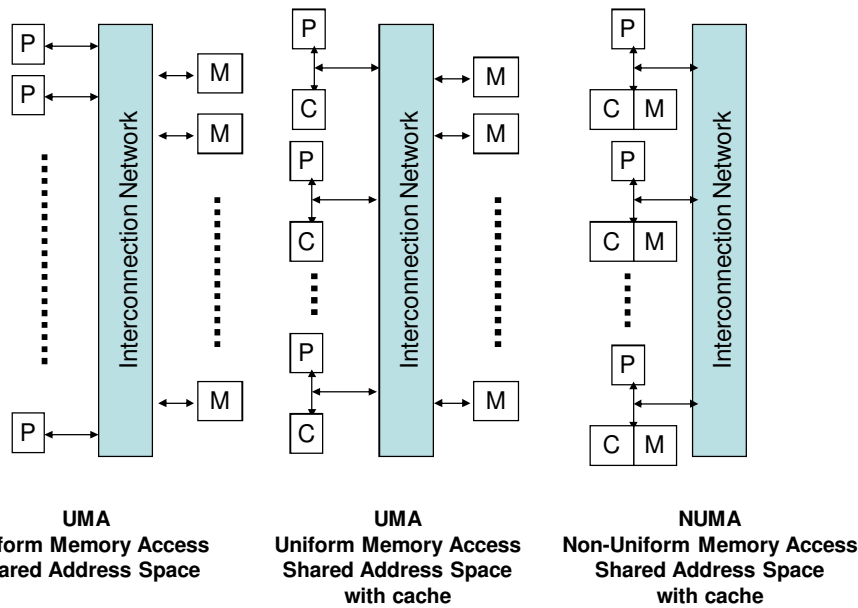


Shared Address Space Computing: Hardware Issues

Alistair Rendell

See Chapter 2 of Lin and Synder,
Chapter 2 of Grama, Gupta, Karypis and Kumar,
and also "*Computer Architecture: A Quantitative
Approach*", J.L. Hennessy and D.A. Patterson, Morgan
Kaufmann



Grama fig 2.5

Shared Address Space Systems

- Systems with caches but otherwise flat memory generally called UMA
- If access to local cheaper than remote (NUMA), this should be built into your algorithm
 - How to do this and O/S support is another matter
 - (man numa on linux will give details of numa support)
- Global address space easier to program
 - Read only interactions invisible to programmer and coded like sequential program
 - Read/write are harder, require mutual exclusion for concurrent accesses
- Programmed using threads and directives
 - We will consider pthreads and OpenMP
- Synchronization using locks and related mechanisms

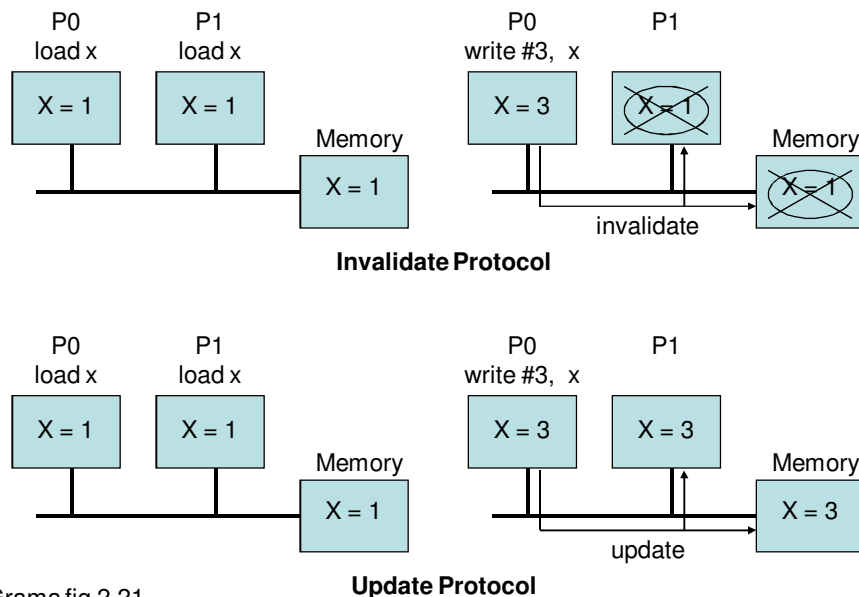
Caches on Multiprocessors

- Multiple copies of some data word being manipulated by two or more processors at the same time
- Two requirements
 - Address translation mechanism that locates memory word in system
 - Concurrent operations on multiple copies have well defined semantics
- Latter generally known as **cache coherency protocol**
 - (I/O using DMA on machines with caches also leads to coherency issues)
- Some machines only provide shared address space mechanisms and leave coherence to programme
 - Cray T3D provided get and put and cache invalidate

Shared-Address-Space and Shared-Memory Computers

- Shared-memory historically used for architectures in which memory is physically shared among various processors, and all processors have equal access to any memory segment
 - This is identical to the UMA model
- Compared to distributed-memory computer where different memory segments are physically associated with different processing elements.
- Either of these physical models can present the logical view of a disjoint or shared-address-space platform
 - A distributed-memory shared-address-space computer is a NUMA system

Cache Coherency Protocols



Gramma fig 2.21

Update v Invalidate

- Update Protocol
 - When a data item is written, all of its copies in the system are updated
- Invalidate Protocol (most common)
 - Before a data item is written, all other copies are marked as invalid
- Comparison
 - Multiple writes to same word with no intervening reads require multiple write broadcasts in an update protocol, but only one initial invalidation
 - Multiword cache blocks, each word written in a cache block must be broadcast in an update protocol, but only one invalidate per line is required
 - Delay between writing a word in one processor and reading the written data in another is usually less for update
- **False sharing**: two processors modify different parts of the same cache line
 - Invalidate protocol leads to ping-ponged cache lines
 - Update protocol performs reads locally but updates

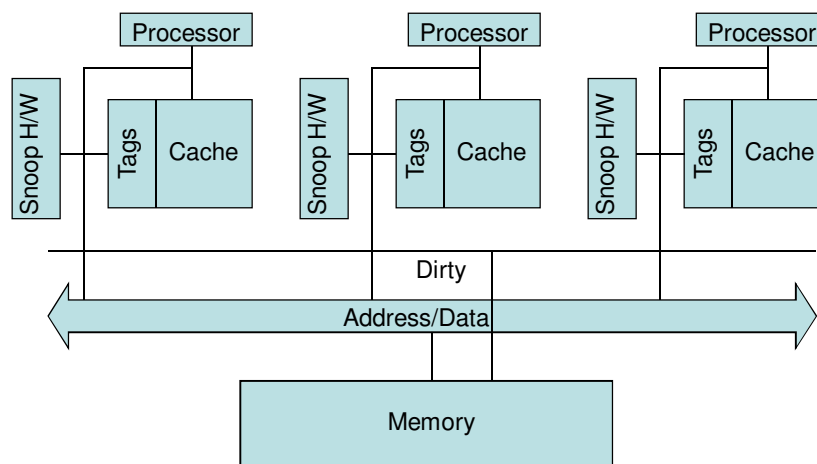
Implementation Techniques

- On small scale bus based machines
 - A processor must obtain access to the bus to broadcast a write invalidation
 - With two competing processors the first to gain access to the bus will invalidate the others' data
- A cache miss needs to locate top copy of data
 - Easy for write through cache
 - For write back each processor snoops the bus and responds by providing data if it has the top copy
- For writes we would like to know if any other copies of the block are cached
 - i.e. whether a write back cache needs to put details on the bus
 - Handled by having a tag to indicate shared status
- Minimizing processor stalls
 - Either by duplication of tags or having multiple inclusive caches

Snoopy Cache Systems

- All caches broadcast all transactions
 - Suited to bus or ring interconnects
- All processors monitor the bus for transactions of interest
- Each processors cache has a set of tag bits that determine the state of the cache block
 - Tags updated according to state diagram for relevant protocol
- Eg snoop hardware detects that a read has been issued for a cache block that it has a dirty copy of, it asserts control of the bus and puts data out
- *What sort of data access characteristics are likely to perform well/badly on snoopy based systems?*

Snoopy Cache Based System



Gramma fig 2.24

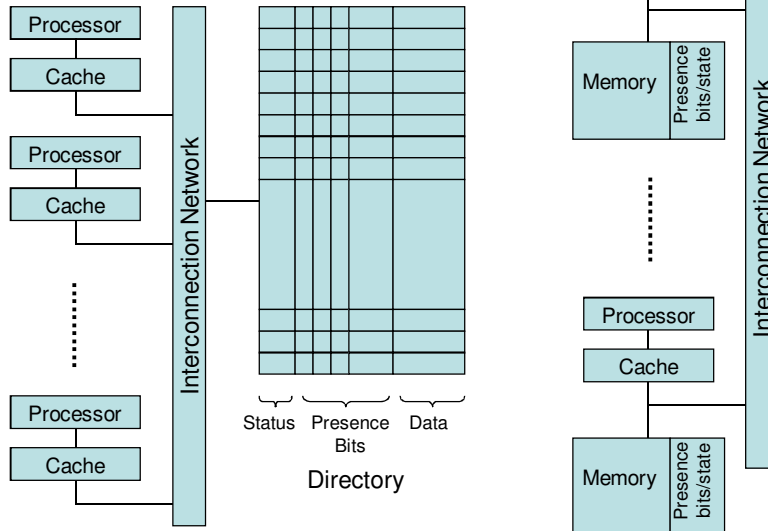
Directory Cache Based Systems

- Need to broadcast clearly not scalable
 - Solution is to only send information to processing elements specifically interested in that data
- This requires a directory to store information
 - Augment global memory with **presence bitmap** to indicate which caches memory located in

Directory Based Cache Coherency

- Must handle a read miss and a write to a shared, clean cache block
- To implement the direct must track the state of each cache block
- A simple protocol might be:
 - Shared: one or more processors have the block cached, and the value in memory is up to date
 - Uncached: no processor has a copy
 - Exclusive: only one processor (the owner) has a copy and the value in memory is out of date
- We also need to track the processors that have copies of shared cache block, or ownership of

Directory Based Cache Coherency



Grama fig 2.25

Directory Based Systems

- *How much memory is required to store the directory?*
- *What sort of data access characteristics are likely to perform well/badly on directory based systems?*
 - *How do distributed and centralized systems compare?*

Costs on SGI Origin 3000 (processor clock cycles)

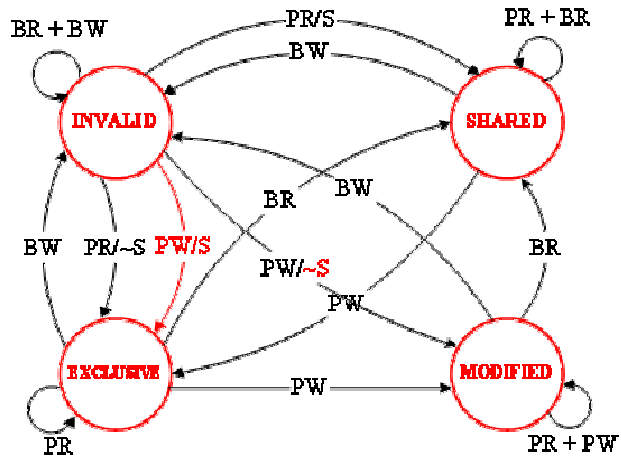
	≤16 CPU	> 16 CPU
Cache Hit	1	1
Cache miss to local memory	85	85
Cache miss to remote home directory	125	150
Cache miss to remotely cached data (3 hop)	140	170

Data from: "Computer Architecture: A Quantitative Approach", By David A. Patterson, John L. Hennessy, David Goldberg Ed 3, Morgan Kaufmann, 2003

Real Cache Coherency Protocols

- From wikipedia
 - "Most modern systems use variants of the MSI protocol to reduce the amount of traffic in the coherency interconnect. The [MESI protocol](#) adds an "Exclusive" state to reduce the traffic caused by writes of blocks that only exist in one cache. The [MOSI protocol](#) adds an "Owned" state to reduce the traffic caused by write-backs of blocks that are read by other caches [The processor owner of the cache line services requests for that data]. The [MOESI protocol](#) does both of these things. The [MESIF protocol](#) uses the "Forward" state to reduce the traffic caused by multiple responses to read requests when the coherency architecture allows caches to respond to snoop requests with data."

MESI (on a bus)



PR = processor read BR = observed bus read
 PW = processor write BW = observed bus read
 S/-S = shared/NOT shared

<https://www.cs.tcd.ie/Jeremy.Jones/vivio/caches/MESIHelp.htm>