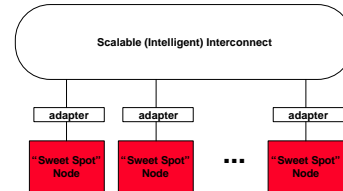


Composing Scalability and Node Design in CC-NUMA

CS 258, Spring 99
David E. Culler
Computer Science Division
U.C. Berkeley

The Composibility Question



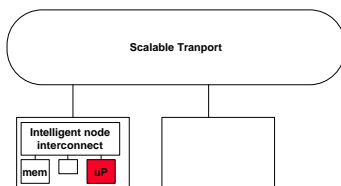
- Distributed address space => issue is NI
- CC Shared address space => composing protocols

4/16/99

CS258 S99

2

“Get the node right” Approach: Origin



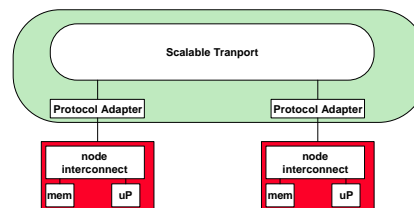
- Basic coherence mechanisms in the node designed to scale
 - manage downward scalability cost
 - still have to adapt to microprocessor protocol domain

4/16/99

CS258 S99

3

“Commodity CC node” approach



- Node speaks a high-level prespecified protocol
- Intelligent Scalable Interconnect
- Sequent NUMA-2000, Data General, HP exemplar, Fujitsu Synergy

4/16/99

CS258 S99

4

Outline

- SGI wrapup
- SCI + Zeon = numaQ 2000

4/16/99

CS258 S99

5

Preserving Sequential Consistency

- R10000 is dynamically scheduled
 - allows memory operations to issue and execute out of program order
 - but ensures that they become visible and complete in order
 - doesn't satisfy sufficient conditions, but provides SC
- An interesting issue w.r.t. preserving SC
 - On a write to a shared block, requestor gets two types of replies:
 - » exclusive reply from the home, indicates write is serialized at memory
 - » invalidation acks, indicate that write has completed wrt processors
 - But microprocessor expects only one reply (as in a uniprocessor system)
 - » so replies have to be dealt with by requestor's HUB
 - To ensure SC, Hub must wait till inval acks are received before replying to proc
 - » can't reply as soon as exclusive reply is received
 - would allow later accesses from proc to complete (writes become visible) before this write

4/16/99

CS258 S99

6

Serialization of Operations

- **Need a serializing agent**
 - home memory is a good candidate, since all misses go there first
- **Possible Mechanism: FIFO buffering requests at the home**
 - until previous requests forwarded from home have returned replies to it
 - but input buffer problem becomes acute at the home
- **Possible Solutions:**
 - let input buffer overflow into main memory (MIT Alewife)
 - don't buffer at home, but forward to the owner node (Stanford DASH)
 - » serialization determined by home when clean, by owner when exclusive
 - » if cannot be satisfied at "owner", e.g. written back or ownership given up, NACKed back to requestor without being serialized
 - serialized when retried
 - don't buffer at home, use busy state to NACK (Origin)
 - » serialization order is that in which requests are accepted (not NACKed)
 - maintain the FIFO buffer in a distributed way (SCI)

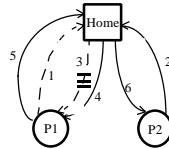
4/16/99

CS258 S99

7

Serialization to a Location (contd)

- **Having single entity determine order is not enough**
 - it may not know when all actions for that operation are done everywhere



1. P1 issues read request to home node for A
 2. P2 issues read-exclusive request to home corresponding to write of A. But won't process it until it is done with read
 3. Home receives 1, and in response sends reply to P1 (and sets directory presence bit). Home now thinks read is complete. Unfortunately, the reply does not get to P1 right away.
 4. In response to 2, home sends invalidate to P1; it reaches P1 before transaction 3 (no point-to-point order among requests and replies).
 5. P1 receives and applies invalidate, sends ack to home.
 6. Home sends data reply to P2 corresponding to request 2.
- Finally, transaction 3 (read reply) reaches P1.

- Home deals with write access before prev. is fully done
- P1 should not allow new access to line until old one "done"

Deadlock

- **Two networks not enough when protocol not request-reply**
 - Additional networks expensive and underutilized
- **Use two, but detect potential deadlock and circumvent**
 - e.g. when input request and output request buffers fill more than a threshold, and request at head of input queue is one that generates more requests
 - or when output request buffer is full and has had no relief for T cycles
- **Two major techniques:**
 - take requests out of queue and NACK them, until the one at head will not generate further requests or output request queue has eased up (DASH)
 - fall back to strict request-reply (Origin)
 - » instead of NACK, send a reply saying to request directly from owner
 - » better because NACKs can lead to many retries, and even livelock

4/16/99

CS258 S99

9

Support for Automatic Page Migration

- **Misses to remote home consume BW and incur latency**
- **Directory entry has 64 miss counters**
 - trap when threshold exceeded and remap page
- **problem: TLBs everywhere may contain old virtual to physical mapping**
 - explicit shutdown expensive
- **set directly entries in old page (old PA) to poison**
 - nodes trap on access to old page and rebuild mapping
 - lazy shutdown

4/16/99

CS258 S99

10

Back-to-back Latencies (unowned)

Satisfied in	back-to-back latency (ns)	hops
L1 cache	5.5	0
L2 cache	56.9	0
local mem	472	0
4P mem	690	1
8P mem	890	2
16P mem	990	3

- measured by pointer chasing since ooo processor

4/16/99

CS258 S99

11

Protocol latencies

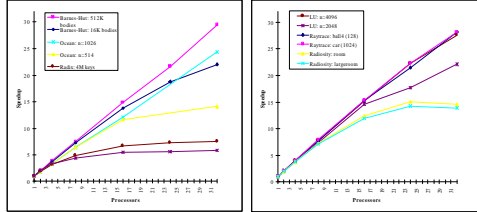
Home	Owner	Unowned	Clean-Exclusive	Modified
Local	Local	472	707	1,036
Remote	Local	704	930	1,272
Local	Remote	472*	930	1,159
Remote	Remote	704*	917	1,097

4/16/99

CS258 S99

12

Application Speedups



4/16/99

CS258 S99

13

Summary

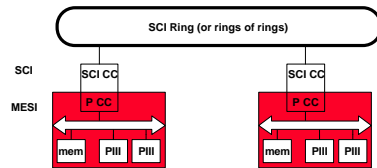
- In directory protocol there is substantial implementation complexity below the logical state diagram
 - directory vs cache states
 - transient states
 - race conditions
 - conditional actions
 - speculation
- Real systems reflect interplay of design issues at several levels
- Origin philosophy:
 - memory-less: node reacts to incoming events using only local state
 - an operation does not hold shared resources while requesting others

4/16/99

CS258 S99

14

Composing Commodity SMPs



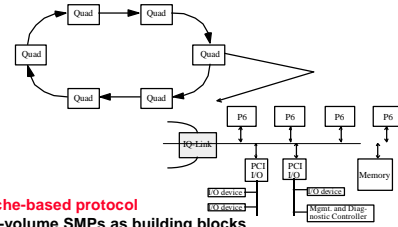
- Key Concepts
 - composing logically disparate protocols
 - caches providing protocol abstraction
 - programming distributed FSMs with data structures
 - towards a 'scalable ready' node
 - » requirements and constraints

4/16/99

CS258 S99

15

NUMA-Q System Overview



- SCI Flat cache-based protocol
- Use of high-volume SMPs as building blocks
- Quad bus is 532MB/s split-transaction in-order responses
 - limited facility for out-of-order responses for off-node accesses
- Cross-node interconnect is 1GB/s unidirectional ring
- Larger SCI systems built out of multiple rings connected by bridges

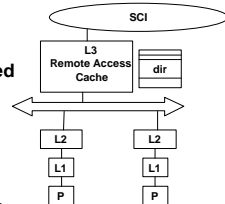
4/16/99

CS258 S99

16

Conceptual Hierarchy

- Remote access cache represents node to SCI protocol
 - directory refers to other RAC
- Only caches blocks fetched from remote homes
- Processor caches kept coherent with remote cache via snoop protocol
- Inclusion preserved between RAC and proc. \$s
- Pseudo proc/pseudo memory of RAC-CC adapts to bus transport

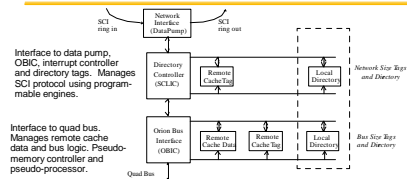


4/16/99

CS258 S99

17

NUMA-Q IQ-Link Board



- Plays the role of Hub Chip in SGI Origin
- Can generate interrupts between quads
- Remote cache (visible to SC I) block size is 64 bytes (32MB, 4-way)
 - processor caches not visible (snoopy-coherent and with remote cache)
- Data Pump (GaAs) implements SCI transport, pulls off relevant packets

4/16/99

CS258 S99

18

NUMA-Q SCI Interconnect

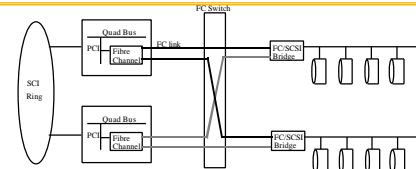
- **Single ring for initial offering of 8 nodes**
 - larger systems are multiple rings connected by LANs
- **18-bit wide SCI ring driven by Data Pump at 1GB/s**
- **Strict request-reply transport protocol**
 - keep copy of packet in outgoing buffer until ack (echo) is returned
 - when take a packet off the ring, replace by “positive echo”
 - if detect a relevant packet but cannot take it in, send “negative echo” (NACK)
 - sender data pump seeing NACK return will retry automatically

4/16/99

CS258 S99

19

NUMA-Q I/O



- Machine intended for commercial workloads; I/O is very important
- Globally addressible I/O, as in Origin
 - very convenient for commercial workloads
- Each PCI bus is half as wide as memory bus and half clock speed
- I/O devices on other nodes can be accessed through SCI or Fibre Channel
 - I/O through reads and writes to PCI devices, not DMA
 - Fibre channel can also be used to connect multiple NUMA-Q, or to shared disk
 - If I/O through local FC fails, OS can route it through SCI to other node and FC

4/16/99

CS258 S99

20

SCI Directory Structure

- **Flat, Cache-based: sharing list is distributed with caches**



- home holds state and pointer to head of sharing list
- sharing lists has head, tail and middle nodes, downstream (fwd) and upstream (bkwd) pointers
- **Directory states (for home mem block)**
 - home: no remote cache
 - fresh: R/O copies in sharing list, mem valid
 - gone: remote cache has writable copy (exclusive or dirty)
- **RAC cache block states (29 states)**
 - position of entry: only, head, mid, tail
 - state of entry: dirty, clean, fresh, copy, pending, ...
- **3 basic operations on the list**

4/16/99 construct, rollout, purge

CS258 S99

21

2-level coherence in NUMA-Q

- directory entries and pointers stored in S-DRAM in IQ-Link board
- remote cache and SCLIC of 4 procs looks like one node to SCI
- SCI protocol does not care how many processors and caches are within node
- keeping those coherent with remote cache is done by OBIC and SCLIC

4/16/99

CS258 S99

22

programming FSMs: read miss

- **Requestor**
 - allocate block entry
 - state: pending
 - start list-construct to add self to head of sharing list
 - » send request to home
- **Home**
 - update state and sets head pointer to requestor
 - fresh no SL:
 - » home replies with data, set fresh w/ SL
 - » req set state to FRESH-ONLY
 - fresh w/ SL:
 - » home replies with data & old head, updates head ptr
 - » req moves to new pending state, sends request to old head
 - » old head moves HEAD_FRESH -> MID_VALID, ONLY_FRESH -> TAIL_VALID, updates back ptr, and replies
 - » req moves to HEAD_FRESH

4/16/99

CS258 S99

23

Read miss (cont)

- **Home: gone**
 - updates head and replies with ptr to old head
 - doesn't know or care about details of the block state
- **req:**
 - new pending state, sends request to old head for data and attach
- **old-head:**
 - respond with data, updates back ptr
 - HEAD_DIRTY -> MID_VALID, ONLY_DIRTY -> TAIL_VALID
- **req:**
 - pending -> HEAD_DIRTY
 - » !!! this was a read miss !!! Can update, but must invalidate SL first
 - » can fetch -gone block into HEAD_DIRTY too

Latency?

CS258 S99

24

What if old head was PENDING?

- NACK and retry (ala SGI Origin)?
- Buffer?
- Building pending list in front of “true head”
 - use the distributed cache state as the buffer
 - retain home ordering

4/16/99

CS258 S99

25

Write Request

- ONLY_DIRTY: OK
- head of sharing list & HEAD_DIRTY
 - sequentially invalidate tail as series of request/response
- HEAD_FRESH
 - request to home to make gone & HEAD_DIRTY, then as above
- not in Sharing list:
 - allocate entry
 - become head, and do as above
- In sharing list, but not head
 - remove self from list
 - » request/response with neighbors
 - do as above

4/16/99

CS258 S99

26

Write-back

- Mid
 - Set pending, Send request to neighbors to patch out
 - What if they are pending?
 - » Priority to tail
- Head
 - request to next
 - update home
 - what if home no longer points back to this node?
 - » Home says retry
 - » eventually new head will try to link to this head, and this head can patch itself out
- general notion of mismatch with protocol state

4/16/99

CS258 S99

27

Order without Deadlock?

- SCI: serialize at home, use distributed pending list per line
 - just like sharing list: requestor adds itself to tail
 - no limited buffer, so no deadlock
 - node with request satisfied passes it on to next node in list
 - low space overhead, and fair
 - But high latency
 - » on read, could reply to all requestors at once otherwise
- Memory-based schemes
 - use dedicated queues within node to avoid blocking requests that depend on each other
 - DASH: forward to dirty node, let it determine order
 - » it replies to requestor directly, sends writeback to home
 - » what if line written back while forwarded request is on the way?

4/16/99

CS258 S99

28

Protocol Interactions

- PII bus split-phase but ‘in-order’
 - adapter waives off request with ‘deferred response’
 - initiates new transaction on response
 - » unfortunately deferred request/response does not update memory, so adapter must take special action
- incoming transactions at home must be serialized with local transactions
 - what’s the serializing agent

4/16/99

CS258 S99

29

Cache-based Schemes

- Protocol more complex
 - e.g. removing a line from list upon replacement
 - » must coordinate and get mutual exclusion on adjacent nodes’ ptrs
 - » they may be replacing their same line at the same time
 - NUMA-Q protocol programmable in firmware
 - » large occupancy
- Higher latency and overhead
 - every protocol action needs several controllers to do something
 - in memory-based, reads handled by just home
 - sending of invals serialized by list traversal
 - » increases latency
 - NUMA-Q: 250 ns local, 2.5us remote
- But IEEE Standard...

4/16/99

CS258 S99

30

Verification

- **Coherence protocols are complex to design and implement**
 - much more complex to verify
- **Formal verification**
- **Generating test vectors**
 - random
 - specialized for common and corner cases
 - using formal verification techniques

4/16/99

CS258 S99

31

Open question

- **Best of both worlds**
 - cost-effective, composable node
 - scalable mode of composition
- **What are the fundamental properties of each?**

4/16/99

CS258 S99

32