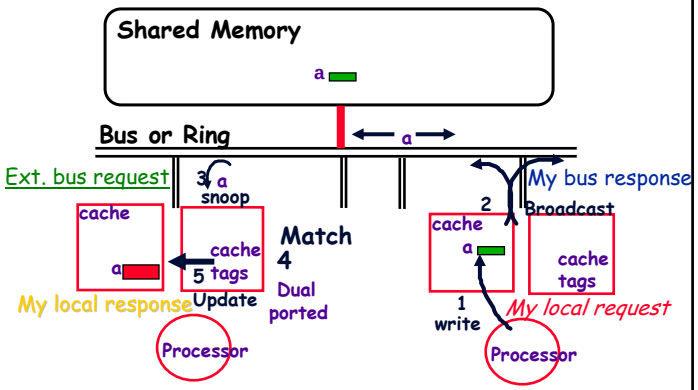
Cache Coherence in Manycores

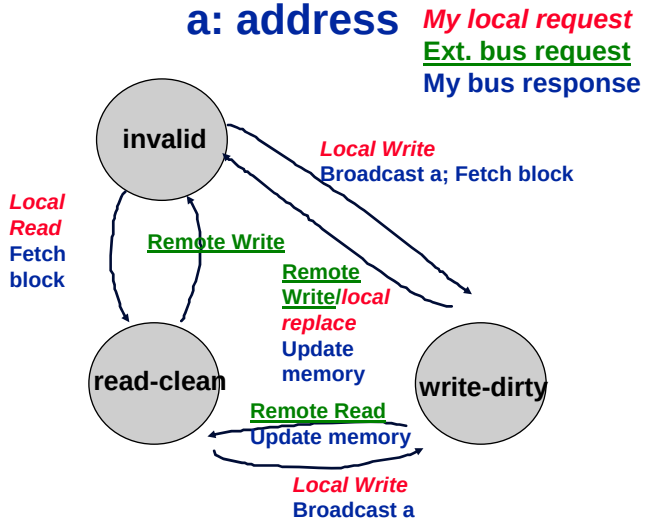
6.173
Fall 2010
Agarwal

Snooping Caches Definitions



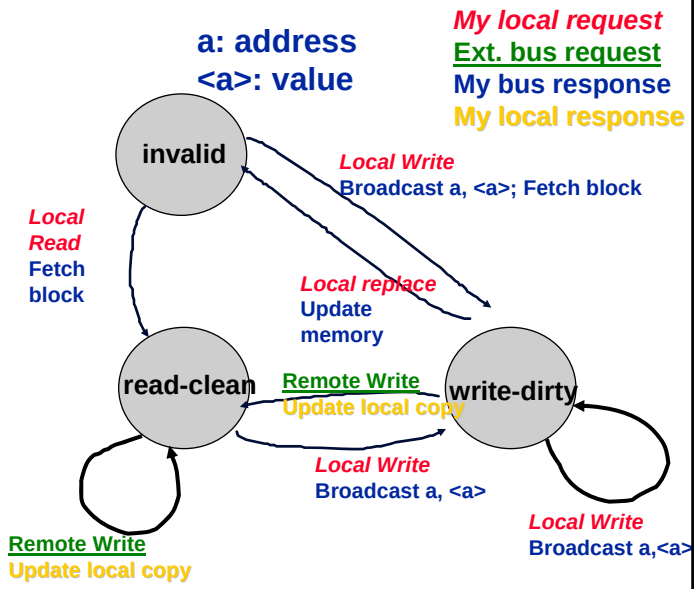
Remember, "a" is memory word

State diagram for cache block in *ownership* protocols

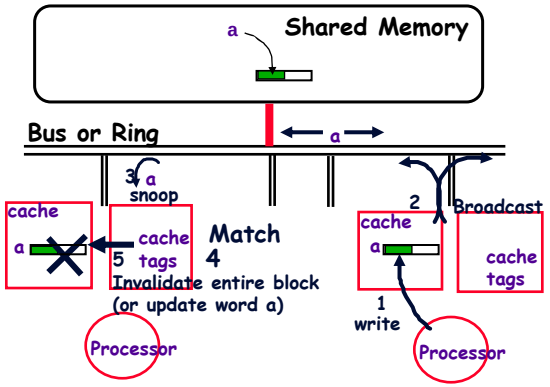


In ownership protocol:
writer owns exclusive copy

State diagram for *update* protocols



What to Do When Cache Blocksize > 1 word



- Suppose "a" is word address
- State associated with entire cache block

Discuss: Effect of false sharing

Solving the Coherence Problem

- Small multicores
  - Software coherence
  - Snooping caches
- Manycores
  - Software coherence
  - full map directories
  - limited pointers
  - chained pointers
    - singly linked
    - doubly linked
  - limitless schemes
  - Hierarchical methods

Today

We will study

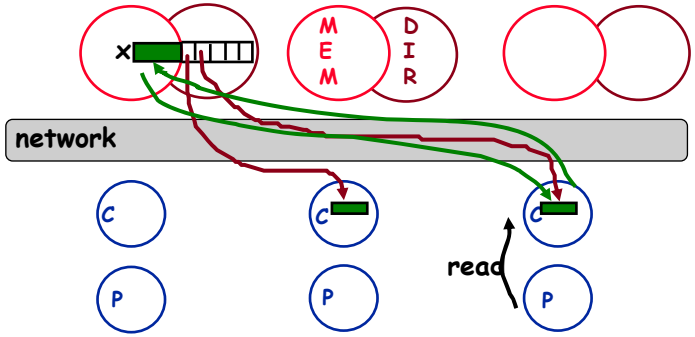
Coherence structures

Coherence protocols

Cache side state diagrams

Directory side state diagrams

Directory Coherence Schemes
First Read

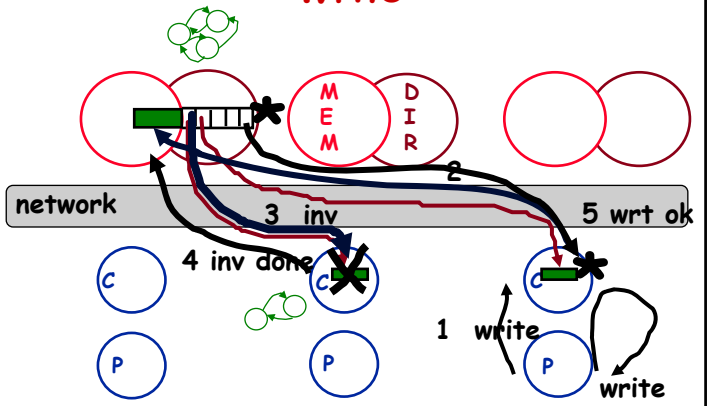


General directories:

- On read, track cached copy in directory
- Distribute directories with MEMs

  Directory bandwidth scales in proportion to memory bandwidth
- Most directory accesses during memory access -- so not too many extra network requests (except, write to read VAR)
- Manycores?

Full-Map Directory
Write



General directories:

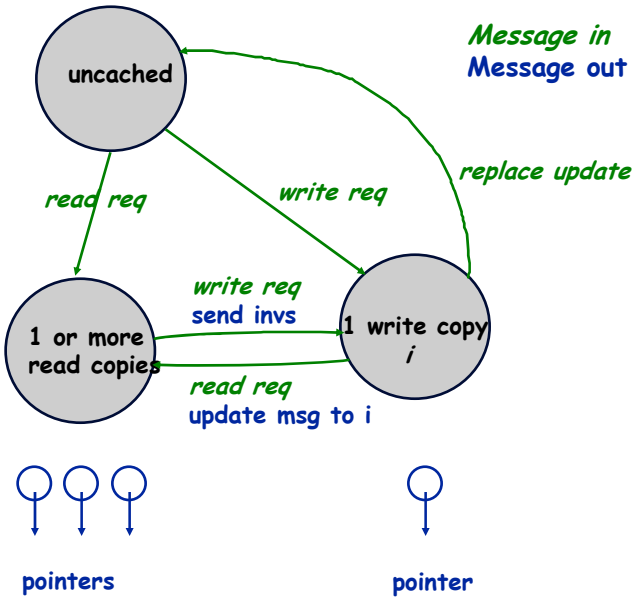
- On write, check directory
- If shared, send inv msgs
- Full-map directory: Dir$_n$NB

NB
(no broadcast)

#pointers

Needs state machine at directory and at cache

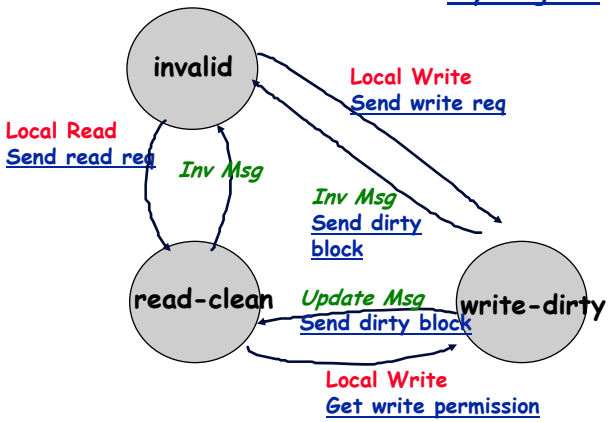
Directory controller state diagram for memory block



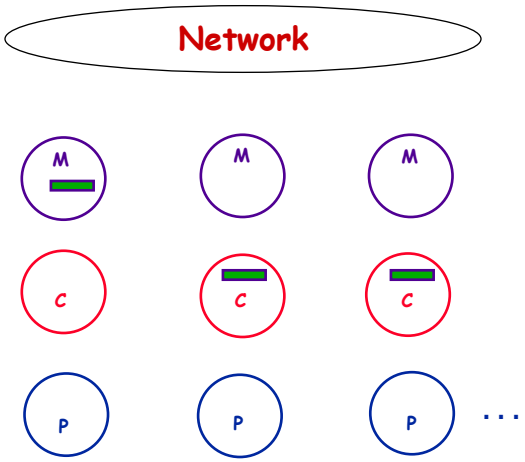
Cache state machine for directory scheme

a

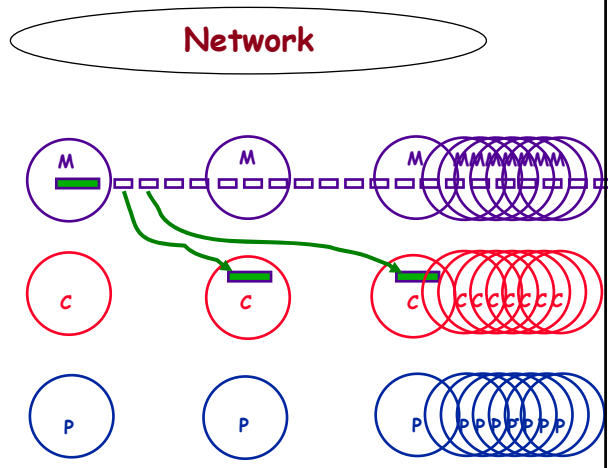
My local request
My Msg in
My Msg out



Also, need to ack (implicitly or explicitly) every message from memory – we'll see why later.



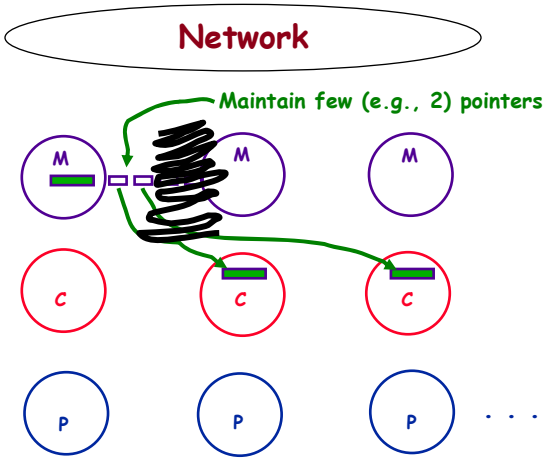
Next, let's look at several practical and scalable implementations of directory schemes



Full-map directories do not scale

Dir$_n$NB

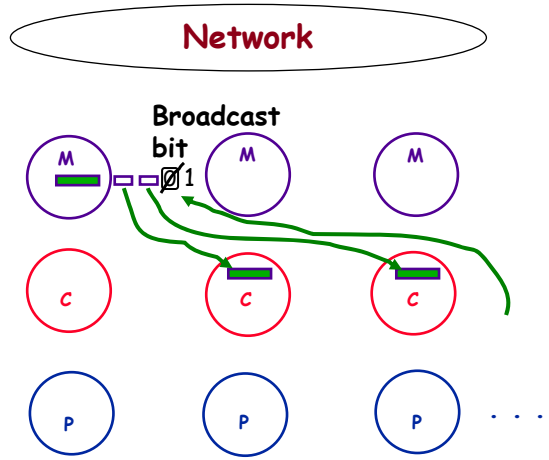
can get large!



Limited directories: Exploit *worker set* behavior

- Rarely more than i processors share (i is small)
- Invalidate 1 if (i+1)th processor comes along: **Dir_iNB**
- Insight: The set of 2 pointers can be managed like a fully-associative 2 entry cache on the virtual space of all pointers

But what do you do about widely shared data?



Limited directories: maintain i pointers (i < n), where n is number of cores

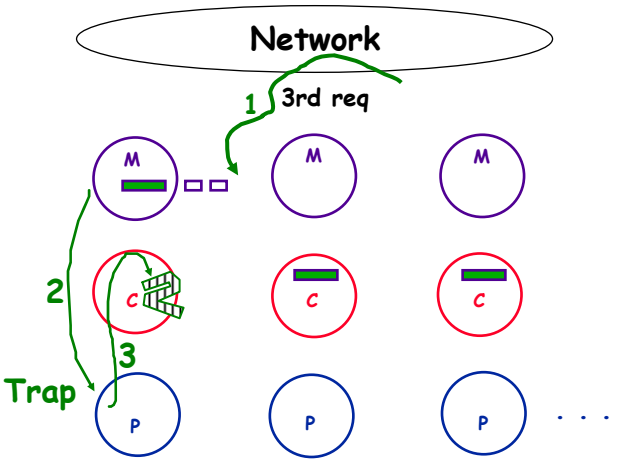
- What to do about widely shared data
- Invalidate 1 if (i+1)th processor comes along: **Dir_iNB** and live with it

Set a broadcast invalidate bit and let more than i processors share: **Dir_iB**

What to do on a write?

If B set, send broadcast invalidate

What is Dir_0B?



Limitless directories:

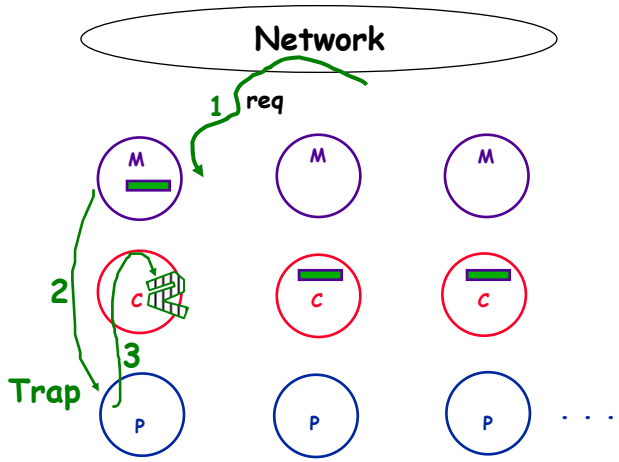
Limited directories Locally Extended through Software Support

- Trap processor when 3rd request comes
- Processor extends directory into memory (local cache)

Dir? → Dir_nNB

Why might the performance be OK?

Any unique software opportunities here?



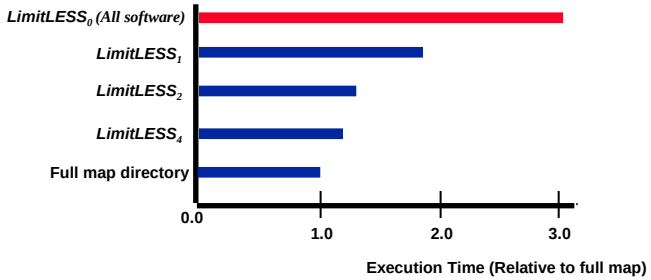
A crazy idea…. zero pointer limitless aka, all software directories:

- Trap processor, and let processor do usual directory operations in software
- Processor extends directory into memory (local cache)
- Zero hardware!

Software can even implement protocol best suited to each variable

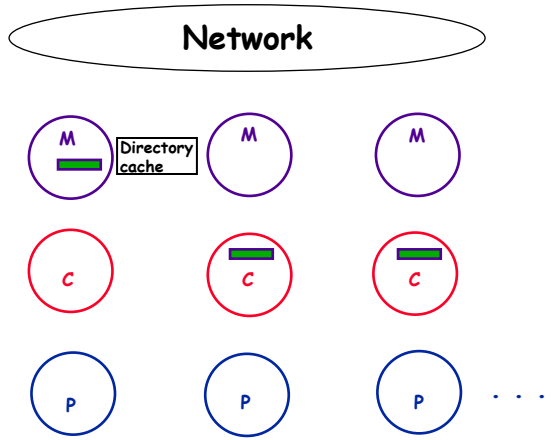
Or it can change protocols on the fly (using, for example, competitive algorithms)

Performance



Weather modeling

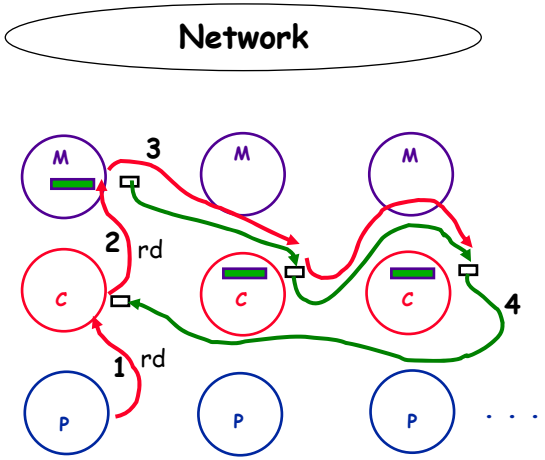
Some Variants



Directory caches:

Cache a few directory entries

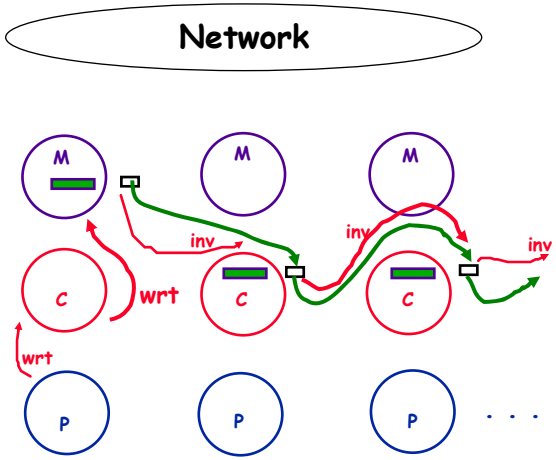
No need to maintain directory entry for uncached blocks, or for private data (compiler/OS can tag private pages)



Chained directories: Simply different data structure for directory

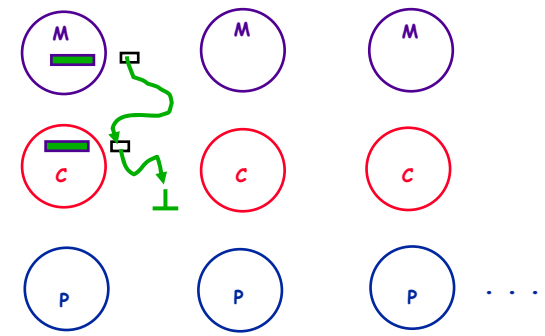
- Link all cache entries
- On read, add new entry to end of list

What to do on write?



Chained directories: What to do on a write?

- Invalidate all entries on write

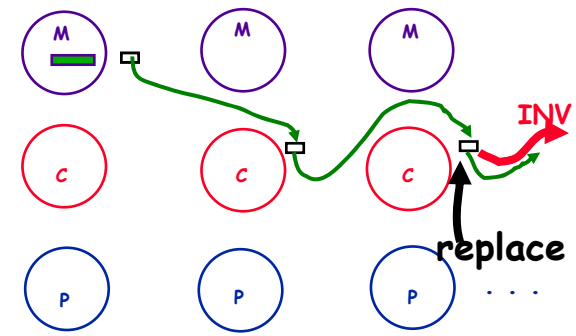


Chained directories: What to do on a write?

- Invalidate all entries on write
- Only written block added to new list

Problems?

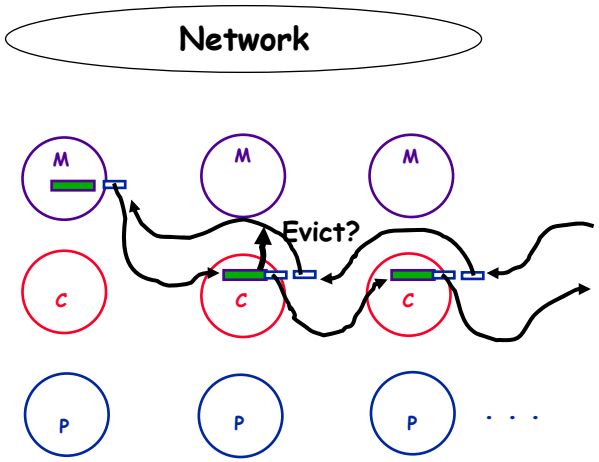
But longer latencies
Also more complex… why?



Chained directories:

- Must handle replacements of elements in chain due to misses!

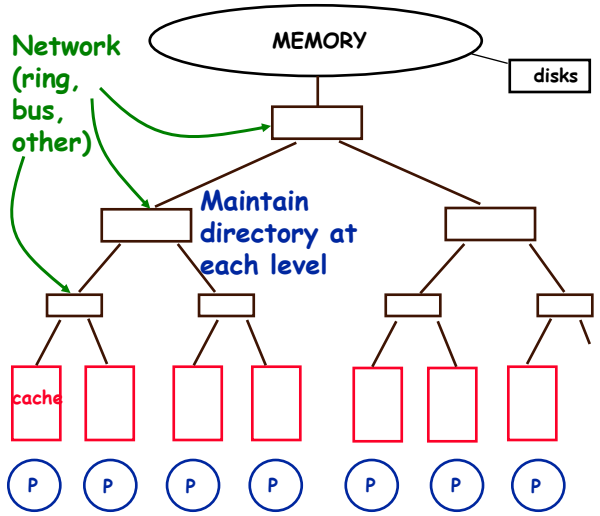
Invalidate rest



Another idea…

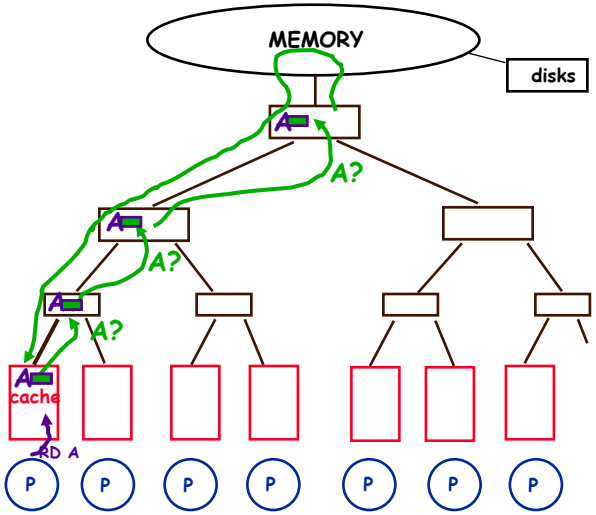
Doubly linked chains

Even more complex



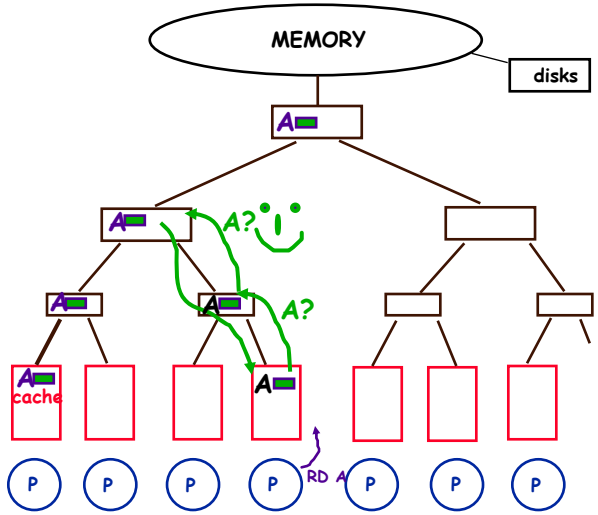
Hierarchical Protocols - E.g. KSR, Kendall Square Research Inc. (actually had rings…)

Multicores will probably initially scale this way as we look to do coherence across two chips



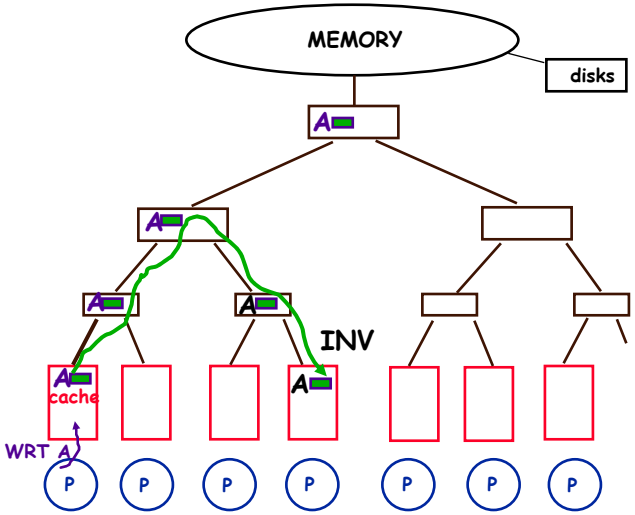
Hierarchical

First read



Hierarchical

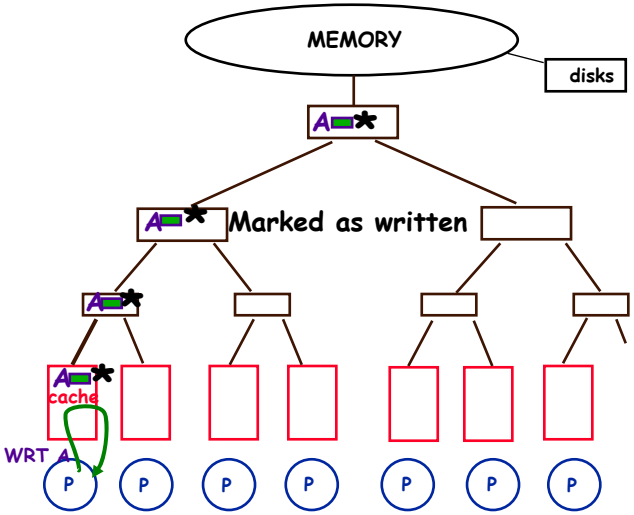
Subsequent read



Hierarchical

Write

- 27 -



Hierarchical

Subsequent writes

- 28 -