Memory System Performance in a NUMA Multicore Multiprocessor

Zoltan Majo and Thomas R. Gross

Department of Computer Science
ETH Zurich
Summary

• NUMA multicore systems are unfair to local memory accesses

• **Local execution** sometimes suboptimal
Outline

• NUMA multicores: how it happened

• Experimental evaluation: Intel Nehalem

• Bandwidth sharing model

• The next generation: Intel Westmere
NUMA multicores: how it happened

First generation: SMP

Total bandwidth [GB/s]

Active cores

DRAM memory

Northbridge

MC

BusC

BusC

BusC

BusC

0 1 2 3

4 5 6 7

SMP
NUMA multicore: how it happened

Next generation: NUMA

0 1 2 3
BusC IC

4 5 6 7
IC BusC

Northbridge

MC

DRAM memory

Total bandwidth [GB/s]

Active cores
NUMA multicores: how it happened

Next generation: NUMA

![Diagram of NUMA architecture with arrows indicating communication between memory blocks]

Total bandwidth [GB/s]

- SMP
- NUMA (local)

Active cores

1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7

DRAM memory

DRAM memory
NUMA multicores: how it happened
Bandwidth sharing

• Frequent scenario:
  bandwidth shared between cores

• **Sharing model** for the Intel Nehalem
Outline

• NUMA multicores: how it happened

• Experimental evaluation: Intel Nehalem

• Bandwidth sharing model

• The next generation: Intel Westmere
Evaluation system

Processor 0

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 3 cache</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global Queue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC</td>
<td>QPI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Processor 1

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 3 cache</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global Queue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QPI</td>
<td>MC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Intel Nehalem E5520
- 2 x 4 cores
- 8 MB level 3 cache
- 12 GB DDR3 RAM
- 5.86 GT/s QPI
Bandwidth sharing: local accesses
Bandwidth sharing: remote accesses
Bandwidth sharing: combined accesses
Global Queue

• Mechanism to *arbitrate* between different types of memory accesses

• We look at *fairness* of the Global Queue:
  – *local* memory accesses
  – *remote* memory accesses
  – *combined* memory accesses
Benchmark program

- STREAM **triad**

```c
for (i=0; i<SIZE; i++)
    {
        a[i]=b[i]+SCALAR*c[i];
    }
```

- Multiple co-executing triad **clones**
Multi-clone experiments

• All memory allocated on Processor 0

• Local clones: \( C \)  Remote clones: \( C \)

• Example benchmark configurations:

  \[
  \begin{array}{c}
  \text{(2L, 0R)} \\
  \begin{array}{ccc}
  & C & C \\
  \end{array}
  \\
  \text{Processor 0}
  \\
  \end{array}
  \hspace{1cm}
  \begin{array}{ccc}
  \text{(0L, 3R)} \\
  \begin{array}{ccc}
  C & C & C \\
  \end{array}
  \\
  \text{Processor 1}
  \end{array}
  \hspace{1cm}
  \begin{array}{ccccc}
  \text{(2L, 3R)} \\
  \begin{array}{cccc}
  C & C & C & C \\
  \end{array}
  \\
  \text{Processor 0}
  \hspace{0.5cm}
  \text{Processor 1}
  \end{array}
  \]
GQ fairness: local accesses

![Diagram of processor configurations and total bandwidth](image)
GQ fairness: remote accesses

![Diagram showing processor configurations and bandwidth comparison](image)

**Total bandwidth [GB/s]**

**Benchmark configurations**
- Core 0
- Core 1
- Core 2
- Core 3
Global Queue fairness

• Global Queue **fair** when there are **only local/remote** accesses in the system

• What about **combined accesses**?
GQ fairness: combined accesses

Execute clones in **all possible** configurations

<table>
<thead>
<tr>
<th># local clones</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td># remote clones</td>
<td></td>
<td></td>
<td>(2L, 3R)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GQ fairness: combined accesses

Execute clones in **all possible** configurations

<table>
<thead>
<tr>
<th></th>
<th># local clones</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># remote clones</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
GQ fairness: combined accesses

Total bandwidth [GB/s]

Benchmark configurations

local clones  remote clones
GQ fairness: combined accesses

Execute clones in **all possible** configurations

<table>
<thead>
<tr>
<th># local clones</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td># remote clones</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Combined accesses

![Combined accesses diagram]

- Total bandwidth [GB/s]

- (1L,1R), (2L,1R), (3L,1R), (4L,1R)

- remote clone
- local clone 1
- local clone 2
- local clone 3
- local clone 4
Combined accesses

- In configuration *(4L, 1R)* remote clone gets *30% more bandwidth* than a local clone

- Remote execution can be better than local
Outline

• NUMA multicores: how it happened

• Experimental evaluation: Intel Nehalem

• Bandwidth sharing model

• The next generation: Intel Westmere
Bandwidth sharing model
Sharing factor ($\beta$)

• Characterizes the **fairness** of the Global Queue

• Dependence of sharing factor on **contention**?
Contention affects sharing factor
Contestion affects sharing factor

![Graph showing the relationship between additional contention and sharing factor.](image-url)
Combined accesses

![Combined accesses Chart]

- Total bandwidth [GB/s]
- Remote clone
- Local clone 1
- Local clone 2
- Local clone 3
- Local clone 4

Legend:
- Remote clone
- Local clone 1
- Local clone 2
- Local clone 3
- Local clone 4

Chart Description:
- (1L,1R)
- (2L,1R)
- (3L,1R)
- (4L,1R)
Contestation affects sharing factor

• Sharing factor decreases with contention

• With local contention remote execution becomes more favorable
Outline

- NUMA multicores: how it happened
- Experimental evaluation: Intel Nehalem
- Bandwidth sharing model
- The next generation: Intel Westmere
The next generation

Intel Westmere X5680
2 x 6 cores
12 MB level 3 cache
144 GB DDR3 RAM
6.4 GT/s QPI
The next generation

Total bandwidth [GB/s]

Benchmark configurations

- (1L, 1R)
- (2L, 1R)
- (3L, 1R)
- (4L, 1R)
- (5L, 1R)
- (6L, 1R)

- remote clone
- local clone 1
- local clone 2
- local clone 3
- local clone 4
- local clone 5
- local clone 6
Conclusions

• Optimizing for **data locality** can be **suboptimal**

• Applications:
  
  – OS scheduling (see ISMM’11 paper)
  
  – data placement and computation scheduling
Thank you! Questions?