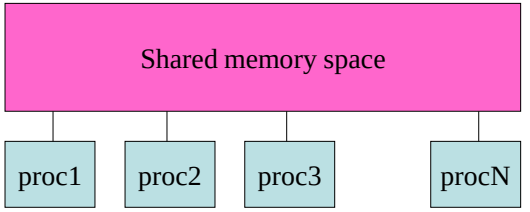
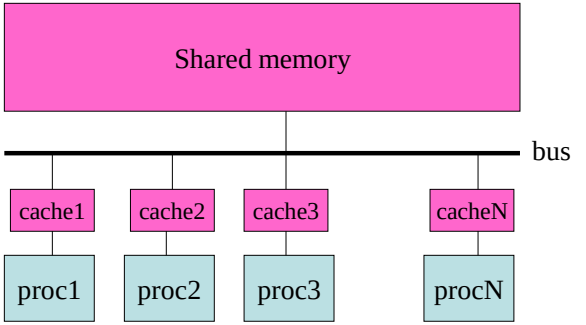
Shared Memory: A Look Underneath



57

Physical Implementation

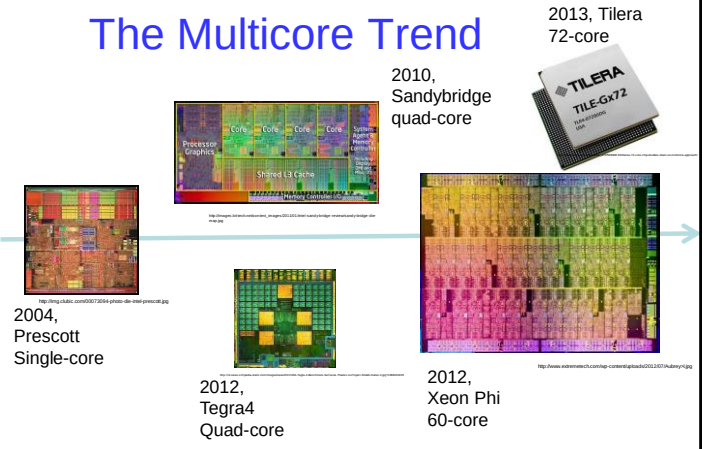


58

Multicore Processors Everywhere

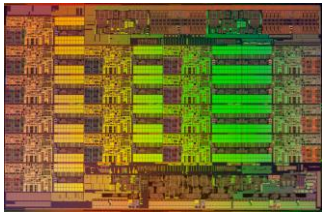
59

The Multicore Trend



60

Haswell Xeon E5 2699 V3



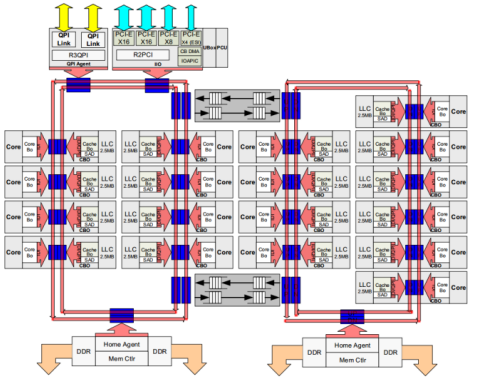
http://cdn4.wccftech.com/wp-content/uploads/2014/09/Xeon-E5-2600-V3-Die.jpg

2.3-3.66 GHz, 145W, 45M L3 cache,
2 sockets, 18 cores, 2 threads,
for a total of 72 hardware threads

61

Haswell: Logical Blocks

14-18 Core (HCC)



http://images.anandtech.com/doci/8730/1%20Die%20Config%2014-18C_678x452.png

62

Shared Memory Implementation

- Coherence - defines the behavior of reads and writes to the same memory location
  - ensuring that modifications made by a processor propagate to all copies of the data
  - Program order preserved
  - Writes to the same location by different processors serialized
- Synchronization - coordination mechanism
- Consistency - defines the behavior of reads and writes with respect to access to other memory locations
  - defines when and in what order modifications are propagated to other processors

63

Coherence

A multiprocessor memory system is coherent if the results of any execution of a program are such that, for each location, it is possible to construct a hypothetical serial order of all operations to the location that is consistent with the result of the execution and

- it ensures that modifications made by a processor propagate to all copies of the data
- program order is preserved for each process in this hypothetical order
- writes to the same location by different processors are serialized and the value returned by each read is the value written by the last write in the hypothetical order

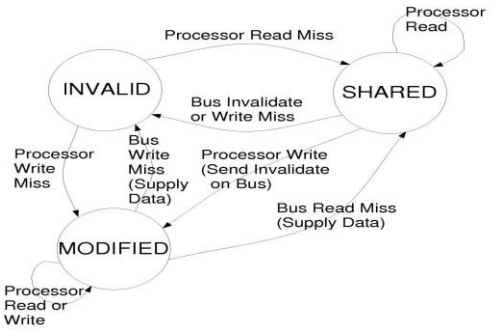
64

Snoop-Based Coherence

- Makes use of a shared broadcast medium to serialize events (all transactions visible to all controllers and in the same order)
  - Write update-based protocol
  - Write invalidate-based (e.g., basic MSI, MESI protocols)
- Cache controller uses a finite state machine (FSM) with a handful of stable states to track the status of each cache line
- Consists of a distributed algorithm represented by a collection of cooperating FSMs

65

A Simple Invalidate-Based Protocol - State Transition Diagram



66

Four-State MESI Protocol

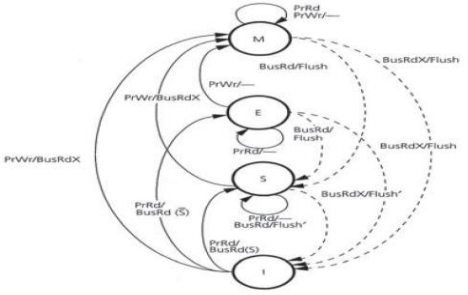


FIGURE 5.15 State transition diagram for the Illinois MESI protocol. MESI stands for the modified (dirty), exclusive, shared, and invalid states, respectively. The notation is the same as that in Figure 5.13. The E state helps reduce bus traffic for sequential programs where data is not shared. Whenever feasible, the Illinois version of the MESI protocol makes caches, rather than main memory, supply data for BusRd and BusRdX transactions. Since multiple processors may have a copy of the memory block in their cache, we need to select only one to supply the data on the bus. Flush' is true only for that processor; the remaining processors take their usual action (invalidation or no action). In general, Flush' in a state diagram indicates that the block is flushed only if cache-to-cache sharing is in use and then only by the cache that is responsible for supplying the data.

From Parallel Computer Architecture: Culler, Singh, and Gupta

67

Correctness Requirements

- Need to avoid
  - Deadlock – caused by a cycle of resource dependencies
  - Livelock – activity without forward progress
  - Starvation – extreme form of unfairness where one or more processes do not make forward progress while others do

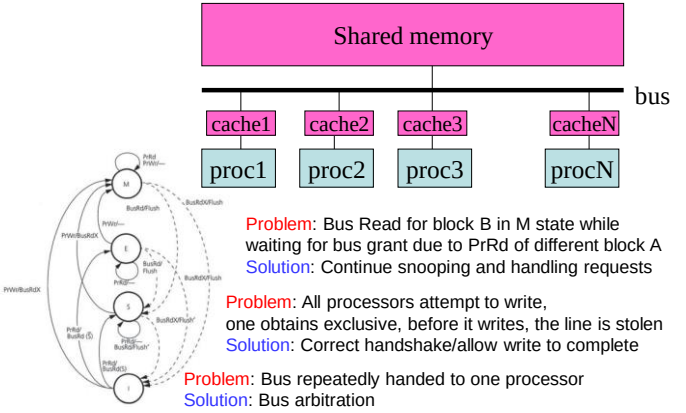
69

Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously:

- **Mutual exclusion:** only one process at a time can use a resource
- **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes
- **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task
- **Circular wait:** there exists a set $\{P_0, P_1, \ldots, P_n, P_0\}$ of waiting processes such that
  - P_0 is waiting for a resource that is held by P_1,
  - P_1 is waiting for a resource that is held by P_2,
  - …,
  - P_{n-1} is waiting for a resource that is held by P_n,
  - and P_n is waiting for a resource that is held by P_0.

Example Deadlock, Livelock, Starvation



Problem: Bus Read for block B in M state while waiting for bus grant due to PrRd of different block A
Solution: Continue snooping and handling requests

Problem: All processors attempt to write, one obtains exclusive, before it writes, the line is stolen
Solution: Correct handshake/allow write to complete

Problem: Bus repeatedly handed to one processor
Solution: Bus arbitration

MESI Protocol with Transient States

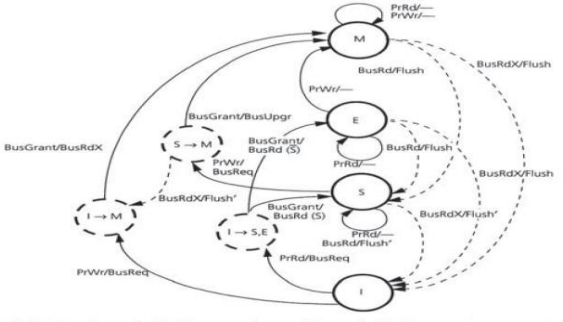


FIGURE 6.5 Expanded MESI protocol state diagram indicating transient states for bus acquisition. The cache controller monitors the bus while arbitration is ongoing for its request. A conflicting transaction may change the transition between stable states.

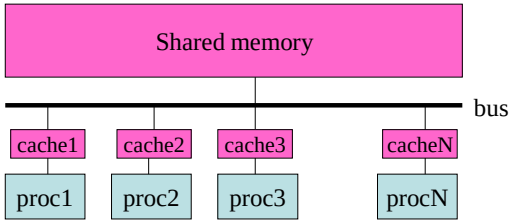
From Parallel Computer Architecture: Culler, Singh, and Gupta

Design Challenges

- Cache controller and tag design
- Non-atomic state transitions
- Serialization
- Cache hierarchies
- Split-transaction buses

Snoop-Based or Broadcast Coherence

- Make use of a broadcast medium to manage replicas
- Benefit: Low metadata requirements
- Challenge: High bandwidth requirements

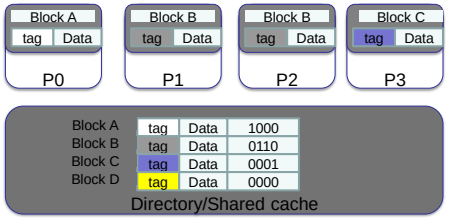


74

Solution: Directory-based Cache Coherence

Directory: maintain per-core sharer information to save bandwidth

Full map: associate sharing vector with tags of shared L2



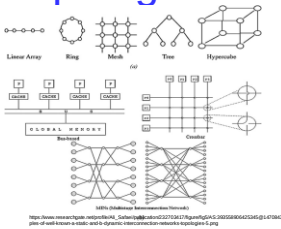
75

Multiprocessor Interconnects

- Topology
- Routing algorithm
- Switching strategy (circuit vs. packet)
- Flow control mechanism

76

Interconnect Topologies



- Fully connected
  - Single large switch
  - Bus
- Linear arrays and rings
- Multi-dimensional meshes and tori
- Trees
- Multi-stage interconnection network: e.g.Butterfly
- Hypercube

77

Butterfly Network

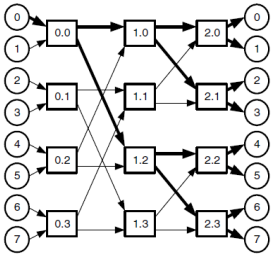


Figure 2.1 An 8-node butterfly network. Data flows from the input nodes on the left (circles) through three stages of switch nodes (rectangles) to the output nodes on the right (circles). The switch nodes are labeled with their stage and address. All channels are unidirectional, as indicated by the arrows.

Dally and Towles: https://dl.acm.org/doi/pdf/10.5555/2821589

78

Switching Strategy

- Circuit-switched: first packet sets up route, subsequent packets follow route without any header processing
- Packet-switched: each packet is independently routed
  - Store-and-forward: each hop receives all packets of a message before forwarding it on
  - Cut-through: each packet forwarded as soon as it is received
  - Virtual cut-through: cut-through routing, but buffer packets when there is contention
  - Wormhole routing: packet spread across multiple hops, in effect holding a circuit open.

79

Metrics

- Hardware cost – number of wires, pin count, length of wires, physical arrangement
- Topology diameter
  - Length of maximum shortest path between any two nodes in the network
- Latency
  - Overhead+routing_delay+channel_occupancy(bandwidth)+contention_delay
- Bandwidth – local, global, bisection
  - Bisection bandwidth
    - Sum of bandwidths of minimum set of channels/links that, if removed, partitions the network into 2 euqal unconnected sets of nodes

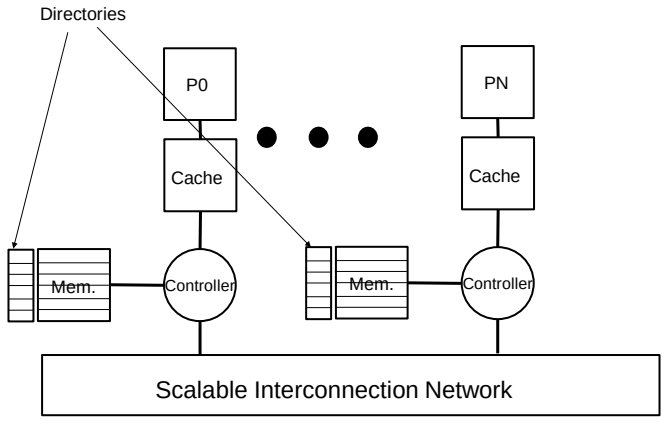
80

Directory-Based Coherence

- Distribute memory, use point-to-point interconnect for scalability
- Need to manage coherence for each memory line – state stored in directory
  - Simple memory-based (e.g., DASH, FLASH, SGI Origin, MIT Alewife, HAL)
  - Cache-based (linked list (e.g., Sequent NUMA-Q, IEEE SCI)

81

Scalable Multiprocessor



82

Memory-based Directory Cache Coherence

- Directory:

  Maintains per-core sharing information to save bandwidth

- Current Designs

Shadow tags: duplicate L1 tags, look up to create sharing vector

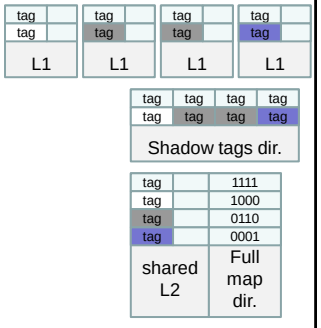
+ support non-inclusive cache

- high lookup energy

Full map: associate sharing vector with tags of shared L2

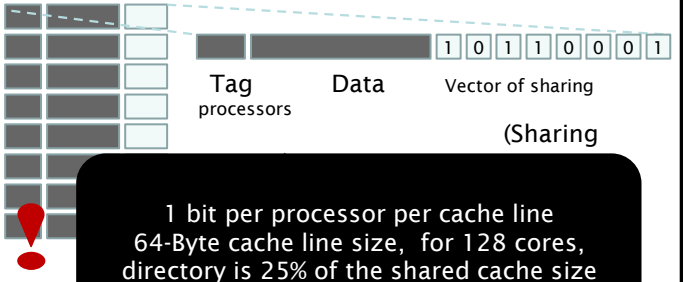
+ negligible lookup energy

- large area



83

Conventional Full Map Directory



1 bit per processor per cache line
64-Byte cache line size, for 128 cores, directory is 25% of the shared cache size

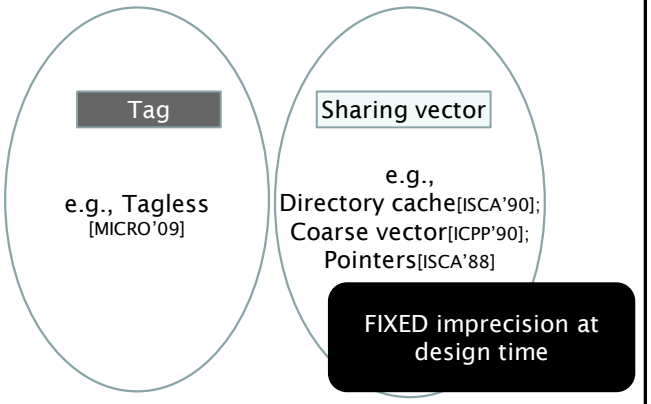
84

Simple Memory-based Directory Coherence

- Advantage
  - Precise sharing information
- Disadvantage
  - Space/storage proportional to PxM
- Work-around for either width or height
  - Increase cache block size
  - 2-level protocol
  - Limited pointer scheme
  - Directory cache

85

Scaling directory designs



86

Cache-Based Directory Coherence

- Home main memory contains a pointer to the first sharer + state bits
- Pointers at each cache line to maintain a doubly-linked list
- Advantage – reduced space overhead
- Disadvantage – serialized invalidates (latency and occupancy)

87

Summary

- Non-atomic state transitions complicate coherence implementation
- Directory protocols used to scale processors to large core counts
- Relaxed consistency models allow read/write reorderings
  - Implications for the hardware
  - Implications for the compiler

88

A Framework for Sharing Patterns

- Predictable vs. unpredictable
- Regular vs. irregular
- Coarse vs. fine-grain (contiguous vs. non-contiguous in the address space)
- Near-neighbor vs. long range in an interconnection topology
- In terms of invalidation patterns
  - Read-only
  - Producer-consumer
  - Broadcast/multicast
  - Migratory
  - Irregular read-write

89