RAIDs & Cache Coherence

'22H2

송 인 식

Outline

- RAIDs
- Cache Coherence

What is RAID?

- Redundant Array of Independent (Inexpensive) Disks
- A set of disk stations treated as one logical station
- Data are distributed over the stations
- Redundant capacity is used for parity allowing for data repair

Levels of RAID

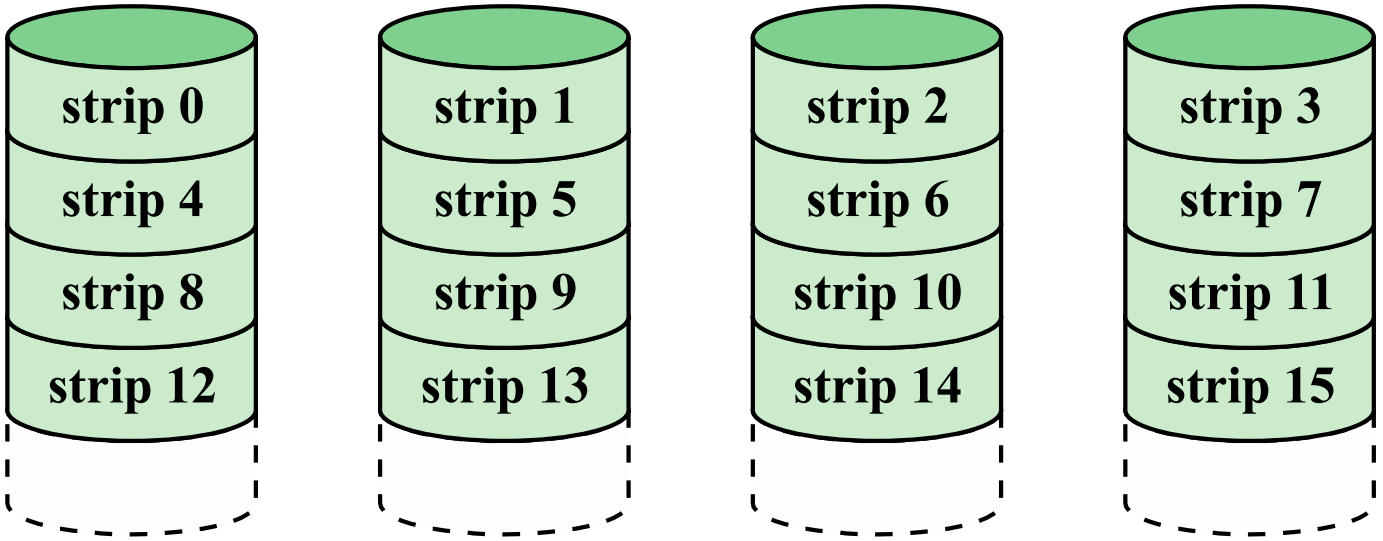
Category	Level	Description	Disks Required	Data Availability	Large I/O Data Transfer Capacity	Small I/O Request Rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

N = number of data disks; m proportional to log N

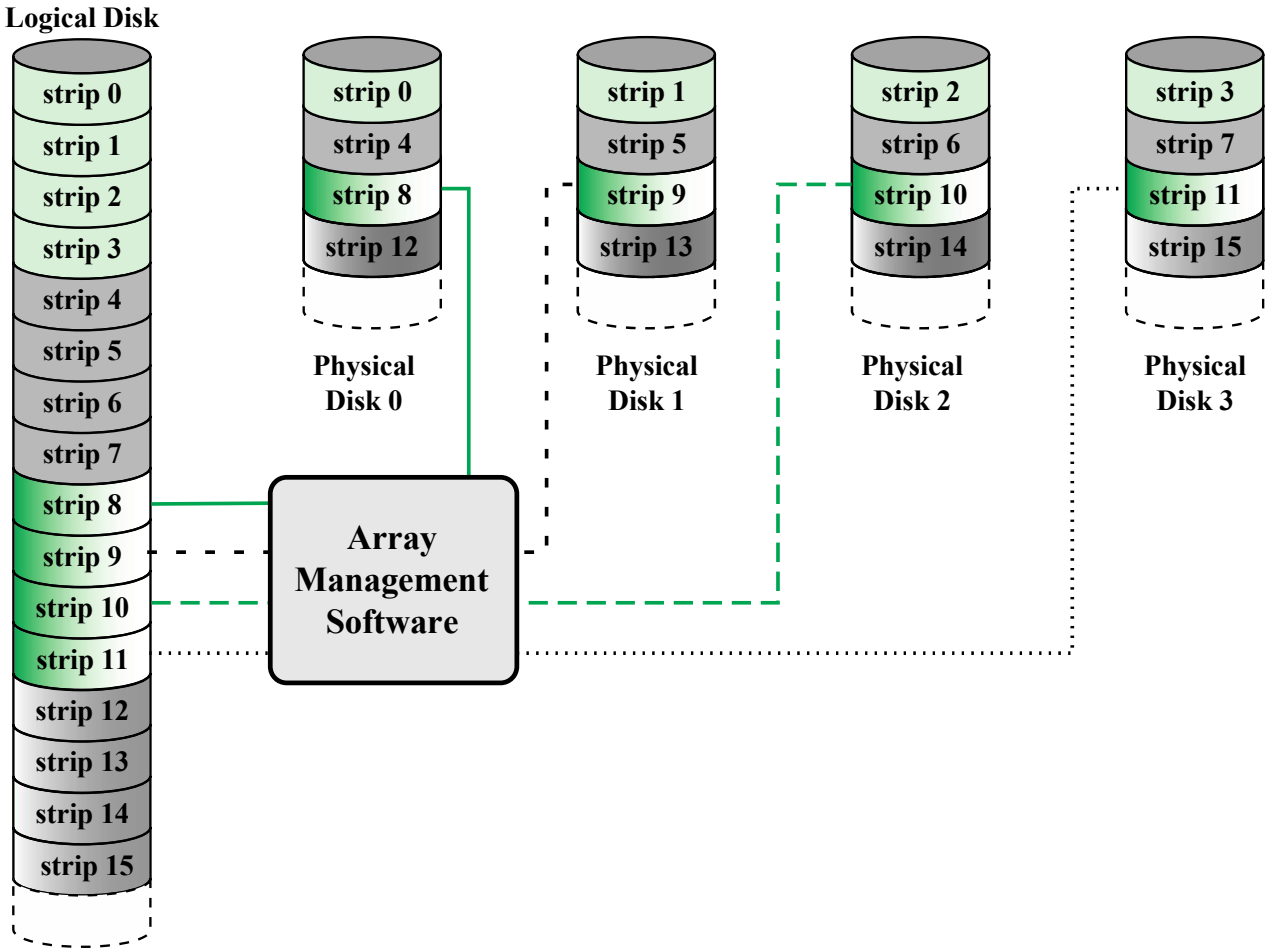
RAID 0

- All data (user and system) are distributed over the disks so that there is a reasonable chance for parallelism
- Disk is logically a set of strips (blocks, sectors,…). Strips are numbered and assigned consecutively to the disks (see picture.)

Raid 0 (No redundancy)



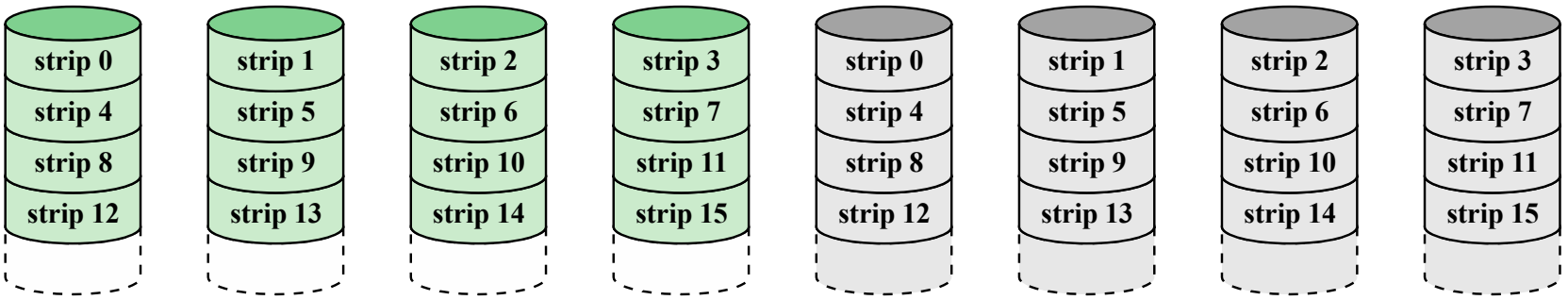
Data mapping Level 0



RAID 0

- Performance depends highly on the the request patterns
- High data transfer rates are reached if
  - Integral data path is fast (internal controllers, I/O bus of host system, I/O adapters and host memory busses)
  - Application generates efficient usage of the disk array by requests that span many consecutive strips
- If response time is important (transactions) more I/O requests can be handled in parallel

Raid 1 (mirrored)



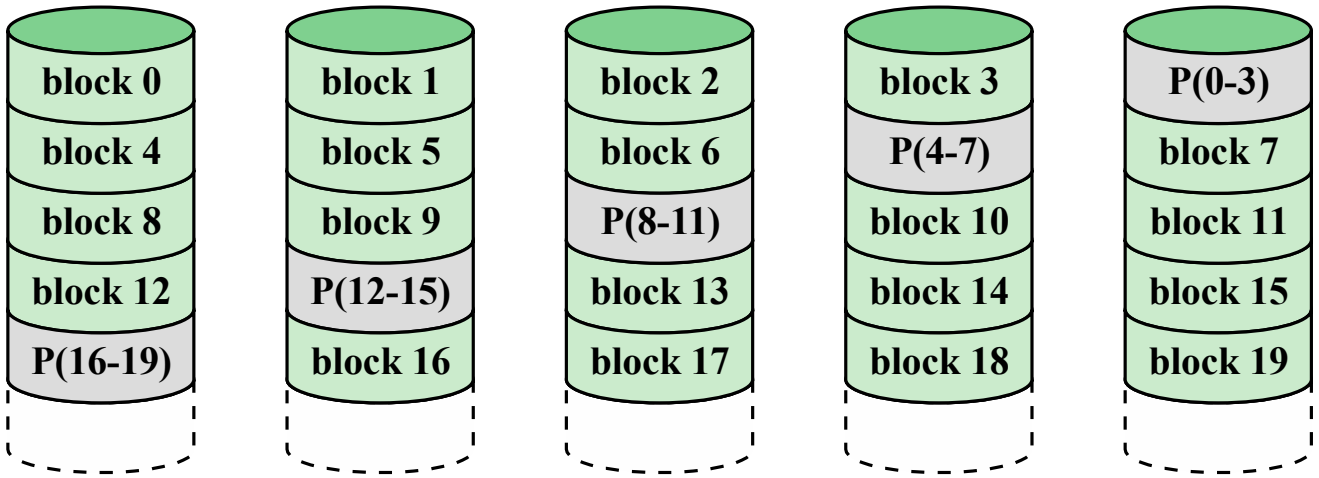
Raid 1

- RAID 1 does not use parity, it simply mirrors the data to obtain reliability
- Plus:
  - Reading request can be served by any of the two disks containing the requested data (minimum search time)
  - Writing request can be performed in parallel to the two disks: no "writing penalty"
  - Recovery from error is easy, just copy the data from the correct disk

Raid 1

- Minus:
  - Price for disks is doubled
  - Will only be used for system critical data that must be available at all times
- RAID 1 can reach high transfer rates and fast response times (~2*RAID 0) if most of the requests are reading requests. In case most requests are writing requests, RAID 1 is not much faster than RAID 0.

RAID 5 (block-level distributed parity)



RAID 5

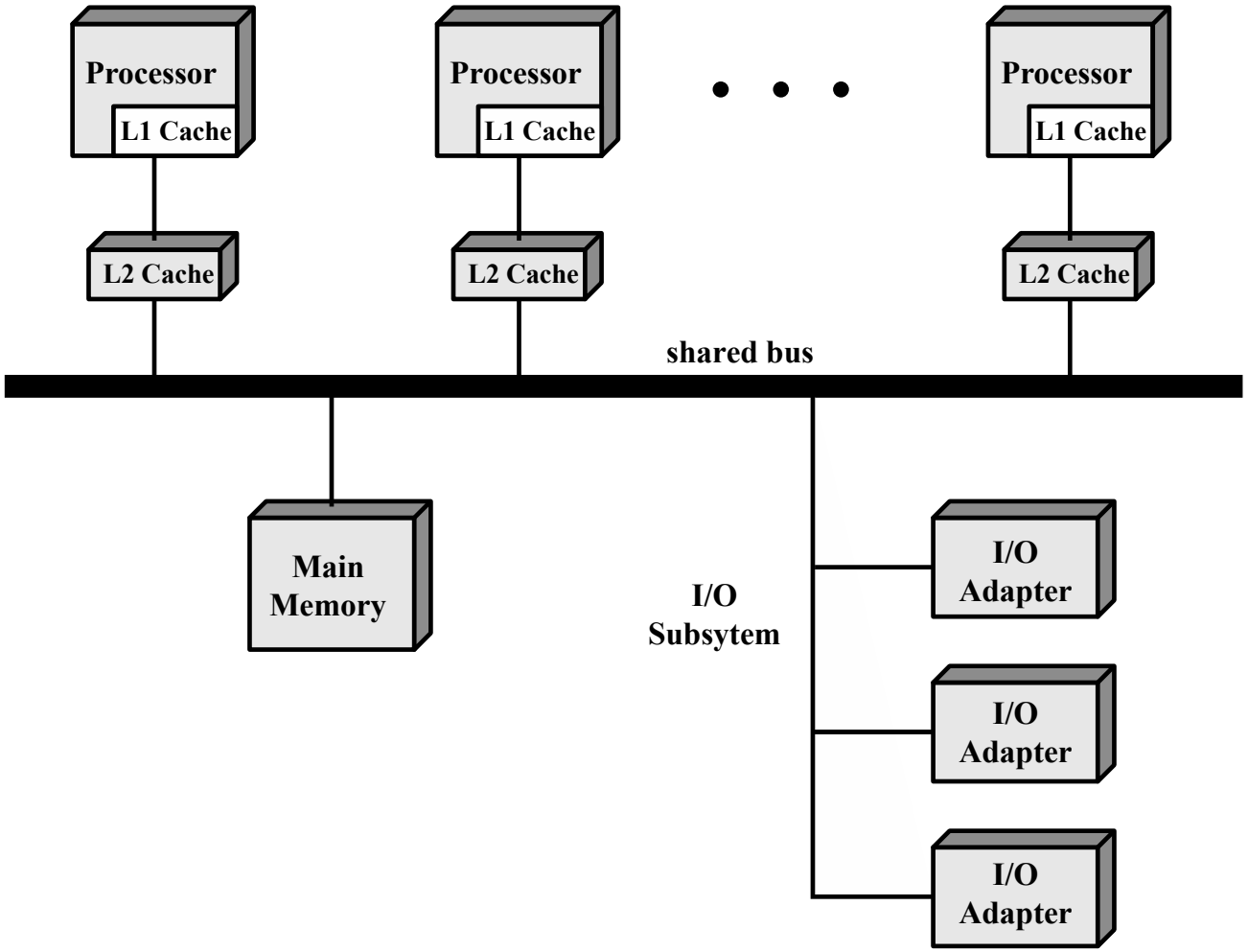
- Distribution of the parity strip to avoid the bottle neck.
- Can use round robin:

Parity disk = (-block number/4) mod 5

Outline

- RAIDs
- Cache Coherence

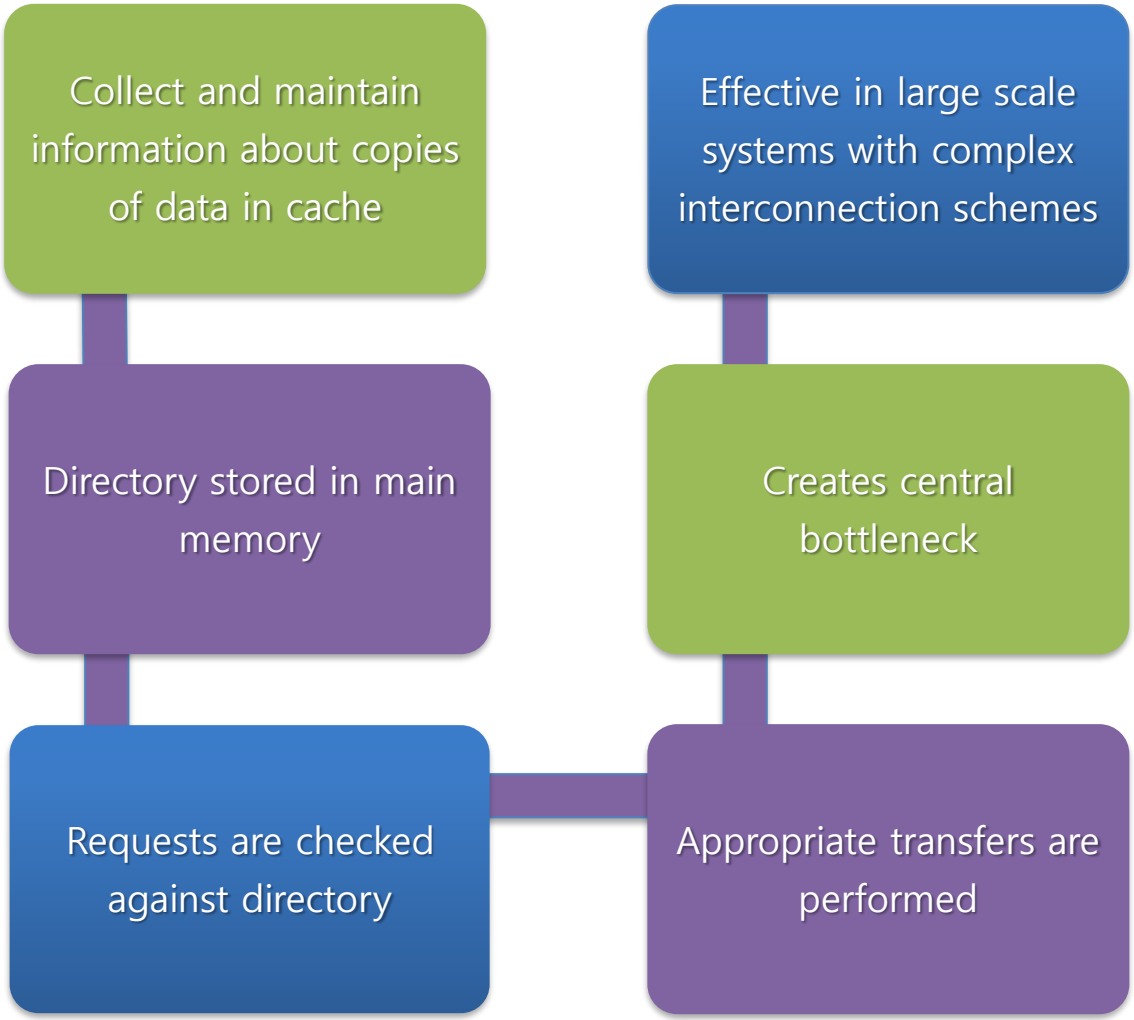
Symmetric Multiprocessor Organization



Cache Coherence Protocols

- Directory protocols
- Snoopy protocols

Directory Protocols



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

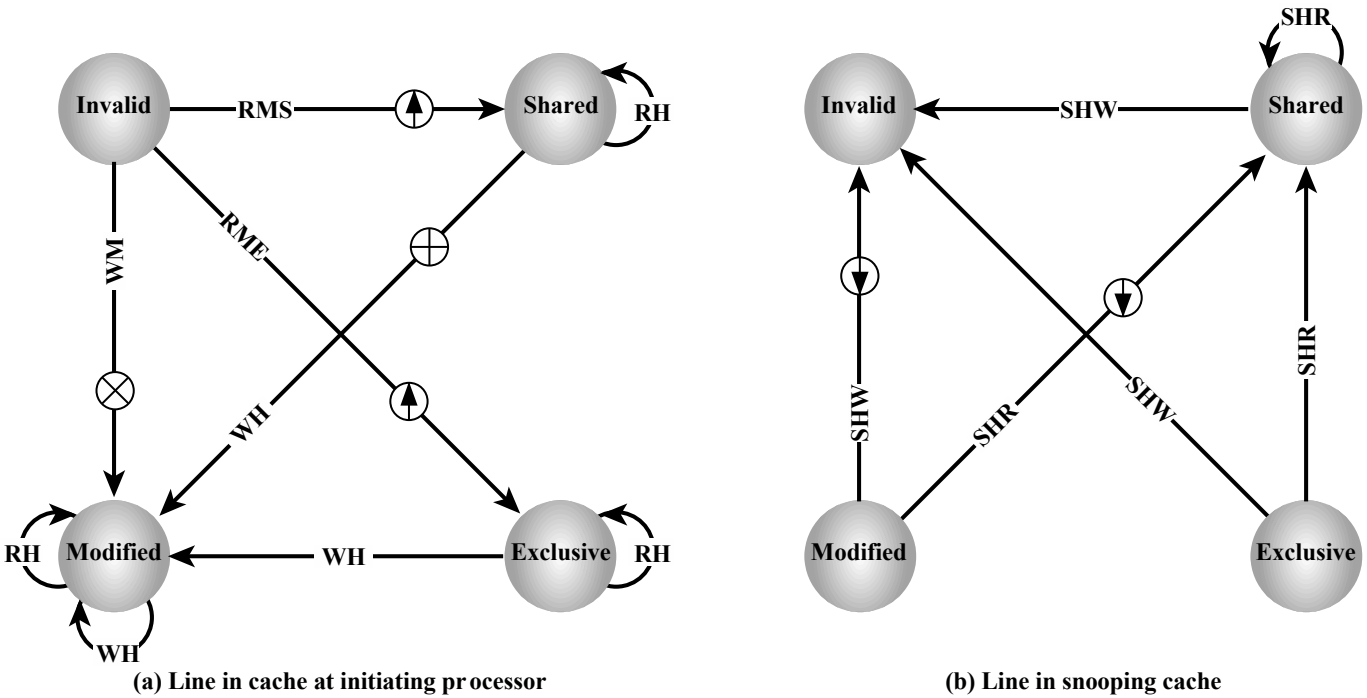
MESI Protocol

- 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

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

MESI State Transition Diagram



Questions?