Module-4
CSEN 3104
Lecture 31
21/10/2019

Dr. Debranjan Sarkar

Vector Memory-Memory versus Vector Register Machines

- Vector memory-memory instructions hold all vector operands in main memory
- The first vector machines, TI ASC (1971), and CDC Star-100 (1973) were memory-memory machines
- Cray-1 (1976) was first vector register machine

- Example Source Code
- for (i=0; i<n; i++)
- {
- c[i] = a[i] + b[i];
- d[i] = a[i] – b[i];
- }

Vector Memory-Memory versus Vector Register Machines

- Vector Memory-Memory code
- ADDV C, A, B
- SUBV D, A, B

- Vector Register code
- LV V1, A
- LV V2, B
- ADDV V3, V1, V2
- SV V3, C
- SUBV V4, V1, V2
- SV V4, D

Vector Memory-Memory vs. Vector Register Machines

- Vector memory-memory architectures (VMMA) require greater main memory bandwidth
  - Because all operands must be read in and out of memory
- VMMAs make it difficult to overlap execution of multiple vector operations
  - Because dependencies must be checked on memory addresses
- VMMAs incur greater startup latency
  - Scalar code was faster on CDC Star-100 for vectors < 100 elements
  - For Cray-1, vector/scalar breakeven point was around 2 elements

Cache Coherence

Cache Coherence Problem

- In a memory hierarchy for a multi-processor system, data inconsistency may occur between adjacent levels or within the same level
- For example, cache and main memory may contain inconsistent copies of the same data object
- Multiple copies may possess different copies of the same memory block because multiple processors operate asynchronously and independently
- Caches in a multi-processing environment introduce the cache coherence problem
- When multiple processors maintain locally cached copies of a unique shared-memory location, any local modification of the location may result in a globally inconsistent view of memory
- Cache coherence schemes prevent this problem by maintaining a uniform state for each cached block of data

Cache inconsistency: Causes

- Maintaining cache coherency is a problem in multiprocessor system when the processors contain local cache memory (multiple private caches are used)
- 3 sources of the problem are identified:
  - Sharing of writable data
  - Process migration
  - I/O activity
- Inconsistency in sharing writable data (Show figure)
- Consider a multiprocessor with 2 processors, each with private cache and both sharing the main memory
- Let X be a data element which has been referenced by both the processors
- Before update, the three copies of X are consistent
- Under write-through policy, inconsistency occurs between the 2 copies in the 2 caches

Cache inconsistency: Causes

- Inconsistency due to process migration (Show figure)
- Inconsistency occurs in case of both the write-back and write-through policies
- Special precautions must be exercised to avoid such inconsistencies
- A coherence protocol must be established before processes can migrate safely from one process to another
- Inconsistency during IO operation bypassing cache (Show figure)
- When the IO processor loads a new data X’ into the main memory, bypassing the write-through caches, inconsistency occurs between the cache and Main Memory
- When outputting a data directly from the shared memory (bypassing the caches), the write-back caches also create inconsistency

Cache inconsistency: Solutions

- Solution to IO inconsistency problem (Show figure)
- Attach the IO processor (IOP1 and IOP2) to the private caches (C1 and C2)
- Thus the IO Processors share caches with the CPU
- The IO inconsistency can be maintained if cache-to-cache consistency is maintained via the bus
- Shortcomings:
  - Likely increase in cache perturbations
  - Poor locality of IO data
  - These may result in higher miss rates

Thank you

Module-4
CSEN 3104
Lecture 32
31/10/2019

Dr. Debranjan Sarkar

Cache Coherence

Cache coherence mechanisms

- Two protocol approaches
  - Shared bus: Snoopy protocol
  - Other interconnection schemes: Directory protocol

Cache coherence mechanisms: Snoopy Protocol

- Early multiprocessors used bus-based memory systems
- Bus allows all the processors to observe ongoing memory transactions
- If a bus transaction threatens the consistent state of a locally cached object, the cache controller can invalidate the local copy
- Protocols using this mechanism to ensure coherence is called Snoopy Protocol
- Each processor tracks sharing status of each block

Snoopy Bus Protocol

- Snoopy protocols achieve data consistency between the cache memory and the shared memory through a bus-based memory system.
- Policies used for maintaining cache consistency:
  - **Write-invalidate**
  - **Write-update**
- ‘Write Invalidate’ policy will invalidate all remote copies when a local cache block is updated
- ‘Write Update’ policy will broadcast the new data block to all caches containing a copy of the block
- We have three processors P_1, P_2, and P_n having a consistent copy of data element ‘X’ in their local cache memory and in the shared memory (Fig. a)

Snoopy Bus Protocol

- Processor P_1 writes X' in its cache memory using **write-invalidate protocol**
- So, all other copies are invalidated via the bus, denoted by 'I' (Fig. b)
- Invalidated blocks are also known as **dirty**, i.e. they should not be used
- The **write-update protocol** updates all the cache copies via the bus through broadcast mechanism
- The memory copy is also updated, if write-through caches are used (Fig. c)
- The memory copy is updated later at block replacement time, in case of write-back caches

Disadvantages of Snoopy Bus Protocols

- Write-invalidate protocol may lead to heavy bus traffic, caused by read-misses, resulting from the processor updating a variable and other processors trying to read the same variable
- Write-update protocol may update data items in remote caches which will never be used by other processors
- These problems pose additional limitations in using buses to build large multiprocessors

Cache coherence mechanisms: Directory Based Protocols

- To overcome the limitations of using bus in building large (scalable) multiprocessor system, multistage network is used to interconnect processors
- Unlike the bus-based system, the bandwidth of these networks increases as more processors are added to the system
- Broadcasting is very expensive in a multistage network
- Hence, the consistency commands is sent only to those caches that keep a copy of the block
- Such networks do not have a convenient snooping mechanism and do not provide an efficient broadcast capability
- This is the reason for development of directory-based protocols for network-connected multiprocessors to solve the cache coherence problem

Directory based Protocols

- Sharing status of each block kept in one location
- In a directory-based protocol system, data to be shared are placed in a common directory that maintains the coherence among the caches.
- Here, the directory acts as a filter where the processors ask permission to load an entry from the primary memory to its cache memory.
- If an entry is changed the directory either updates it or invalidates the other caches with that entry.

Directory based Protocols

- *Directory based protocols* have a main directory containing information on shared data across processor caches
- The directory works as a look-up table for each processor to identify coherence and consistency of data which is currently being updated
- A directory-based protocol is a smart way of implementing cache consistency on an arbitrary interconnection network
- While the resulting protocol is complex, it is indeed tractable
- Moreover, the hardware needed to implement such a protocol is quite reasonable for the scale of machine in which it is expected to be used

Directory based Protocols

- Various directory-based protocols differ mainly in:
  - how the directory maintains information, and
  - what information it stores
- Central Directory based protocol
  - Uses a central directory which contains duplicates of all cache directories
  - The central directory provides all the information to enforce consistency
  - It is usually very large in size
  - Must be associatively searched
  - Chance of bottleneck
  - Drawbacks for a large multiprocessor system:
    - Contention, and
    - Long search time

Directory based Protocols

- Distributed Directory based protocol
  - Each memory module maintains a separate directory
  - Each directory records the state and presence information for each memory block
  - The state information is local
  - The presence information indicates which caches have a copy of the block
  - Bottleneck is avoided
- Directory based protocols do not use broadcasts
- So, the locations of all cached copies of each block of shared data must have to stored
- The list of cache locations is called a cache directory
- It may be centralized or distributed
- A directory entry for each block of data contains a number of pointers to specify the locations of copies of the block
- Each directory entry also contains a dirty bit to specify whether a particular cache has permissions to write the associated block of data

A few References and Bibliography

- Computer Architecture and Organization *by* John P. Hayes, (WCB/McGraw Hill)
- Advanced Computer Architecture: Parallelism, Scalability, Programmability *by* Kai Hwang & Naresh Jotwani (Tata McGraw Hill Education Pvt. Ltd.)
- Computer Architecture and Parallel Processing *by* Kai Hwang & Faye A. Briggs (McGraw Hill Book Company)

Thank you