Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems

Mohammad Dashti\textsuperscript{1}, Alexandra Fedorova\textsuperscript{1}, Justin Funston\textsuperscript{1}, Fabien Gaud\textsuperscript{1}, Renaud Lachaize\textsuperscript{2}, Baptiste Lepers\textsuperscript{3}, Vivien Quéma\textsuperscript{4}, Mark Roth\textsuperscript{1}

\textsuperscript{1}Simon Fraser University
\textsuperscript{2}Université Joseph Fourier
\textsuperscript{3}CNRS
\textsuperscript{4}Grenoble INP

March 19, 2013
New multicore machines are NUMA
Well-know issue: remote access latency overhead

- Impacts performance by at most 30%
New issue: Memory controller and interconnect congestion
Current solutions

- Try to improve locality
  - Thread scheduling and page migration (USENIX ATC’11)
  - Thread Clustering (EuroSys’07)
  - Page replication (ASPLOS’96)
  - Etc.

- But the main problem is MC/interconnect congestion
MC/Interconnect congestion impact on performance

► 16 threads, one per core
► Memory either allocated on first touch or interleaved

Example: Streamcluster

First touch scenario

Interleave scenario
MC/Interconnect congestion impact on performance (2)

Performance difference (%) between best and worst policy

- Best policy is First Touch
- Best policy is Interleaving

Up to 100% performance difference
Why do applications benefit from interleaving? (1)

<table>
<thead>
<tr>
<th></th>
<th><strong>Interleaving</strong></th>
<th><strong>First touch</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Local access ratio</td>
<td>25%</td>
<td>25%</td>
</tr>
<tr>
<td>Memory latency (cycles)</td>
<td>471</td>
<td>1169</td>
</tr>
<tr>
<td>Memory controller imbalance</td>
<td>7%</td>
<td>200%</td>
</tr>
<tr>
<td>Interconnect imbalance</td>
<td>21%</td>
<td>86%</td>
</tr>
<tr>
<td>Perf. improvement / first touch</td>
<td>105%</td>
<td>-</td>
</tr>
</tbody>
</table>

⇒ Interconnect and memory controller congestion drive up memory access latency
Why do applications benefit from interleaving? (2)

<table>
<thead>
<tr>
<th>PCA</th>
<th>Interleaving</th>
<th>First touch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local access ratio</td>
<td>25%</td>
<td>33%</td>
</tr>
<tr>
<td>Memory latency (cycles)</td>
<td>480</td>
<td>665</td>
</tr>
<tr>
<td>Memory controller imbalance</td>
<td>4%</td>
<td>154%</td>
</tr>
<tr>
<td>Interconnect imbalance</td>
<td>19%</td>
<td>64%</td>
</tr>
<tr>
<td>Perf. improvement / first touch</td>
<td>38%</td>
<td>-</td>
</tr>
</tbody>
</table>

⇒ Balancing load on memory controllers is more important than improve locality
Conclusions

- **Balance is more important than locality**
  - Memory controller and interconnect congestion can drive up access latency

- **Always manually interleaving memory is NOT the way to go**

⇒ **Need a new solution**
Carrefour: A new memory traffic management algorithm

- First goal: balance memory pressure on interconnect and MC
- Second goal: improve locality
Mechanism #1: Page relocation
Mechanism #1: Page relocation

- Better locality
- Lower interconnect load
- Balanced load on MC

- Cannot be applied if region is shared by multiple threads
Mechanism #2: Page replication

- Better locality
- Lower interconnect load
- Balanced load on MC

- Higher memory consumption
- Expensive synchronization

Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems
Mechanism #2: Page replication

- Better locality
- Lower interconnect load
- Balanced load on MC

- Higher memory consumption
- Expensive synchronization
Mechanism #3: Page interleaving

Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems
Mechanism #3: Page interleaving

- Balanced load on interconnect
- Balanced load on MC
- Can decrease locality
Carrefour in details

Goal: Combine these techniques to:
1. Balance memory pressure
2. Increase locality
Accurate and low-overhead page access statistics

- Adaptive IBS sampling
- Include cache accesses
- Use hardware counter feedback
Carrefour in details

Per application profiling

Global application metrics
- Memory intensity
- Memory imbalance
- Local access ratio
- Memory read ratio

Memory congestion?
- Enable migrations?
- Enable interleaving?
- Enable replications?

Migrate / Interleave / Replicate page

Expensive!

Per page metrics
- RW ratio
- Accessed by nodes

Expensive!

Efficient page replication
- Use a careful implementation (fine grain locks)
- Prevent data synchronization
Evaluation

- Carrefour is implemented in Linux 3.6
- Machines
  - 16 cores, 4 nodes, 64 GB of RAM
  - 24 cores, 4 nodes, 64 GB of RAM
- Benchmarks (23 applications)
  - Parsec
  - FaceRec
  - Metis (Map/Reduce)
  - NAS
- Compare Carrefour to
  - Linux (default)
  - Linux Autonuma
  - Manual Interleaving
Carrefour significantly improves performance!
Carrefour overhead

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Maximum overhead / default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonuma</td>
<td>25%</td>
</tr>
<tr>
<td>Carrefour</td>
<td>4%</td>
</tr>
</tbody>
</table>

- Carrefour average overhead when no decision are taken: 2%
Conclusion

- In modern NUMA systems:
  - Remote latency overhead is not the main bottleneck
  - MC and interconnect congestion can drive up memory latency

- **Carrefour**: a memory traffic management algorithm
  - First goal: balance memory pressure on interconnect and MC
  - Second goal: improve locality

- **Performance**: 
  - Improves performance significantly (up to 270%)
  - Outperforms others solutions
Questions?

https://github.com/Carrefour
Carrefour supports multi-applications workloads

![Performance improvement with respect to Linux (%)](chart)

- **MG** Streamscluster
- **PCA** Streamscluster
- **FaceRecLong** Streamscluster

**Legend:**
- **AutoNUMA**
- **Manual interleaving**
- **Carrefour**
Detailed profiling

Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems
Energy consumption

Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems