CMSC 411
Computer Systems Architecture
Lecture 23
Multiprocessors, cont.

Alan Sussman
als@cs.umd.edu

Administrivia

- Finish reading Chapter 5
- Homework 5 posted, due Tuesday
- Exam 2 answers posted today
  - and exam returned Tuesday
- Project 2 – what was the problem?
- Course evaluations open, at https://CourseEvalUM.umd.edu

SNOOPING WRITE BACK PROTOCOL

Limitations in Symmetric Shared-Memory Multiprocessors and Snooping Protocols

- Single memory to accommodate all CPUs
  - ⇒ Multiple memory banks
- Bus must support both coherence traffic & normal memory traffic
  - ⇒ Multiple buses or interconnection networks (cross bar or small point-to-point)
- Example - AMD Opteron:
  - Memory connected directly to each dual-core chip
  - Point-to-point connections for up to 4 chips
  - Remote memory and local memory latency are similar, allowing OS to view Opteron as uniform memory access (UMA) computer

Performance of Symmetric Shared-Memory MPs

- **Cache performance is combination of**
  1. **Uniprocessor cache miss traffic**
  2. **Traffic caused by communication**
     - » **Results in invalidations and subsequent cache misses**
- **4th C: *coherency miss***
  - **Joins Compulsory, Capacity, Conflict**

Coherency Misses

1. True sharing misses arise from the communication of data through the cache coherence mechanism
   - Invalidates due to 1st write to shared block
   - Reads by another CPU of modified block in different cache
   - *Miss would still occur if block size were 1 word*
2. False sharing misses when a block is invalidated because some word in the block, other than the one being read, is written into
   - Invalidation does not cause a new value to be communicated, but only causes an extra cache miss
   - Block is shared, but no word in block is actually shared
     ⇒ miss would not occur if block size were 1 word

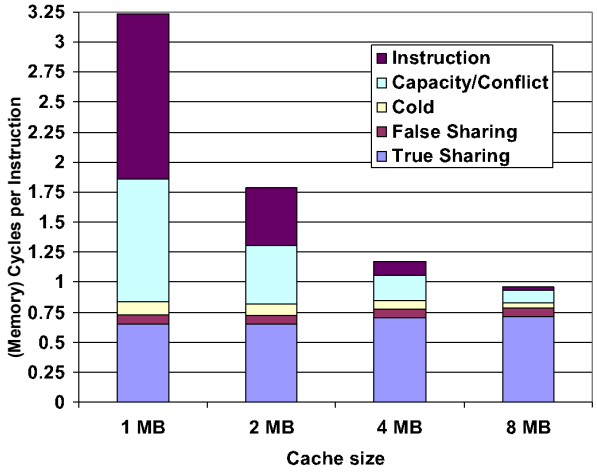
Example: True v. False Sharing v. Hit?

- **Assume x1 and x2 in same cache block. P1 and P2 both read x1 and x2 before.**

Time	P1	P2	True, False, Hit? Why?
1	Write x1		True miss; invalidate x1 in P2
2		Read x2	False miss; x1 irrelevant to P2
3	Write x1		False miss; x1 irrelevant to P2
4		Write x2	False miss; x1 irrelevant to P2
5	Read x2		True miss; invalid x2 in P1

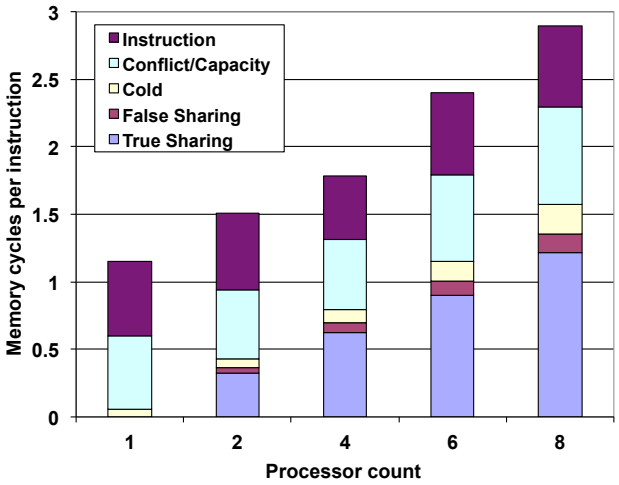
MP Performance 4 Processor Commercial Workload: OLTP, Decision Support (Database), Search Engine

- True sharing and false sharing unchanged going from 1 MB to 8 MB (L3 cache)
- Uniprocessor cache misses improve with cache size increase (Instruction, Capacity/ Conflict, Compulsory)



MP Performance 2MB Cache
Commercial Workload: OLTP, Decision Support (Database), Search Engine

• True sharing, false sharing increase going from 1 to 8 CPUs



Outline

- Review
- Coherence
- Write Consistency
- Administrivia
- Snooping
- Building Blocks
- Snooping protocols and examples
- Coherence traffic and Performance on MP
- **Directory-based protocols and examples**
- **Conclusion**

A Cache Coherent System Must:

- **Provide set of states, state transition diagram, and actions**
- **Manage coherence protocol**
  - **(0) Determine when to invoke coherence protocol**
  - **(a) Find info about other copies to determine action**
    - » **whether need to communicate with other cached copies**
  - **(b) Locate the other copies**
  - **(c) Communicate with those copies (invalidate/update)**
- **(0) is done the same way on all systems**
  - **state of the line is maintained in the cache**
  - **protocol is invoked if an "access fault" occurs on the line**
- **Different approaches distinguished by (a) to (c)**

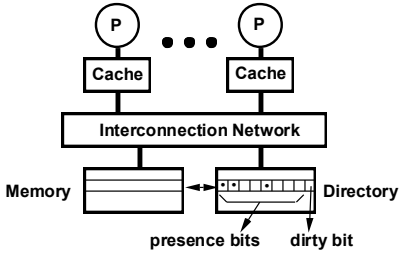
Bus-based Coherence

- **All of (a), (b), (c) done through broadcast on bus**
  - **faulting processor sends out a "search"**
  - **others respond to the search probe and take necessary action**
- **Could do it in scalable network too**
  - **broadcast to all processors, and let them respond**
- **Conceptually simple, but broadcast doesn't scale with *p* (number of processors)**
  - **on bus, bus bandwidth doesn't scale**
  - **on scalable network, every fault leads to at least p network transactions**
- **Scalable coherence:**
  - **can have same cache states and state transition diagram**
  - **different mechanisms to manage protocol**

Scalable Approach: Directories

- Every memory block has associated directory information
  - keeps track of copies of cached blocks and their states
  - on a miss, find directory entry, look it up, and communicate only with the nodes that have copies if necessary
  - in scalable networks, communication with directory and copies is through network transactions
- Many alternatives for organizing directory information

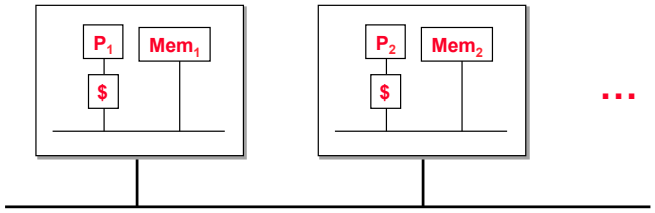
Basic Operation of Directory



- *k* processors.
- With each cache-block in memory: *k* presence-bits, 1 dirty-bit
- With each cache-block in cache: 1 valid bit, and 1 dirty (owner) bit

- Read from main memory by processor i:
  - If dirty-bit OFF then { read from main memory; turn p[i] ON; }
  - if dirty-bit ON then { recall line from dirty proc (cache state to shared); update memory; turn dirty-bit OFF; turn p[i] ON; supply recalled data to i;}
- Write to main memory by processor i:
  - If dirty-bit OFF then { supply data to i; send invalidations to all caches that have the block; turn dirty-bit ON; turn p[i] ON; ... }
  - ...

Picture To Keep In Mind



- Cost structure
  - cache ~2 cycles
  - local memory ~80 cycles
  - remote memory ~200 cycles

Directory Protocol

- Similar to Snoopy Protocol: Three states
  - Shared: ≥ 1 processors have data, memory up-to-date
  - Uncached (no processor has it; not valid in any cache)
  - Exclusive: 1 processor (owner) has data; memory out-of-date
- In addition to cache state, must track which processors have data when in the shared state (usually bit vector, 1 if processor has copy)
- Keep it simple(r):
  - Writes to non-exclusive data
    - » write miss
  - Processor blocks until access completes
  - Assume messages received and acted upon in order sent

Directory Protocol 2

- No bus and don't want to broadcast:
  - interconnect no longer single arbitration point
  - all messages have explicit responses
- Terms: typically 3 processors involved
  - Local node where a request originates
  - Home node where the memory location of an address resides
  - Remote node has a copy of a cache block, whether exclusive or shared
- Example messages on next slide: P = processor number, A = address

State Transition Diagram for One Block

- States identical to snoopy case; transactions very similar.
- Transitions caused by read misses, write misses, invalidates, data fetch requests
- Generates read miss & write miss msg to home directory.
- Write misses that were broadcast on the bus for snooping => explicit invalidate & data fetch requests.
- Note: on a write, a cache block is bigger, so need to read the full cache block – write allocate

CPU -Cache State Machine

- State machine for *CPU* requests for each memory block
- Invalid state if in memory