Exploration of Memory and Cluster Modes in Directory-Based Many-Core CMPs

Subodha Charles and Prabhat Mishra
University of Florida, USA

Chetan Arvind Patil and Umit Y. Ogras
Arizona State University, USA

This work was partially supported by the National Science Foundation (NSF) grants CNS-1526687 and CNS-1526562
Outline

- Introduction
- Existing NoC Exploration Methods
- Accurate Modeling and Exploration
  - Motivation
  - Modeling of Directory–Memory Traffic
  - Exploration of Memory and Cluster Modes
- Experimental Results
- Conclusion
Increased Complexity of SoC Design
Increased Complexity of SoC Design
NoCs are Critical for Performance

Early interconnection designs were buses and point-to-point

Does Not Scale!

Solution: NoC
# Architecture of a Many-Core CMP

![Diagram of Many-Core CMP Architecture](image)

- **PCU**
  - 0: Core / L1 / L2
  - 1: Core / L1 / L2
  - 2: Core / L1 / L2
  - 3: Core / L1 / L2

- **Memory Controller (MC)**
  - 4: Core / L1 / L2
  - 5: Core / L1 / L2
  - 6: Core / L1 / L2
  - 7: Core / L1 / L2
  - 8: Core / L1 / L2
  - 9: Core / L1 / L2
  - 10: Core / L1 / L2
  - 11: Core / L1 / L2

- **Socket Interface**

- **PCle**

- **MC**
Outline

● Introduction

● **Existing NoC Exploration Methods**

● Accurate Modeling and Exploration
  ❖ Motivation
  ❖ Modeling of Directory–Memory Traffic
  ❖ Exploration of Memory and Cluster Modes

● Experimental Results

● Conclusion
Traffic Optimization on NoC

Min # of MCs
Eitschberger et al.
MCC ‘13

Optimum MC Placement
Xu et al.
CODES+ISSS ‘13

Dynamic Workload Data Mapping
Awasthi et al.
PACT ‘10
# Optimum MC Placement

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
<th>40</th>
<th>48</th>
<th>56</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
<th>40</th>
<th>48</th>
<th>56</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
<th>40</th>
<th>48</th>
<th>56</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Column 0/7**

**Column 2/5**

**Diamond**

**Optimum**

**Xu et al. CODES+ISSS ‘13**

**Slash**
Introduction

Existing NoC Exploration Methods

Accurate Modeling and Exploration
  - Motivation
  - Modeling of Directory–Memory Traffic
  - Exploration of Memory and Cluster Modes

Experimental Results

Conclusion
KNL: 2\textsuperscript{nd} Generation Xeon-Phi

38 tiles
36 active, 2 recovery

Each tile;
2 VPUs, Out of order
4 threads per core

4 separate NoCs
Traffic Model of gem5 Simulator

Life Cycle of a memory request:

(1) Request forwarded to Directory Controller after miss in private cache

(2) Data retrieved from memory

(3) MC forwards data to the requestor
A Memory Controller at Each Tile?

Is this a realistic assumption???

Number of MCs < Number of tiles

- Packaging constraints
- High I/O pin cost

<table>
<thead>
<tr>
<th>Processor</th>
<th># Cores</th>
<th># Memory Controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Xeon Phi 7210 [21]</td>
<td>64</td>
<td>8 MCDRAM &amp; 2 DDR4</td>
</tr>
<tr>
<td>Tilera Tile64 [3]</td>
<td>64</td>
<td>4 DDR2 in 16 ports</td>
</tr>
<tr>
<td>Intel Xeon 8160M</td>
<td>24</td>
<td>2 DDR4, 6 channels</td>
</tr>
<tr>
<td>AMD Opteron 6386 SE</td>
<td>16</td>
<td>1 DDR3, 4 channels</td>
</tr>
</tbody>
</table>
Hotspots Introduced by MCs

(a) MCs modeled at each tile.

(b) MCs modeled only at places marked in blue.
The interactions between cores, directory controllers and memory controllers should be accurately modelled to enable exploration of NoC optimization
Outline

- Introduction
- Existing NoC Exploration Methods
- Accurate Modeling and Exploration
  - Motivation
  - Modeling of Directory–Memory Traffic
  - Exploration of Memory and Cluster Modes
- Experimental Results
- Conclusion
Modified Traffic Model

Life Cycle of a memory request:

1. Request forwarded to Directory Controller after miss in private cache.
2. Forward request to MC.
3. Data retrieved from memory.
4. MC forwards data to the requestor.
Modified Traffic Model

The inclusion of the new step (2) has a significant impact

- Introduces hotspots
- Realistic estimate of power and performance data.
- Exploration of MC placement.
- Exploration of Cluster and Memory modes
Modified Traffic Model
Outline

- Introduction
- Existing NoC Exploration Methods
- Accurate Modeling and Exploration
  - Motivation
  - Modeling of Directory–Memory Traffic
  - Exploration of Memory and Cluster Modes
- Experimental Results
- Conclusion
Cluster Modes in KNL

All-to-all Mode
A request from a core can be forwarded to any directory controller. The memory request can be forwarded to any MC as well.

Quadrant Mode
Four virtual quadrants. A request from a core can be forwarded to any directory controller. But the memory request should be sent to an MC on the same quadrant as the directory.
Memory Modes in KNL

Flat Mode
DDR and MCDRAM in the same address space

Cache Mode
MCDRAM acting as last-level cache
Traffic Flow – Memory and Cluster Modes

Flat, All-to-all Mode

Cache, All-to-all Mode

Flat, Quadrant Mode
Outline

- Introduction
- Existing NoC Exploration Methods
- Accurate Modeling and Exploration
  - Motivation
  - Modeling of Directory–Memory Traffic
  - Exploration of Memory and Cluster Modes
- Experimental Results
- Conclusion
Experimental Setup

- Architecture Simulator: gem5
- NoC model: Garnet2.0
- A CMP similar to Xeon-Phi 7210 modeled in gem5
- Our implementation added in the cache coherence traffic transitions.
- Gem5 output statistics fed into McPAT simulator to extract power results.
Network Traffic Analysis

- The default gem5 model gives highly optimistic results.
- The two modified models – KNL (all-to-all) and KNL (quadrant) gives comparable results.
- KNL (quadrant) gives better performance as it has high affinity between directory and memory controllers.
- Exploration of memory controller placement under the modified model.

- Compared with the work done by Xu et al. “Optimal” is no longer the optimal placement.

- The default gem5 model again gives highly optimistic results.
Memory and Cluster Mode Exploration

- Compared to All-to-all Flat mode, All-to-all Cache mode gives highest benefit: 18.62% less execution time on average.

- Observations are in agreement with results obtained from Xeon Phi 7210 hardware platform.
Conclusion

Memory and Cluster Mode Exploration

- Compared to All-to-all Flat mode, All-to-all Cache mode gives highest benefit: 18.62% less execution time on average

- Observations are in agreement with results obtained from Xeon Phi 7210 hardware platform
Thank you!

Questions?