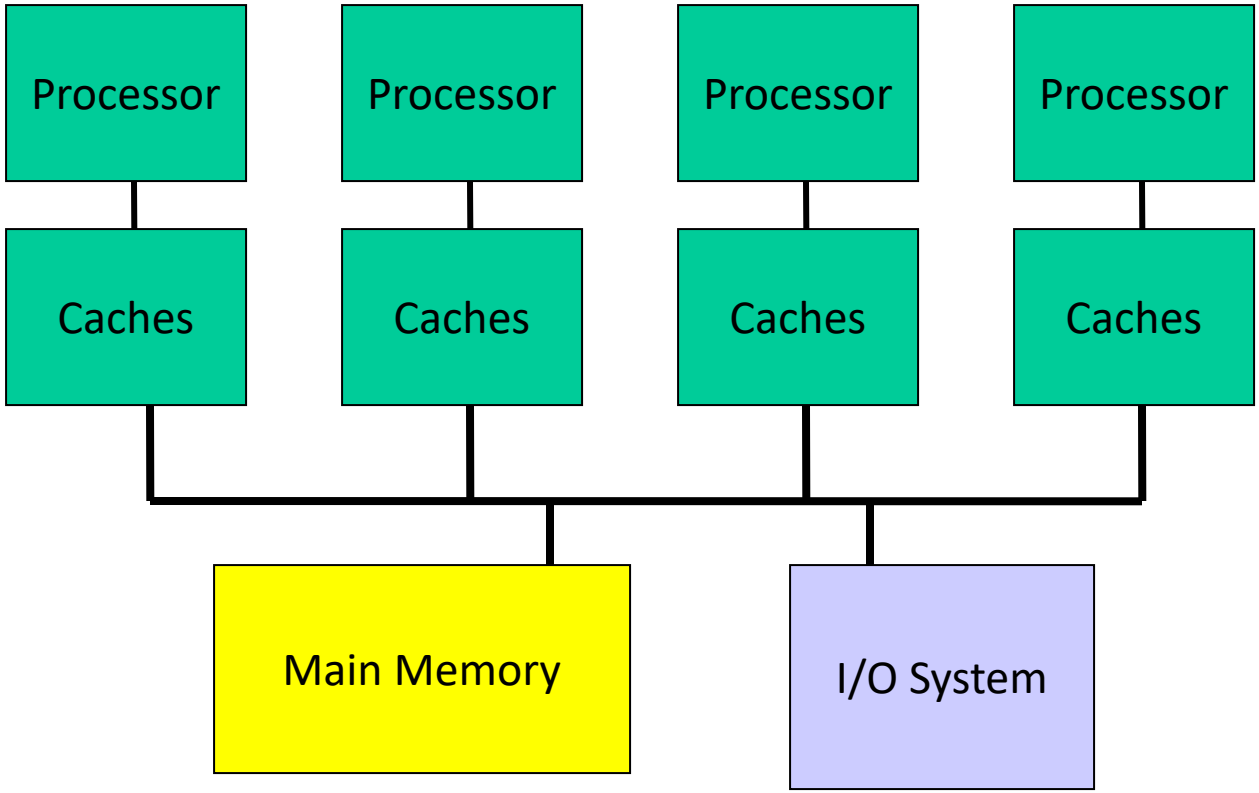
- Write propagation: P1 writes to X, sufficient time elapses, P2 reads X and gets the value written by P1
- Write serialization: Two writes to the same location by two processors are seen in the same order by all processors
- The memory *consistency* model defines "time elapsed" before the effect of a processor is seen by others and the ordering with R/W to other locations (loosely speaking – more later)

SMPs or Centralized Shared-Memory

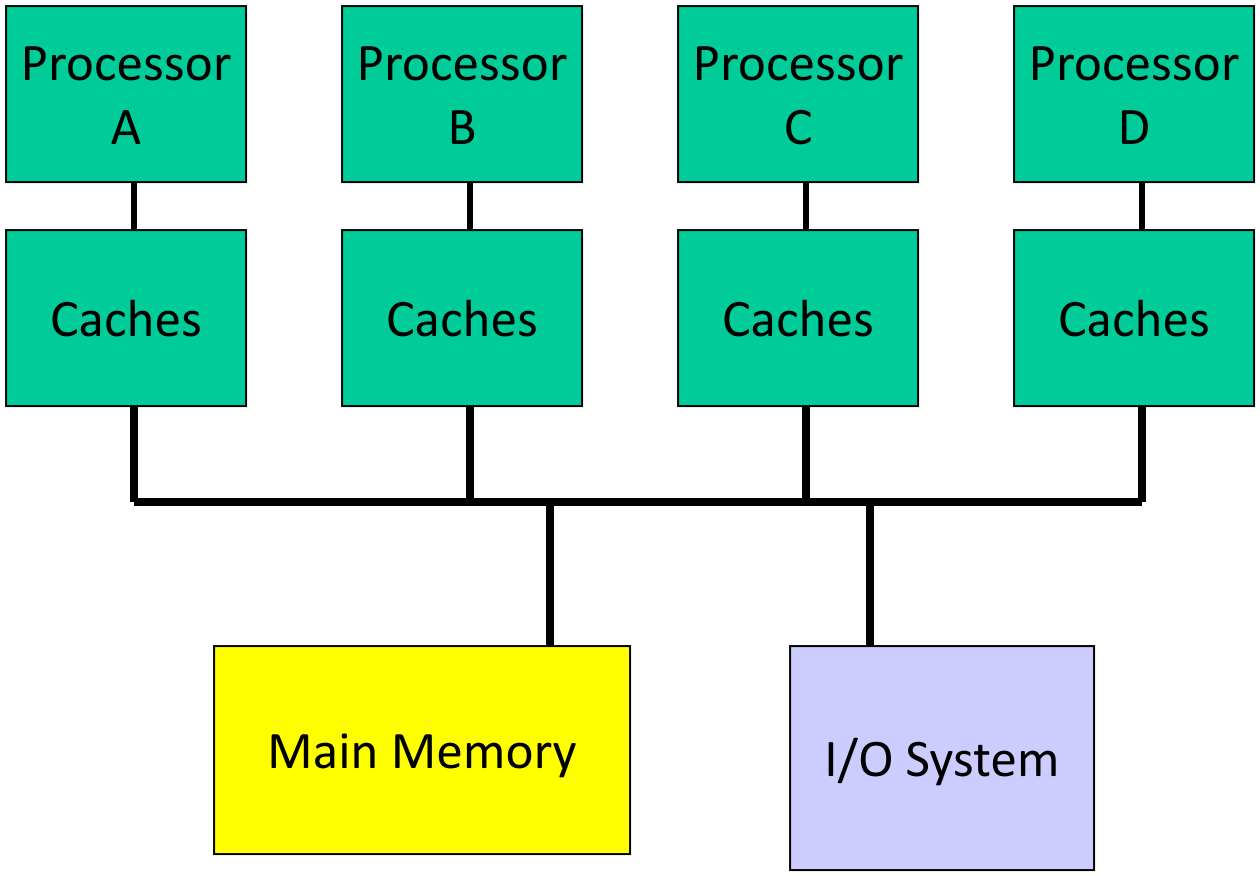


Design Issues

- Invalidate
- Find data
- Writeback / writethrough
- Cache block states
- Contention for tags
- Enforcing write serialization



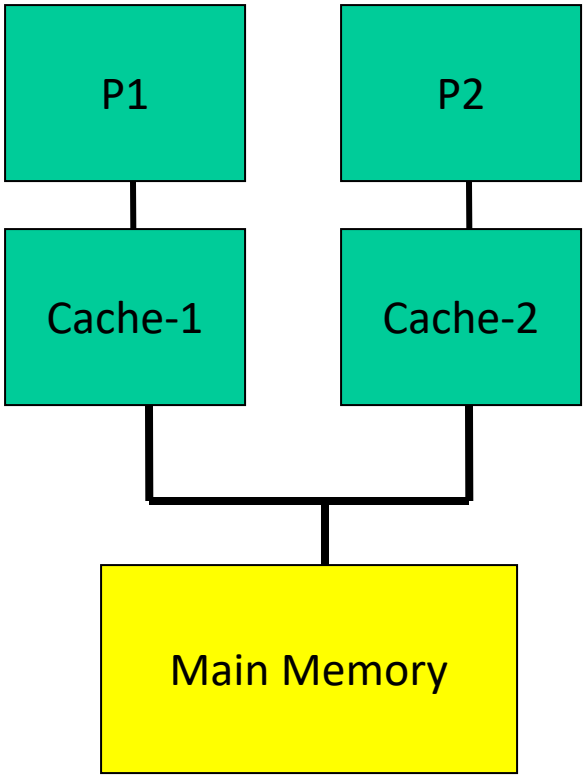
SMP Example



A: Rd X
B: Rd X
C: Rd X
A: Wr X
A: Wr X
C: Wr X
B: Rd X
A: Rd X
A: Rd Y
B: Wr X
B: Rd Y
B: Wr X
B: Wr Y

Example

- P1 reads X: not found in cache-1, request sent on bus, memory responds, X is placed in cache-1 in shared state
- P2 reads X: not found in cache-2, request sent on bus, everyone snoops this request, cache-1 does nothing because this is just a read request, memory responds, X is placed in cache-2 in shared state



- P1 writes X: cache-1 has data in shared state (shared only provides read perms), request sent on bus, cache-2 snoops and then invalidates its copy of X, cache-1 moves its state to modified
- P2 reads X: cache-2 has data in invalid state, request sent on bus, cache-1 snoops and realizes it has the only valid copy, so it downgrades itself to shared state and responds with data, X is placed in cache-2 in shared state, memory is also updated

Example

Request	Cache Hit/Miss	Request on the bus	Who responds	State in Cache 1	State in Cache 2	State in Cache 3	State in Cache 4
				Inv	Inv	Inv	Inv
P1: Rd X	Miss	Rd X	Memory	S	Inv	Inv	Inv
P2: Rd X	Miss	Rd X	Memory	S	S	Inv	Inv
P2: Wr X	Perms Miss	Upgrade X	No response. Other caches invalidate.	Inv	M	Inv	Inv
P3: Wr X	Write Miss	Wr X	P2 responds	Inv	Inv	M	Inv
P3: Rd X	Read Hit	-	-	Inv	Inv	M	Inv
P4: Rd X	Read Miss	Rd X	P3 responds. Mem wrtbk	Inv	Inv	S	S

Cache Coherence Protocols

- Directory-based: A single location (directory) keeps track of the sharing status of a block of memory

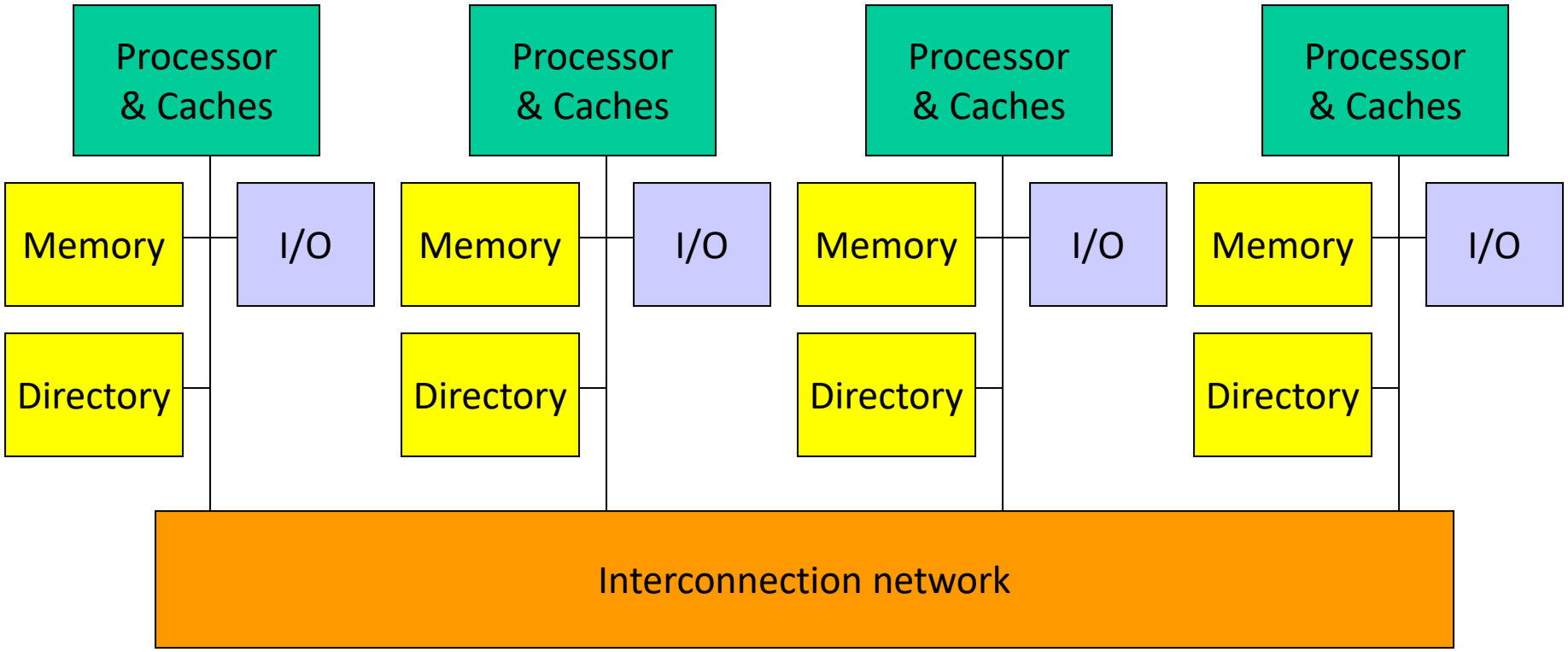
- Snooping: Every cache block is accompanied by the sharing status of that block – all cache controllers monitor the shared bus so they can update the sharing status of the block, if necessary

  - Write-invalidate: a processor gains exclusive access of a block before writing by invalidating all other copies
  - Write-update: when a processor writes, it updates other shared copies of that block

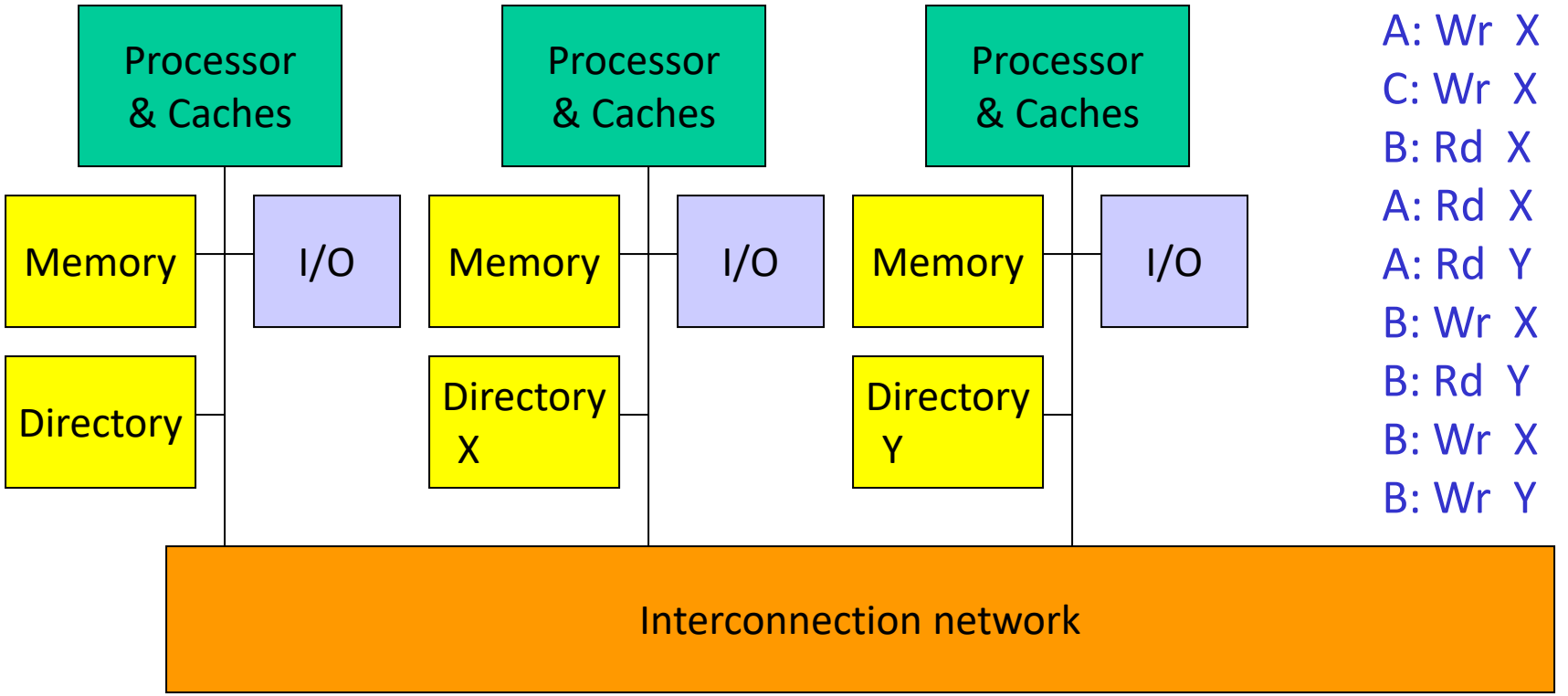
Directory-Based Cache Coherence

- The physical memory is distributed among all processors
- The directory is also distributed along with the corresponding memory
- The physical address is enough to determine the location of memory
- The (many) processing nodes are connected with a scalable interconnect (not a bus) – hence, messages are no longer broadcast, but routed from sender to receiver – since the processing nodes can no longer snoop, the directory keeps track of sharing state

Distributed Memory Multiprocessors



Directory-Based Example



A: Rd X
B: Rd X
C: Rd X
A: Wr X
A: Wr X
C: Wr X
B: Rd X
A: Rd X
A: Rd Y
B: Wr X
B: Rd Y
B: Wr X
B: Wr Y

Example

Request	Cache Hit/Miss	Messages	Dir State	State in C1	State in C2	State in C3	State in C4
				Inv	Inv	Inv	Inv
P1: Rd X	Miss	Rd-req to Dir. Dir responds.	X: S: 1	S	Inv	Inv	Inv
P2: Rd X	Miss	Rd-req to Dir. Dir responds.	X: S: 1, 2	S	S	Inv	Inv
P2: Wr X	Perms Miss	Upgr-req to Dir. Dir sends INV to P1. P1 sends ACK to Dir. Dir grants perms to P2.	X: M: 2	Inv	M	Inv	Inv
P3: Wr X	Write Miss	Wr-req to Dir. Dir fwds request to P2. P2 sends data to Dir. Dir sends data to P3.	X: M: 3	Inv	Inv	M	Inv
P3: Rd X	Read Hit	-	-	Inv	Inv	M	Inv
P4: Rd X	Read Miss	Rd-req to Dir. Dir fwds request to P3. P3 sends data to Dir. Memory wrtbk. Dir sends data to P4.	X: S: 3, 4	Inv	Inv	S	S

Cache Block States

- What are the different states a block of memory can have within the directory?

- Note that we need information for each cache so that invalidate messages can be sent

- The block state is also stored in the cache for efficiency

- The directory now serves as the arbitrator: if multiple write attempts happen simultaneously, the directory determines the ordering