From Multicore to Multisocket

John Mellor-Crummey

Department of Computer Science
Rice University

johnmc@rice.edu
Agenda

• Last class
  —organizations and policies for managing on-chip cache on tiled multicore

• Today: glueless multi-chip systems
  —topologies
  —chip to-chip communication
  —multi-chip coherence protocols
Evolution of Commodity Processor Interconnects
A Brief History of Intel Interconnects

- **shared bus (≤ 2004)**
- **dual bus (~2005)**
- **dedicated links (~2007)**
- **QuickPath (2009)**
From Bus to AMD HyperTransport (2006)

Legacy x86 Architecture
- 20-year old traditional front-side bus (FSB) architecture
- CPUs, Memory, I/O all share a bus
- Major bottleneck to performance
- Faster CPUs or more cores ≠ performance

AMD64’s Direct Connect Architecture
- Industry-standard technology
- Direct Connect Architecture reduces FSB bottlenecks
- HyperTransport™ interconnect offers scalable high bandwidth and low latency
- 4 memory controllers – increases memory capacity and bandwidth

Figure credit: P. Conway and B. Hughes, The AMD Opteron CMP NorthBridge Architecture: Now and in the Future, Hot Chips, 2006.
HyperTransport Glueless Interconnect

NorthBridge Components
— crossbar (XBAR)
— system request interface (SRI)
— memory controller (MCT)
— 3 HT ports

12.8 GB/s
128-bit

4.0 GB/s per direction @ 2 GT/s Data Rate

Figure credit: P. Conway and B. Hughes, The AMD Opteron CMP NorthBridge Architecture: Now and in the Future, Hot Chips, 2006.
All cores on the chip share an L3 victim cache.
Diameter vs. Bandwidth

4N SQ (2GT/s HyperTransport)
Diam 2 Avg Diam 1.00
XFIRE BW 14.9GB/s

4N FC (2GT/s HyperTransport)
Diam 1 Avg Diam 0.75
XFIRE BW 29.9GB/s

(2X)

Figure credit: P. Conway and B. Hughes, The AMD Opteron CMP NorthBridge Architecture: Now and in the Future, Hot Chips, 2006.
Link Width vs. Count

• Configurations
  — HT 2: 4 x 16 HT links, fully-connected 4 node
  — HT 3: 8 x 8 HT links, fully-connected 8 node

• Impact
  — reduced network diameter
    - fewer hops to memory
    - lower latency
  - evenly balanced utilization of HT links
    - lower queuing delays
  — increased coherent bandwidth
    - cHT packets visit fewer links

Figure credit: P. Conway and B. Hughes, The AMD Opteron CMP NorthBridge Architecture: Now and in the Future, Hot Chips, 2006.
8-node Topologies

8N TL (2GT/s HyperTransport)
Diam 3 Avg Diam 1.62
XFire BW 15.2GB/s

8N 2x4 (4.4GT/s HyperTransport3)
Diam 2 Avg Diam 1.12
XFire BW 72.2GB/s

8N FC (4.4GT/s HyperTransport3)
Diam 1 Avg Diam 0.88
XFire BW 94.4GB/s (6X)

Note: transfer rates are not all identical

Figure credit: P. Conway and B. Hughes, The AMD Opteron CMP NorthBridge Architecture: Now and in the Future, Hot Chips, 2006.
AMD Northbridge Lessons Learned - I

- Memory latency is the key to application performance
- Better topology is the most direct way to reduce memory latency in multisocket systems

Figure credit: P. Conway and B. Hughes, The AMD Opteron CMP NorthBridge Architecture: Now and in the Future, Hot Chips, 2006.
Intel QuickPath Agents

• Home agent
  — services coherent transactions
    – including handshaking as necessary with caching agents
  — supervises a portion of the coherent memory
  — responsible for managing conflicts that might arise among different caching agents
    – provides the appropriate data and ownership responses as required by a given transaction’s flow

• Caching agent
  — may initiate transactions into coherent memory
  — may retain copies in its own cache structure
  — defined by the messages it may sink and source according to the behaviors defined in the cache coherence protocol
    – can provide copies of the coherent memory contents to other caching agents
Coherence: Cache States on Intel QuickPath

- Five protocol states (MESIF)
  - modified
  - exclusive
  - shared
  - invalid
  - forward: new read-only state to enable cache-to-cache clean line forwarding

<table>
<thead>
<tr>
<th>State</th>
<th>Clean/Dirty</th>
<th>May Write?</th>
<th>May Forward?</th>
<th>May Transition To?</th>
</tr>
</thead>
<tbody>
<tr>
<td>M - Modified</td>
<td>Dirty</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>E - Exclusive</td>
<td>Clean</td>
<td>Yes</td>
<td>Yes</td>
<td>MSIF</td>
</tr>
<tr>
<td>S - Shared</td>
<td>Clean</td>
<td>No</td>
<td>No</td>
<td>I</td>
</tr>
<tr>
<td>I - Invalid</td>
<td>-</td>
<td>No</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>F - Forward</td>
<td>Clean</td>
<td>No</td>
<td>Yes</td>
<td>SI</td>
</tr>
</tbody>
</table>
Coherence Protocols

• Snooping protocols
  — home snoop: home agent is responsible for the snooping of other caching agents
    - basic request sequence involves 4 steps
    - data can be delivered third step
  — source snoop: source of the request can issue both the request and any required snoop messages
    - basic request sequence involves 3 steps
    - two-hop snoop: data can be delivered in second step

• Directory protocol
  — AMD HT Assist
Coherence: Home Snoop 1/4

Assume 4FC topology

**Step 1.**
P1 requests data which is managed by the P4 home agent. (Note that P3 has a copy of the line.)

**Labels:**
- P1 is the requesting caching agent
- P2 and P3 are peer caching agents
- P4 is the home agent for the line

**Precondition:**
P3 has a copy of the line in either M, E or F-state

**Figure credit:** An Introduction to the Intel® QuickPath Interconnect
Step 2.
P4 (home agent) checks the directory and sends snoop requests only to P3.

Figure credit: An Introduction to the Intel® QuickPath Interconnect

Labels:
P1 is the requesting caching agent
P2 and P3 are peer caching agents
P4 is the home agent for the line
Precondition: P3 has a copy of the line in either M, E or F-state
Step 3.
P3 responds to the snoop by indicating to P4 that it has sent the data to P1. P3 provides the data back to P1.

Labels:
P1 is the requesting caching agent
P2 and P3 are peer caching agents
P4 is the home agent for the line
Precondition: P3 has a copy of the line in either M, E or F-state

Figure credit: An Introduction to the Intel® QuickPath Interconnect
Step 4.
P4 provides the completion of the transaction.

Labels:
P1 is the requesting caching agent
P2 and P3 are peer caching agents
P4 is the home agent for the line

Precondition:
P3 has a copy of the line in either M, E or F-state

Figure credit: An Introduction to the Intel® QuickPath Interconnect
Step 1.
P1 requests data which is managed by the P4 home agent. (Note that P3 has a copy of the line.)

Labels:  
P1 is the requesting caching agent  
P2 and P3 are peer caching agents  
P4 is the home agent for the line  
Precondition: P3 has a copy of the line in either M, E or F-state

Figure credit: An Introduction to the Intel® QuickPath Interconnect
Step 2.
P2 and P3 respond to snoop request to P4 (home agent).
P3 provides data back to P1.

Figure credit: An Introduction to the Intel® QuickPath Interconnect

Labels:
- P1 is the requesting caching agent
- P2 and P3 are peer caching agents
- P4 is the home agent for the line
- Precondition: P3 has a copy of the line in either M, E or F-state
Coherence: Source Snoop 3/3

Step 3. P4 provides the completion of the transaction.

Figure credit: An Introduction to the Intel® QuickPath Interconnect

Labels:
- P1 is the requesting caching agent
- P2 and P3 are peer caching agents
- P4 is the home agent for the line

Precondition: P3 has a copy of the line in either M, E or F-state
Allocation of crossbar command buffer across virtual channels can also have a big impact on performance: optimal allocation is $f(# \text{ nodes}, \text{ topology, protocol, txn mix, routing tables})$.

Figure credit: P. Conway and B. Hughes, The AMD Opteron CMP NorthBridge Architecture: Now and in the Future, Hot Chips, 2006.
AMD HT Assist (Istanbul and Beyond)

HT Assist example

Without HT Assist

- CPU 3 request information from CPU 1
- CPU 1 broadcasts to see if another CPU has more recent data
- CPU 3 sits idle while these probes are resolved
- The requested data is sent (9 transactions)

With HT Assist

- CPU 3 request information from CPU 1
- CPU 1 checks its L3 directory to locate the requested data
- CPU 1 finds CPU 2 has the most recent data copy and directly probes CPU 2
- The requested data is sent (4 transactions)

Or

- CPU 3 request information from CPU 1
- CPU 1 checks its L3 directory to locate the requested data
- CPU 1 finds it has the most recent data copy
- The requested data is sent (2 transactions)

Figure credit: http://blogs.zdnet.com/perlow/?p=10187
AMD Istanbul vs. Intel Nehalem

(Nehalem uses 3 memory channels; Istanbul uses 2)

Figure credit: http://www.advancedclustering.com/company-blog/stream-benchmarking.html
Benefits of HT-Assist

• HT Assist benefits 4 to 8 socket configurations

• “On the same 4-socket system, we measured 42GB/s of memory bandwidth with the STREAM benchmark with HT Assist, while only getting 25.5GB/s when HT Assist is disabled.”*

★ 42GB/s using 4 x Six-Core AMD Opteron™ processors ("Istanbul") Model 8435 in Tyan Thunder n4250QE (S4985-E) motherboard, 32GB (16x2GB DDR2-800) memory, SuSE Linux® Enterprise Server 10 SP1 64-bit with HT Assist enabled vs. 25.5GB/s with HT Assist disabled

Source: http://forums.amd.com/blog/blogpost.cfm?threadid=116331&catid=271
Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing
Wanted: Multiprocessors for Scalable Servers

Design goal: scalable parallelism for commercial workloads

- OLTP application characteristics
  - large data footprints
  - low instruction-level parallelism
  - high thread-level parallelism
  - high communication rates

- OLTP performance concerns
  - memory stalls for both instructions and data
  - only small gains from multiple issue, out of order execution, SMT

- CMP seems to offer best opportunity for scalable performance
  - multiple cores for thread-level parallelism
  - simple processors
  - shorter wires
Piranha Overview

- **8 simple cores**
  - single issue
  - I+D L1 caches
- **8 banks shared L2**
  - support fast parallel access
- **8 fast memory channels**
  - 1.6GB/s each
  - 12.8GB/s total

**Figure credit:** L. Barroso et al. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing, ISCA, June 2000.
Piranha Details

• Processor
  — 500 MHz clock
  — 8 stage pipeline: IF, RR, ALU 1-5, WB
  — branch optimization
    – pre-compute logic for branch conditions
    – branch target buffer
  — 64KB, 2-way set-associative, single-cycle L1 cache
  — 2-bit state per cache line for MESI coherence protocol

• Glueless scaling of systems up to 1024 nodes
Piranha L2 Cache and Memory

- **L2 cache**
  - 1MB unified cache
  - instruction + data
  - partitioned into eight banks
    - which processor’s cache does this resemble?
  - no inclusion: L2 misses go to L1
  - L2 is a large victim cache
    - filled only when data in L1 is replaced
    - cooperates with protocol engine for system-wide coherence

- **Memory interface**
  - subservient to L2 cache
  - one memory controller and memory bank for each L2 bank
On-chip Coherence

• L1 miss sent to L2 bank based on address bits

• Possible L2 responses
  — if cache line is present, service request
  — forward to local L1 (owner)
  — obtain data from local memory controller
  — forward request to protocol engine
    - partial information inferred from directory
      if cache line is present on remote node and is exclusively used
Off-chip Coherence

• Coherence engines
  — home engine: export local data
  — remote engine: imports remote data

• Router
  — buffering shared between lanes
  — use “hot potato” routing to reduce buffering
    − enables a message to theoretically reach an empty buffer anywhere in the network
    − buffering requirements grow linearly instead of quadratically with PE
  — “cruise missile invalidate”
    − visits a pre-determined set of nodes
    − receives a single ACK from last node visited
Directory Implementation

- Goal: maintain coherence at node rather than core level
- Problem: full bit vectors cause scaling problems
- Approach: use two representations for scalability
  - 42 bits for encoding sharers
  - limited pointer
    - when data is kept on a four or less nodes, maintain a list
  - coarse vector
    - 1 bit per k nodes: reduces bit-vector space by factor of k
Piranha I/O Chip: A Derivative Design

Figure credit: L. Barroso et al. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing, ISCA, June 2000.
**Figure 3.** Example configuration for a Piranha system with six processing (8 CPUs each) and two I/O chips.

Figure credit: L. Barroso et al. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing, ISCA, June 2000.
## Piranha Analysis: Simulation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Piranha (P8)</th>
<th>Next-Generation Microprocessor (OOO)</th>
<th>Full-Custom Piranha (P8F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Speed</td>
<td>500 MHz</td>
<td>1 GHz</td>
<td>1.25 GHz</td>
</tr>
<tr>
<td>Type</td>
<td>in-order</td>
<td>out-of-order</td>
<td>in-order</td>
</tr>
<tr>
<td>Issue Width</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Instruction Window Size</td>
<td>-</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>Cache Line Size</td>
<td>64 bytes</td>
<td>64 bytes</td>
<td>64 bytes</td>
</tr>
<tr>
<td>L1 Cache Size</td>
<td>64 KB</td>
<td>64 KB</td>
<td>64 KB</td>
</tr>
<tr>
<td>L1 Cache Associativity</td>
<td>2-way</td>
<td>2-way</td>
<td>2-way</td>
</tr>
<tr>
<td>L2 Cache Size</td>
<td>1 MB</td>
<td>1.5 MB</td>
<td>1.5 MB</td>
</tr>
<tr>
<td>L2 Cache Associativity</td>
<td>8-way</td>
<td>6-way</td>
<td>6-way</td>
</tr>
<tr>
<td>L2 Hit / L2 Fwd Latency</td>
<td>16 ns / 24 ns</td>
<td>12 ns / NA</td>
<td>12 ns / 16 ns</td>
</tr>
<tr>
<td>Local Memory Latency</td>
<td>80 ns</td>
<td>80 ns</td>
<td>80 ns</td>
</tr>
<tr>
<td>Remote Memory Latency</td>
<td>120 ns</td>
<td>120 ns</td>
<td>120 ns</td>
</tr>
<tr>
<td>Remote Dirty Latency</td>
<td>180 ns</td>
<td>180 ns</td>
<td>180 ns</td>
</tr>
</tbody>
</table>

Figure credit: L. Barroso et al. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing, ISCA, June 2000.
Piranha Simulation Results

OLTP Speedup and L1 Miss Breakdown

Change in performance vs. number of cores on chip

Note: L2 Fwd = got data from other on-chip L1

Figure credit: L. Barroso et al. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing, ISCA, June 2000.
Multisocket Support on Other Processors
2002: Compaq Alpha EV7

Figure 3. The glueless scalability of the EV7 can support from 4 to 64 processors with ease. The physical address protocol is designed to support up to 128 processors. The processors’ connections wrap around in a 2D torus shape, providing redundancy and shorter interprocessor paths.
EV7 Support for Global Coherence

• Directory-based cache-coherence

• Directory entry for each 64-byte cache block.

• Directory entry tells the user the state of the data block
  — local: DRAM or cache
  — shared: multiple CPUs have copies
    - small number of shares: directory records CPU IDs of the shares
    - large number of shares: directory switches to coarse vectors
      20 bits are used to represent regions in the machine
      64-PE machine: each bit would represent a block of four PE
  — exclusive: a processor has an exclusive copy of the block cached; store the ID of the CPU that cached the block
2013: Oracle Ultrasparc T5

- Glueless 1-hop scaling to eight sockets
- A precise directory tracks all L3s in the system
  - striped across all processors
  - stored in on-chip SRAMs
  - flexible for different socket counts
- Higher BW efficiency than snoop-based protocols enables better scaling
  - 50% more effective bandwidth than comparable snoopy implementation

2013: Oracle Ultrasparc T5

- Each link is 14 lanes wide and runs up to 15Gbps per lane
- Directly connected links minimize latency
- Trunked links achieve more bandwidth in smaller configurations
- Supports single lane failover

Take Away Points

• Systems like Piranha and Compaq EV7 were designed for large scale over a decade ago

• Memory latency is a principal consideration for performance of multisocket systems

• Topology directly influences for memory latency

• Coherence protocol determines how many transactions traverse the interconnect per memory access
  —directories protocols have fewer transactions than snooping
References