Scalable Shared-Memory Implementations

Erik Hagersten
Uppsala University
Sweden

Cache-to-cache in snoop-based

Thread
Thread
Thread

A: Read A
... Read A
... Write A

B: Read B
... Read A

My RTS wait for data
Gotta answer

"Upgrade" in dir-based

A: Who has a copy
B: Who has a copy

Inv
Ack
Inv

A: Who has a copy
B: Who has a copy

ReadRequest
ReadDemand

Ack
Forward
Directory-based coherence: per-cacheline info in the memory

Directory-based snooping: NUMA. Per-cacheline info in the home node

Why directory-based
- P2P messages → high bandwidth
- Suits out-of-the-box coherence
- Much more scalable!
- Note:
  - Dir-based can be used to build a uniform-memory architecture (UMA)
  - Bandwidth will be great!!
  - Memory latency will be OK
  - Cache-to-cache latency will not!
  - Memory overhead can be high (storing directory...)

Cache-to-cache in snoop-based
Cache-to-cache in dir-based

- Read Request
- Read Demand
- Ack
- Forward

Thread

Thread

Thread

- Read A
- Read A
- Read A
- Write A

Fully mapped directory

- $k$ Nodes
- Each node is the "home" for $1/k$ of the memory
- Dir entry per cacheline in home memory: $k$ presence-bits + 1 dirty-bit
- Requests are first sent to the home node's CA

Reducing the Memory Overhead: SCI

--- Scalable Coherence Interface (SCI)
- Home only holds pointer to rest of the directory info ($\log(N)$ bits)
- Distributed linked list of copies, weaves through caches
  - Cache tag has pointer, points to next cache with a copy
- On read, add yourself to head of the list (comm. needed)
- On write, propagate chain of invalidations down the list
- On replacement: remove yourself from the list

Cache Invalidation Patterns

- Barnes-Hut Invalidation Patterns
- Radiosity Invalidation Patterns
Overflow Schemes for Limited Pointers

- Broadcast (Dir,B)
  - broadcast bit turned on upon overflow
  - bad for widely-shared invalidated data
- No-broadcast (Dir,NB)
  - on overflow, new sharer replaces one of the old ones (invalidated)
  - bad for widely read data
- Coarse vector (Dir,CV)
  - change representation to a coarse vector, 1 bit per k nodes
  - on a write, invalidate all nodes that a bit corresponds to

cc-NUMA issues

- Memory placement is key!
- Gotta’ migrate data to where it’s being used
- Gotta’ have cache affinity
  - Long time between process switches in the OS
  - Reschedule processor on the CPU it ran last
- SGI Origin 2000’s migration always turned off

Three options for shared memory

- COMA cache-only
- NUMA non-uniform
- UMA uniform (a.k.a. SMP)
Sun’s E6000 Server Family

Erik Hagersten
Uppsala University
Sweden

What Approach to Shared Memory

What Approach to Shared Memory

(a) Shared cache
(b) Bus-based shared memory
(c) Dancehall
(d) Distributed-memory

Looks like a NUMA but drives like a UMA

Physical View
- Memory bandwidth scales with the processor count
- One “interconnect load” per (2xCPU + 2xMem)
- Optimize for the dancehall case (no memory shortcut)

SUN Enterprise Overview

- 16 slots with either CPUs or IO
- Up to 30 UltraSPARC processors (peak 9 GFLOPs)
- Gigaplane™ bus has peak bw 2.67 GB/s; up to 30GB memory
- 16 bus slots, for processing or I/O boards
Enterprize Server E6000

Interconnect

16 boards

An E6000 Proc Board

80 signals = addr, uid, arb, ...

288 signals = 256 data + ECC

An I/O Board

80 signals = addr, uid, arb, ...

288 signals = 256 data + ECC

Split-Transaction Bus

- Split bus transaction into request and response sub-transactions
- Other transactions may intervene
- Improves bandwidth dramatically
- Response is matched to request
- Buffering between bus and cache controllers
Gigaplane Bus Timing

Address
- 0: Rd A
- 1: Rd B
- 2: uid1
- 3: uid2

State
- 0: Share
- 1: Own
- 2: Tag

Arbitration
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14

Tag
- A
- D

Status
- OK
- Cancel

Data
- 1
- 2
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14

Electrical Characteristics of the Bus
- At most 16 electrical loads per signal
- 8 boards from each side (ex. 15 CPU+1 I/O)
- 20.5 inches "centerplane"
- Well controlled impedance
- ~350-400 signals
- Runs at 90/100 MHz
Timing of a single read trans
Board 1 reading from mem 2

<table>
<thead>
<tr>
<th>Address</th>
<th>State</th>
<th>Arbitration</th>
<th>Tag</th>
<th>Status</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>RdA</td>
<td></td>
<td>A</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>RdA, uid1</td>
<td></td>
<td>4,5</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

AVDARK 2010

Protocol tuned for timing

SRAM lookup

<table>
<thead>
<tr>
<th>Address</th>
<th>State</th>
<th>Arbitration</th>
<th>Tag</th>
<th>Status</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>RdA</td>
<td></td>
<td>A</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>RdA, uid1</td>
<td></td>
<td>4,5</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

AVDARK 2010

Foreign and own transactions queue in IQ
State Change on Address Packet

• Data “A” initially resides in CPU7’s cache
• CPU1: Issues a store request to “A”
• CPU1: Read-To-Write req, ID=d, (i.e., “write request”)
• CPU13: LD “A” -> Read-To-Shared req, ID=e
• CPU15: ST “A” -> RTW req , ID=f

mRTO stored in IQCPU1
Own read IQtrans retired when data arrives
Later requests for A queued in IQCPU1 behind mRTO
IQCPU1 will eventually store: <mRTWId, fRTSIDe, fRTWIdf>

AVDARK 2010
A cascade of "write requests"

A initially resides in CPU7's cache

On the bus:
- CPU1: RTW, ID=a
- CPU2: RTW, ID=b
- ...
- CPU5: RTW, ID=f

CPU tags

IQ1 = <mRTW\_Ida, fRTW\_Idb>
IQ2 = <mRTW\_Idb, fRTW\_Idc>
...
IQ5 = <mRTW\_Idf>
...
IQ7 = <fRTW\_Ida>

Snoop tags

Implementing Sun's SunFire 6800

Erik Hagersten
Uppsala University
Sweden

FirePlane, 24 CPUs

L2 cache = 8MB, snoop tags on-chip
CPU 1+GHz UltraSPARC III
Mem= 4+GB/CPU

FirePlane, 24 CPUs

ID = <CPU\#, Uid>
Here it is!!

FirePlane, 24 CPUs

CPU board

Sun’s WildFire System

- Runs unmodified SMP apps in a more scalable way than E6000
- Minor modifications to E6000 snooping required
- CPUs generate local address OR global address
- Global address --> no replication (NUMA)
- Coherent Memory Replication (~Simple COMA@ SICS)
- Hardware support for detecting migration/replication pages
- Directory cache + address translation cache backed by memory
- Deterministic directory implementation (easy to verify)

Sun’s WildFire System

Erik Hagersten
Uppsala University
Sweden

WildFire: One Solaris spanning four nodes
COMA: self-optimizing DSM

ccNUMA

COMA:
- Self-optimizing architecture
- Problem at high memory pressure
- Complex hardware and coherence protocol

Adaptive S-COMA of Large SMPs

- A page may have space allocated in many nodes
- HW maintains memory coherence per cache line
- Replication under SW control -> simple HW (S-COMA)
- Adaptive replication algorithm in OS (R-NUMA)
- Coherent Memory Replication (CMR)
- Hierarchical affinity scheduler (HAS)
- Few large nodes -> simple interconnect and coherence protocol

A WildFire Node

- 16 slots with either CPUs, IO or...
- WildFire extension board
- Up to 28 UltraSPARC processors
- Gigaplane™ bus has peak bw 2.67 GB/s
- Local access time of 330ns (lmbench)

Sun WildFire Interface Board

- SRAM
- ADDR Controller
- Data Buffers
- This space for rent
Sun WildFire Interface Board

WildFire as a vanilla "NUMA"

NUMA -- local memory access

NUMA -- remote memory access
Global Cache Coherence Prot.

NUMA -- local memory access

Gigaplane Bus Timing

WildFire Bus Extensions

Ignore transaction squashes an ongoing transaction => not put in IQ
WildFire eventually reissues the same transaction
RTSF -- a new transaction sends data to CPU and memory
WildFire Directory -- only 4 nodes!!

- k nodes (with one or more procs).
- 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

ReadRequest 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;}

NUMA "detecting excess misses"

Detecting a page for replication

OS Initializes a CMR page
An access to a CMR page

Address Translation (AT) overhead = 8B/8kB = 0.1%
No extra latency added

Access right OK? NO!!

Change MTAG to “shared”

Access right OK? YES!
**Deterministic Directory**

- MOSI protocol, fully mapped directory (one bit/node)
- Directory blocking: one outstanding trans/cache line
- Directory blocks new requests until completion received
- The directory state and cache state always in agreement (except for silent replacement...)

**Replication Issues Revisited**

"Physical" memory

- Controlled by the OS
- ccNUMA
- Only "promising" pages are replicated
- OS dynamically limits the amount of replication
- Solaris CMR changes in the hat_layer (=port)

**Advantages of Multiprocessor Nodes**

**Pros:**
- Amortization of fixed node costs over multiple processors
- Can use commodity SMPs
- Fewer nodes to keep track of in the directory
- Much communication may stay within node (NUCA)
- Can share "node caches" (WildFire: Coherent Memory Replication)

**Cons:**
- Bandwidth shared among processors and interface
- Bus may increase-latency to local memory
- Snoopy bus at remote node increases delays there too

**Memory cost of replication**

- Example: Replicate 10% of data in all nodes
  - 50 nodes, each with 2 CPUs: 490% overhead
  - 4 nodes, each with 25 CPUs: 30% overhead
Does migration/replication help?
NAS parallel Benchmark Study (Execution time in seconds)
[M. Bull, EPCC 2002]

<table>
<thead>
<tr>
<th></th>
<th>BT</th>
<th></th>
<th>SHALLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Initial Plac.</td>
<td></td>
<td>Initial Placement</td>
<td></td>
</tr>
<tr>
<td>No migr</td>
<td>Migr</td>
<td>NoMigr</td>
<td>Migr</td>
</tr>
<tr>
<td>No Repl</td>
<td>26</td>
<td>6.1</td>
<td>6.1</td>
</tr>
<tr>
<td>Repl</td>
<td>72</td>
<td>6.2</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WildFire’s Technology Limits

<table>
<thead>
<tr>
<th>Interconnect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dir $ reach &gt;&gt; sum(cache size)</td>
</tr>
<tr>
<td>Slow interconnect</td>
</tr>
</tbody>
</table>

SRAM size = DRAMsize/256 Snoop frequency

Sun’s SunFire 15k/25k
Erik Hagersten
Uppsala University
Sweden

StarCat
Sun Fire 15k/25k
(used at Lab2)
StarCat, 72 CPUs

Active Backplane
18x18 addr X-bar
18x18 addr X-bar
Expander board
DirS Glob-coh prot Data Rep.
CPU board
Allocate Dir$ entry only for write requests. Speculate on clean data on Dir$ miss

WildCat coherence w/o CMR & w/ faster interconnect

Directory cache, but no directory (broadcast on Dir$ miss)

A: 
B: 
Interconnect

Thread

Directory Protocol
State
Cache access

Cache access

A: 
B: 

StarCat Performance Data

Active Backplane
18x18 addr X-bar
18x18 addr X-bar
Expander board
DirS Glob-coh prot Data Rep.
CPU board

Data Repeater

Lat = 200-340ns
GBW=43GB/s
LBW=86GB/s
Up to 104 CPU
(trading for I/O)