Designing Efficient Asynchronous Memory Copy Operations Using Hardware Copy Engine: A Case Study with I/OAT

K. Vaidyanathan, W. Huang, L. Chai and D. K. Panda

Network Based Computing Laboratory (NBCL)
The Ohio State University
Presentation Outline

• Introduction and Motivation

• Design

• Experimental Results

• Conclusions and Future Work
Introduction

• Increasing demand from science and engineering applications
  – Weather forecasting, biomedical analysis, etc
• High Computational Requirements
  – Emergence of Multi-core systems
• High Communication Requirements
  – High-performance interconnects such as InfiniBand, 10-Gigabit Ethernet
• High Memory Requirements
  – Wider and faster connection to memory (FB-DIMMs)
  – But, huge gap between memory and CPU performance
Issues with Memory Operations

• CPU stalling issues
• Cache pollution issues
• Memory contention issues
• Performance

→ Ability to overlap computation and memory operation as a memory latency-hiding mechanism is critical

→ Need for Hardware Engine to perform operations directly in memory (e.g. Hardware Copy Engine)
Motivations for Hardware Copy Engine

Copy using CPU
- Register-based Stall on Memory
- Critical Resources Touched

Copy using DMA Engine
- CPU not stalled
- Compute overlap

Hardware Copy Engine
- Reduction in CPU resources and Better Performance for large memory copy operations
- Computation-Memory Operation Overlap
- Avoiding Cache Pollution Effects
Can we use traditional DMA Engines for Memory Copies?

• Direct Memory Access Engines can be used
  – Huge Startup Costs
  – Huge Completion notification costs
  – Applicable only for large message copies

• Intel’s I/O Acceleration Technology (I/OAT)
  – On-chip Asynchronous DMA Copy Engine (ADCE)
    • Alleviates the startup and completion notification costs
    • Can be used for medium message copies
  – However, available only in kernel space
Problem Statement

Can we design an efficient asynchronous memory copy operation using an Asynchronous DMA Copy Engine?

Can we provide an API for user space applications to use such asynchronous memory copy operations?
Presentation Outline

• Introduction and Motivation

• Design

• Experimental Results

• Conclusions and Future Work
Basic Design

- **adma_copy(src, dst, len)** – Blocking copy routine
- **adma_icopy(src, dst, len)** – Non-blocking copy routine
- **adma_check_copy(cookie)** – check for copy completion
- **adma_wait_copy(cookie)** – wait for copy completion
ADCE-Based IPC Design

- `adma_iread(fd, dst, len)` – Non-blocking read routine
- `adma_iwrite(fd, dst, len)` – Non-blocking write routine
- `adma_check(fd, cookie)` – check for completion
- `adma_wait(fd, cookie)` – wait for completion
Other Design Issues

• Handling IPC Synchronization
  – Use semaphores

• Handling Memory Alignment
  – Can have performance hit

• Page Pinning/Locking
  – get_user_pages() & put_page()
  – Can be costly if the page is not resident in memory

• Scheduling across different DMA channels
  – Currently support round robin
  – Can support priority-based memory operations
OSU MVAPICH

- High performance implementation of MPI-1 and MPI-2
- Open-source with BSD Licensing
- Available in Open Fabrics Distribution ([http://www.openfabrics.org](http://www.openfabrics.org))
- Used by over 465 top organizations in the world in over 30 countries
- IPC design integrated into the MVAPICH stack
  - Compared the native shared memory implementation with the ADCE approach
Presentation Outline

• Introduction and Motivation

• Design

• Experimental Results

• Conclusions and Future Work
Experimental Results

• Experimental Test-bed
  – Dual Dual-core Intel 3.46 GHz processors
  – 2 MB L2 Cache and 4 GB memory
  – Linux RedHat AS4, kernel version 2.6.9-30

• Experimental Outline
  – Microbenchmarks
    • Performance
    • Overlap
    • Cache Pollution Effects
  – MPI Applications
    • IS and PSTSWM
Memory Copy Latency for Small Messages

- For hot-cache experiment, CPU-based copy outperforms ADCE-based copy
- For cold-cache experiment, ADCE-based copy performs better (> 16KB)
Large Message Memory Copy: Latency and Bandwidth

- ADCE results in 50% improvement for large messages
Overlap of Memory Copy with Computation

• ADCE achieves close to 90% overlap if buffers are cached and 70% overlap if buffers are not cached
Split-up Overhead of ADCE

- Page pinning consumes a lot of overhead (up to 30%) for large messages
Cache Pollution Effects

- ADCE-based copy does not affect the access latency
IPC Latency and Bandwidth

- ADCEC results in 50% improvement for large messages
Impact with MPI Applications

- ADCE-based approach shows up to 12% improvement with applications
## Overall Impact

<table>
<thead>
<tr>
<th></th>
<th>Message Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Small</td>
</tr>
<tr>
<td>Cold-Cache Performance</td>
<td>✗</td>
</tr>
<tr>
<td>Hot-Cache Performance</td>
<td>✗</td>
</tr>
<tr>
<td>Overlap</td>
<td>✗</td>
</tr>
<tr>
<td>Cache Pollution</td>
<td>✓</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

• Proposed a design for asynchronous memory copy operations for user-space applications using Intel’s copy engine
• Improved performance, overlap computation with memory operation and scope for intelligent cache management
• Need to evaluate applications from several domains
• Scope for reducing the overheads further using multi-core architectures
Acknowledgements

Our research is supported by the following organizations

• Current Funding support by
  - Office of Science
  - NSF
  - Mellanox Technologies
  - Cisco Systems
  - Intel
  - Linux
  - NetApp

• Current Equipment support by
  - Mellanox Technologies
  - AMD
  - Sun Microsystems
  - APPR
  - IBM
  - Microway
  - PathScale
  - SilverStorm
Web Pointers

Group Homepage: http://nowlab.cse.ohio-state.edu

Emails: {vaidyana, huanwei, chail, panda}@cse.ohio-state.edu