Outline

- Symmetric Multiprocessors (SMPs) (CC-UMAs)
  - Cache & Coherence Problem
  - Snooping Cache Coherence
  - MOESI & Sun E6000

- Distributed Shared Memory (CC-NUMAs)

- More Generally

What is (Hardware) Shared Memory?

- Take
  - Classic: multiple (micro-)processors
  - Emerging: multiple processor “cores” on a chip

- Implement a memory system with
  a single global physical address space (usually)

- Minimize memory latency (co-location & caches)

- Maximize memory bandwidth (parallelism & caches)
Snooping Cache-Coherence Protocols

- Use Symmetric Multiprocessors (SMPs) a.k.a. Cache-coherent Uniform Memory Access (CC-UMA)
- Bus provides serialization point (more on this later)
- Each cache controller “snoops” all bus transactions
  - relevant transactions if for a block it contains
  - take action to ensure coherence
    - invalidate
    - update
    - supply value
  - depends on state of the block and the protocol
- Simultaneous Operation of Independent Controllers

Snooping Design Choices

- Controller updates state of blocks in response to processor and snoop events and generates bus actions
- Often have duplicate cache tags
- Snoopy protocol
  - set of states
  - state-transition diagram
  - actions
- Basic Choices
  - write-through vs. write-back
  - invalidate vs. update

The Simple Invalidate Snooping Protocol

- Write-through, no-write-allocate cache
- Actions: PrRd, PrWr, BusRd, BusWr
More Generally: MOESI

- Modified (dirty)
- Owned (dirty but shared)
- Exclusive (clean unshared) only copy, not dirty
- Shared
- Invalid

Variants
- MSI
- MESI
- MOBI
- MOESI

Ownership

Validity

Exclusiveness

Cache Block Size

- Block size is unit of transfer and of coherence
  - Doesn’t have to be, could have coherence smaller [Goodman]
- Unprocessor 3C’s
  - Compulsory, Capacity, Conflict
- SM adds Coherence Miss Type
  - True Sharing miss fetches data written by another processor
  - False Sharing miss results from independent data in same coherence block
- Increasing block size
  - Usually fewer 3C misses but more bandwidth
  - Usually more false sharing misses
- P.S. on increasing cache size
  - Usually fewer capacity/conflict misses (& compulsory don’t matter)
  - No effect on true/false “coherence” misses (so may dominate)
Qualitative Sharing Patterns

- [Weber & Gupta, ASPLOS3]
- Read-Only
- Migratory Objects
  - Manipulated by one processor at a time
  - Often protected by a lock
  - Usually a write causes only a single invalidation
- Synchronization Objects
  - Often more processors imply more invalidations
- Mostly Read
  - More processors imply more invalidations, but writes are rare
- Frequently Read/Written
  - More processors imply more invalidations

SUN Enterprise 6000 Overview

- Up to 30 UltraSPARC processors, MOESI protocol
- Gigaplane™ bus has peak bw 2.67 GB/s, 300 ns latency
- Up to 112 outstanding transactions (max 7 per board)
- 16 bus slots, for processing or I/O boards
  - 2 CPUs and 1GB memory per board
  » memory distributed, but protocol treats as centralized (UMA)

Sun Gigaplane Bus

- Non-multiplexed, split-transaction, 256-data/41-address, 83.5 MHz (Plus 32 ECC lines, 7 tag, 18 arbitration, etc. Total 388)
- Cards plug in on both sides: 8 per side
- 112 outstanding transactions, up to 7 from each board
  - Designed for multiple outstanding transactions per processor
- Snoop result associated with request (5 cycles later)
- Main memory can stake claim to data bus 3 cycles into this, and start memory access speculatively
  - Two cycles later, asserts tag bus to inform others of coming transfer
- MOESI protocol
Outline

- Symmetric Multiprocessors (SMPs) (CC-UMAs)

- Distributed Shared Memory (CC-NUMAs)
  - Challenge
  - Directory Cache Coherence
  - SGI Origin 2000

- More Generally

Large Scale Shared Memory Multiprocessors

- 100s to 1000s of nodes (processors) with single shared physical address space
- Use General Purpose Interconnection Network
  - Still have cache coherence protocol
  - Use messages instead of bus transactions
  - No hardware broadcast
- Communication Assist

Directory Based Cache Coherence

- Avoid broadcast request to all nodes on a miss
  - traffic
  - time
- Maintain directory of which nodes have cached copies of the block (directory controller + directory state)
- On a miss, send message to directory
  - communication assist
- Directory determines what (if any) protocol action is required
  - e.g., invalidation
- Directory waits for protocol actions to finish and then responds to the original request
Directory Example

Distributed Directory
• Distribute Directory among memory modules
• Maintain directory for each memory block
  – memory block = coherence block size: block’s home node = node with directory

Origin2000 System Overview
• Upto 512 nodes (1024 processors)
• Directory state in same or separate DRAMs, accessed in parallel
Origin Cache and Directory States

- Cache states: MESI
- Seven directory states
  - unowned: no cache has a copy, memory copy is valid
  - shared: one or more caches has a shared copy, memory is valid
  - exclusive: one cache (pointed to) has block in modified or exclusive state
  - three pending or busy states, one for each of the above:
    - indicates directory has received a previous request for the block
    - couldn’t satisfy it itself, sent it to another node and is waiting
    - cannot take another request for the block yet
  - poisoned state, used for efficient page migration (later)
- Let’s see how it handles read and “write” requests
  - no point-to-point order assumed in network

Handling a Read Miss

- Hub looks at address
  - If remote, sends request to home
  - If local, looks up directory entry and memory itself
  - Directory may indicate one of many states
- Shared or Unowned State:
  - If shared, directory sets presence bit
  - If unowned, goes to exclusive state and uses pointer format
  - Replies with block to requestor
    - strict request-reply (no network transactions if home is local)
    - Actually, also looks up memory speculatively to get data
    - Directory lookup returns one cycle earlier
    - If directory is shared or unowned, data already obtained by Hub
    - If not one of these, speculative memory access is wasted
- Busy state: not ready to handle
  - NACK, so as not to hold up buffer space for long

Outline

- Symmetric Multiprocessors (SMPs) (CC-UMAs)
- Distributed Shared Memory (CC-NUMAs)
- More Generally:
  - Theory – PRAM
  - Uniform & Non-uniform Memory Access (UMA & NUMA)
  - Aside: Flynn’s Taxonomy: SISD, SIMD, & MIMD
  - Hill’s Alternative View
  - More General Model & Clusters
Outline

• Big Picture
  – Theory – PRAM
  – Uniform Memory Access (UMA)
  – Non-uniform Memory Access (NUMA)
  – Aside: Flynn’s Taxonomy: SISD, SIMD, & MIMD

• Symmetric Multiprocessors (SMPs) (CC-UMAs)
• Distributed Shared Memory (CC-NUMAs)
• Toward Clusters

In Theory

• Sequential
  – Time to sum n numbers? O(n)
  – Time to sort n numbers? O(n log n)
  – What model? RAM

• Parallel
  – Time to sum? Tree for O(log n)
  – Time to sort? Non-trivially O(log n)
  – What model?
    » PRAM [Fortune Willie STOC78]
    » P processors in lock-step
    » One memory (e.g., CREW for concurrent read exclusive write)

Perfection: the PRAM Model

• Parallel RAM
• Fully shared memory
• Unit latency
• No memory contention (unrestricted bandwidth)
• Data placement unimportant
Perfection is NOT Achievable

- Latencies grow as the system size grows
- Bandwidths are restricted by memory organizations and interconnection networks
- Dealing with reality leads to division between
  UMA: Uniform Memory Access
  and
  NUMA: Non-Uniform Memory Access

UMA: Uniform Memory Access

- Latencies are the same, but may be relatively high
- Latencies get worse as system grows
  → scaling difficulties

Uniform Memory Access

- Data placement unimportant
- Typically used in small MP's only
- Contention restricts bandwidth
- Caches are often "allowed" in UMA systems
- Also called symmetric multiprocessors (SMP)
NUMA: NonUniform Memory Access

- Latency low to local memory
- Latency much higher to remote memories
- Performance very sensitive to data placement
- Bandwidth to local memory may be higher
- Contention in network and for memories

![NUMA Diagram](image)

NUMA Multiprocessors, contd.

- Distributed shared memory
  - One logical address space
  - Can be treated as shared memory
- Multicomputers
  - Each processor has its own memory address space
  - Use message passing for communication

![NUMA Multiprocessors Diagram](image)

Flynn’s Taxonomy (Historical) 1 of 3

- SISD: Single Instruction, Single Data
- Operand and instruction storage may be the same
- Your basic uniprocessor

![Flynn’s Taxonomy Diagram](image)
Flynn's Taxonomy (Historical) 2 of 3

- **SIMD**: Single Instruction, Multiple Data
  - Instructions and data storage usually separated
  - Leads to "Data Parallel" programming model
  - Works better for loop-oriented numerical problems
  - Automatic parallelization can work

Flynn's Taxonomy (Historical) 3 of 3

- **MIMD**: Multiple Instruction, Multiple Data
  - More flexible than SIMD and of more interest to us
  - Important for general purpose computing
  - Automatic parallelization more difficult

Hill’s Alternative View

Join At: I/O (Network) Memory Processor

Program With: Message Passing Shared Memory (Dataflow/Systolic), Single-Instruction Single-Data (SIMD) => Data Parallel
A Generic Parallel Machine

- Separation of programming models from architectures
- All models require communication
- Node with processor(s), memory, communication assist

Today's Parallel Computer Architecture

- Extension of traditional computer architecture to support communication and cooperation
  - Communications architecture

User Level
- Multiprogramming
- Shared Memory
- Message Passing
- Data Parallel

Programming Model
- Communication Abstraction

System Level
- Operating System Support
- Hardware
- Physical Communication Medium

A Model

- What's a node?
- What's the network?
- How interface (hardware & software)?
How Interface (Hardware)?

- NI-1 with Ethernet (Cluster)
- NI-1 with special network (IBM SP-2)
- NI-2 with special network (Cray T3E)
- NI-3 with special network (MIT J-Machine) (skip)

Clusters

- PC nodes connected by Ethernet
- Usually programmed with MPI (see next slide)
- MPI Library uses standard TCP/IP
- Advantages
  - Cheap computation cycles
  - Impressive peak rate (e.g., with no communication)
  - Use it if it works for you
- Disadvantages
  - Medium bandwidth
  - Terrible latency
  - Shared-memory not too practical
- Successful! (too successful?)
- Not much (hardware) architecture to discuss

Outline

- Symmetric Multiprocessors (SMPs) (CC-UMAs)
- Distributed Shared Memory (CC-NUMAs)
- More Generally