COMP 633  - Parallel Computing

Lecture 10
September 21, 2017

**CC-NUMA (1)**

**CC-NUMA implementation**

- Reading for next time
  - Memory consistency models tutorial (sections 1-6, pp 1-17)
Topics

• Objectives of the next few lectures
  – Examine some implementation issues in shared-memory multiprocessors
    • cache coherence
    • memory consistency
    • synchronization mechanisms

• Why?
  – Correctness
    • these issues can be the source of very subtle bugs
  – Performance
    • these issues may have profound performance implications
Cache-coherent shared memory

- Implementation
  - shared bus
    - bus may be a “slotted” ring
  - point-to-point interconnect
    - e.g. Intel QPI

- Effect of CPU write on local cache
  - write-through policy – value is written to cache and to memory
  - write-back policy – value written in cache only; memory updated on cache line eviction

- Effect of CPU write on remote cache
  - update – remote value is modified
  - invalidate – remote value is marked invalid
Bus-Based Shared-Memory protocols

- “Snooping” caches
  - $C_i$ caches memory operations from $P_i$
  - $C_i$ monitors all activity on bus due to $C_h$ ($h \neq i$)

- **Update** protocol with *write-through* cache
  - between proc $P_i$ and cache $C_i$
    - read-hit from $P_i$ resolved from $C_i$
    - read-miss from $P_i$ resolved from memory and inserted in $C_i$
    - write (hit or miss) from $P_i$ updates $C_i$ and memory [write-through]
  - between cache $C_i$ and cache $C_h$
    - if $C_i$ writes a memory location cached at $C_h$, then $C_h$ is updated with new value

- consequences
  - every write uses the bus
  - doesn’t scale
Bus-Based Shared-Memory protocols

- **Invalidation protocol with write-back cache**
  - Cache blocks can be in one of three states:
    - **INVALID** — The block does not contain valid data
    - **SHARED** — The block is a current copy of memory data
      - other copies may exist in other caches
    - **EXCLUSIVE** — The block holds the only copy of the correct data
      - memory may be incorrect, no other cache holds this block
  - Handling exclusively-held blocks
    - **Processor events**
      - cache is block “owner”
        - reads and writes are local
    - **Snooping events**
      - on detecting a read-miss or write-miss from another processor to an exclusive block
        - write-back block to memory
        - change state to shared (on ext read-miss) or invalid (on ext write-miss)
Invalidation protocol: example
Implementation: FSM per cache line

- Action in response to CPU event
  - Invalid
  - Shared
  - Excl

- Action in response to bus event
  - Invalid
  - Shared
  - Excl
Scalable shared memory: directory-based protocols

- The Stanford DASH multiprocessor
  - Processing clusters are connected via a scalable network
    - Global memory is distributed equally among clusters
  - Caching is performed using an ownership protocol
    - Each memory block has a “home” processing cluster
    - At each cluster, a directory tracks the location & state of each cached block whose home is on the cluster
Directories track location & state of all cache blocks
- 16 MB cluster memories
- 16 byte cache blocks
- 2+ MB storage overhead per directory
Cache coherence in DASH

- Caching is based on an ownership model – *uncached*, *clean*, & *dirty* states

- Home cluster is the owner for all its *uncached* and *clean* blocks

- Any one cache can own the only copy of a *dirty* block
Cache coherence in DASH: Read miss

- Check local cluster caches first...
  - If found and CLEAN then copy
  - If found and DIRTY then make CLEAN and copy

- If not found consult desired block’s home directory
  - If CLEAN or UNCACHED then block is sent to requestor
  - If DIRTY then request is forwarded to cluster where block is cached. Remote cluster makes block CLEAN and sends copy to requestor

- To make a block CLEAN
  - Send copy to owning cluster
  - mark CLEAN
Cache coherence in DASH: Writes

- Writing processor must first become block’s owner

- If block is cached at requesting processor and block is...
  - DIRTY, then write can proceed
  - CLEAN, then home directory must invalidate all copies

- If block is not cached locally but is cached on the cluster
  - a local block transfer is performed
  - home directory is updated if the state was CLEAN
Cache coherence in DASH: Writes

- If block is not cached on local cluster then block’s home directory is contacted

- If block is...
  - UNCACHED — Block is marked DIRTY and sent to requestor
  - CLEAN — Block is marked as DIRTY and messages sent to caching clusters to invalidate their copies
  - DIRTY — Request is forwarded to caching cluster. There the block is invalidated and forwarded to requestor
Coherence and Consistency

• Coherence
  – behavior of a single memory location
  – viewed from a single processor
  – read returns “most recent” written value

• Consistency
  – behavior of multiple memory locations read and written by multiple processors
  – viewed from one or more of the processors
  – read may not return the “most recent” value
    • What are the permitted ordering among reads and writes of several memory locations?