A Unified Synchronization and Cache Coherence Mechanism

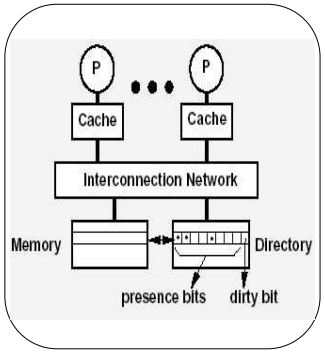
Zhenghua Qi, Raksit Ashok, Richard Weiss* and Csaba Andras Moritz

Electrical and Computer Engineering Department
University of Massachusetts, Amherst

*Hampshire College

Motivation

- Continued advances in technology result in highly capable and complex multiprocessor systems
- Synchronization in shared-memory systems becomes more and more important to address
  - E.g., in fine-grained single-chip multiprocessors
  - Synchronization cost increases with # of processor nodes
- *Synchronization Coherence:* provides a novel solution to support fine-grained synchronization in hardware and transparently to processor nodes



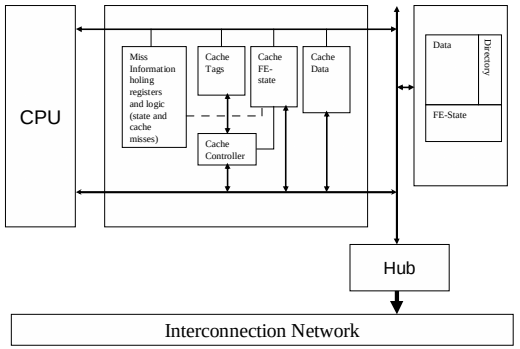
A Directory-based Multiprocessor System

Synchronization Overview

- Coarse-grained Synchronization — *A synchronizing variable is associated with multiple shared locations and is used to control the order of accesses to these locations.*
- Speculative Synchronization (*J.R. Martinez et al*, 2002)
- Speculative Lock Elision (*R.Rajwar et al*, 2001)
- Fine-grained synchronization — *A synchronizing variable is associated with a single word or a block of memory.*
  - Examples: HEP, Tera, and MIT's Alewife machine.

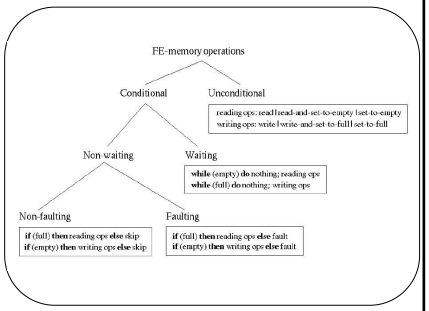
Fine-Grained Synchronization (FGS) — *Implementation*

- Hardware Support
  - A fine-grained synchronization mechanism can be implemented with FE-memory, where an F/E bit is associated with each word.



Fine-Grained Synchronization — *Implementation*

- Full/Empty memory operations
- Typically supported in SW on top of cache coherence (Alewife)
- FGS access semantics
  - J structure
  - L structure
  - M structure
- We implement J structures
- Combine cache coherence and synchronization into one mechanism



What is Synchronization Coherence?

- An efficient mechanism, unifying the cache-coherence and fine-grained synchronization mechanisms in hardware
- Treats a synchronization miss a special case of cache miss, resolving by hardware
- Implementation of non-faulting and waiting FE memory operations
- Key benefits
  - Reduced occupancy in the memory controllers and network bandwidth consumed by protocol messages
  - Reduced synchronization related overheads
  - FGS that is transparent to the processor nodes

Synchronization Coherence — *Architecture Support*

FE-bit:

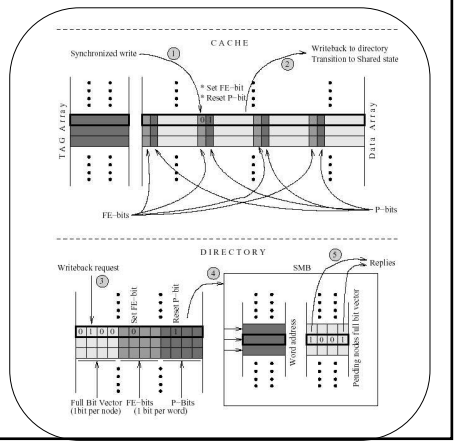
1 – The data is ready to use

0 – The data is not ready yet

P-bit (pending):

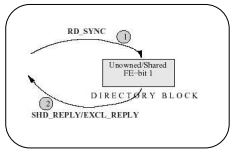
1 – There are consumers waiting for data

0 – No pending consumers

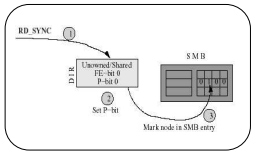


Synchronization Coherence — Protocol *Examples*

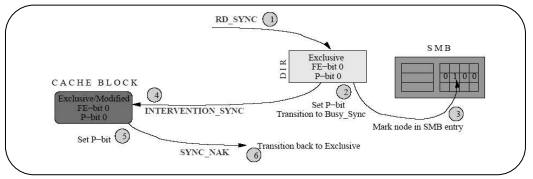
- Case1: Synchronization read hit



- Case2: Synchronization read miss



- Case 3: Synchronization read miss



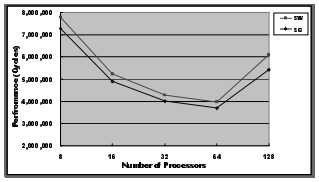
Experimental Setup

- Simulator modules
  - Developed on top of Sim-out-order core from *simple-scalar3.0*
  - Extension module for thread control.
  - Directory-based cache-coherence module
  - Interconnection module supporting k-ary n-cube networks
- Benchmark — *MICCG3D*
  - Solving Ax = b, using LU factorization. Where, A is a sparse matrix, b is a given vector and x is the vector to be solved, representing data in a 3-D space
- Configuration

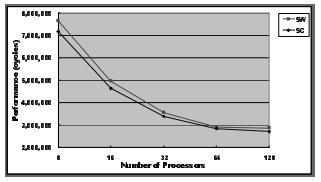
L1 D-Cache	16kB, 4-way, 32byte line
L1 Hit Latency	1 cycle
DRAM Latency	18 cycle
Interconnect Layout	4x2, 4x4, 8x4, 8x8, 16x8,16x16 2^3, 2^4, 2^5, 2^6, 2^7, 2^8
Flit size	32 bits
Interconnect speed	2 cycles per hop
Router delay	2 cycles for the first flit
Message launch delay	4 cycles

Results — *Performance*

Input Data Size: x = y = 8, z = 256
SC: Synchronization Coherence Mechanism
SW: Software-based FG Mechanism (baseline)



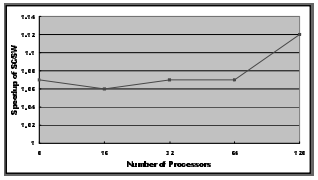
Performance Improvement for 2-D Mesh Network



Performance Improvement for Hypercube Network

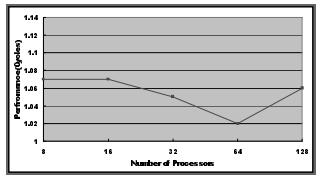
Results — *Speedup*

Input Data Size: x = y = 8, z = 256
SC: Synchronization Coherence Mechanism
SW: Software-based Mechanism (baseline)



Speedup for 2-D Mesh Network

Average: 8%



Speedup for Hypercube Network

Average: 5%

Conclusions

- Synchronization coherence can reduce the synchronization related overhead and improve performance
- Future work:
  - More applications need to be evaluated
  - Support for better and adaptive routing protocols
  - More efficient non-blocking cache model
  - Compilation support
  - SMP applicability