Improving RDMA-based MPI Eager Protocol for Frequently-used Buffers

Mohammad J. Rashti and Ahmad Afsahi
Department of Electrical and Computer Engineering
Queen’s University
Kingston, ON, Canada

Workshop on Communication Architecture for Clusters
Presentation Outline

- Introduction
- Motivation
- Frequent-buffer Eager Communication
  - Proposed Eager Communication Mechanism
  - Detection of Frequently-used Buffers
  - Protocol Adaptation
- Micro-benchmark Results
- Application Results
- Conclusions and Future Research
Introduction

- MPI is the main standard for communication in HPC clusters.
- MPI implementations over modern RDMA-enabled interconnects (e.g., InfiniBand) provide features such as:
  - Operating system bypass
  - Zero-copy data transfer
  - Lower CPU utilization

- MPI implementations (e.g., MPICH2, MVAPICH2) use
  - **Eager** protocol for small messages
    - Send the data eagerly to avoid the pre-negotiation cost
  - **Rendezvous** protocol for large messages
    - Negotiate with the receiver before sending the data
Introduction

- **RDMA-based communication**
  - **Buffer registration** is required prior to communication
    - Costly due to buffer pin-down and address translation
    - Registration tag needs to be advertised
      - This is why in the Eager protocol small messages are copied from application buffers into *pre-registered* and *pre-advertised* intermediate buffers.

- Buffer copy in the Eager protocol is based on the assumption that application buffers are not used frequently.
  - What if some user buffers are frequently used?

  **Can we bypass the buffer copy in the Eager protocol for frequently-used buffers?**
Presentation Outline

- Introduction
- Motivation
- Frequent-buffer Eager Communication
  - Proposed Eager Communication Mechanism
  - Detection of Frequently-used Buffers
  - Protocol Adaptation
- Micro-benchmark Results
- Application Results
- Conclusions and Future Research
Motivation

- Do MPI applications reuse their data buffers at runtime?

**MPI applications under study**

<table>
<thead>
<tr>
<th>MPI Application</th>
<th>Field</th>
<th>Communication Primitives</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPB CG (Class C)</td>
<td>Sparse linear systems solver</td>
<td>MPI send/recv</td>
</tr>
<tr>
<td>NPB LU (Class C)</td>
<td>Navier Stockes equation solver</td>
<td>MPI send/recv, MPI barrier</td>
</tr>
<tr>
<td>ASC AMG2006</td>
<td>Algebraic multi-grid solver</td>
<td>MPI send/recv, MPI broadcast, barrier</td>
</tr>
<tr>
<td>SPEC MPI2007</td>
<td>Simulated SU(3) lattice gauge theory</td>
<td>MPI send/recv, MPI broadcast and all-reduce, all-to-all and barrier</td>
</tr>
</tbody>
</table>
Motivation

- Experimental platform
  - Hosts
    - 4 Dell PowerEdge 2850 SMP servers with two dual-core Intel Xeon 2.8GHz processors, 4GB of DDR-2 SDRAM and an x8 PCI-Express slot
  - Network
    - Mellanox ConnectX DDR HCA and 24-port Infiniscale III switch
  - Software
    - MVAPICH2 1.0.3 over OFED 1.3
    - Fedora Core 5, Kernel 2.6.20
Eager buffer reuse statistics for MPI applications (16 Processes)

<table>
<thead>
<tr>
<th>MPI Application</th>
<th>Buffer Reuse Count</th>
<th>Most Frequently-used Buffer Sizes (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPB CG (Class C)</td>
<td>7904 – 7904</td>
<td>8 – 8</td>
</tr>
<tr>
<td></td>
<td>(Average: 7904)</td>
<td>(Average: 8)</td>
</tr>
<tr>
<td>NPB LU (Class C)</td>
<td>3749 – 3750</td>
<td>1560 – 1640</td>
</tr>
<tr>
<td></td>
<td>(Average: 3749)</td>
<td>(Average: 1600)</td>
</tr>
<tr>
<td>ASC AMG2006</td>
<td>45 – 292</td>
<td>8 – 3648</td>
</tr>
<tr>
<td></td>
<td>(Average: 119)</td>
<td>(Average: 770)</td>
</tr>
<tr>
<td>SPEC MPI2007 104.MILC</td>
<td>2 – 5010</td>
<td>8 – 4800</td>
</tr>
<tr>
<td></td>
<td>(Average: 2049)</td>
<td>(Average: 2876)</td>
</tr>
</tbody>
</table>
Motivation

- How much improvement can we theoretically achieve if we remove the send-side buffer copy for frequently-used buffers?

Eager/Rendezvous switching point is \( \sim 9\text{KB} \)

Theoretical improvement on our platform
Presentation Outline

- Introduction
- Motivation
- Frequent-buffer Eager Communication
  - Proposed Eager Communication Mechanism
  - Detection of Frequently-used Buffers
  - Protocol Adaptation
- Micro-benchmark Results
- Application Results
- Conclusions and Future Research
Proposed Eager Communication Mechanism

How can we capitalize on such a potential where user buffers may be frequently used?

- If a user buffer is used frequently then we can register it to avoid the data copy.
- This way, we can initiate the RDMA operations directly from the application buffer, and decrease the cost of communication by skipping the send-side data copy.

- The registered buffer can be kept in the registration cache of MPI and later on be retrieved in subsequent references to the same buffer.
- The cost of one-time registration is amortized over the cost of multiple copies.
Proposed Eager Communication Mechanism

Receiver copy cannot be avoided due to the Eager communication semantics.
Detection of Frequently-used Buffers

- How do we detect the frequently-used buffers?
  - We keep track of the application buffer usage statistics.
    - When a buffer is accessed for an Eager transfer, the buffer is searched in a buffer usage table (a hash table) containing buffer usage statistics.
    - If the buffer is found in the table and its usage statistics is greater than a threshold, then we check for the buffer in the MPI registration cache (for a possible previous registration).
      - Register the buffer if it is not found in the registration cache.
Detection of Frequently-used Buffers

Which buffer is considered frequently-used?

- Find the minimum buffer reuse threshold, \( n \), for the new protocol to be beneficial.

Let:

- \( C_m \) be the copy cost for a single message with size \( m \)
- \( NT_m \) be the network transfer time
- \( R_m \) be the registration cost
- \( V \) be the implementation overhead (total search time for the buffer usage table and the registration cache)

\[
T_{\text{current}} = n \times (2 \times C_m + NT_m) \\
T_{\text{new}} = n \times (C_m + NT_m + V) + R_m
\]

\[
T_{\text{new}} < T_{\text{current}} \quad \Rightarrow \quad n > \frac{R_m}{C_m - V}
\]
Detection of Frequently-used Buffers

\[ n > \frac{R_m}{C_m - V} \]

\( n \) is negative for \( m \leq 64 \), as \( V > C_m \)

Our Eager protocol dynamically decides when to register a buffer, based on the minimum buffer reuse threshold value, \( n \).

- However, this decision making is done “speculatively” in the sense that it registers the buffer when it is reused by at least a certain portion of the threshold value, \( n \), (25%, 50%, etc.) hoping that the buffer will be reused for more than later on.

<table>
<thead>
<tr>
<th>Message Size (bytes):</th>
<th>Minimum n</th>
<th>Message Size (bytes):</th>
<th>Minimum n</th>
</tr>
</thead>
<tbody>
<tr>
<td>128B : 1280</td>
<td></td>
<td>2KB : 109</td>
<td></td>
</tr>
<tr>
<td>256B : 755</td>
<td></td>
<td>4KB : 59</td>
<td></td>
</tr>
<tr>
<td>512B : 432</td>
<td></td>
<td>8KB : 33</td>
<td></td>
</tr>
<tr>
<td>1KB : 200</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Minimizing the Implementation Overhead

- We have minimized the implementation overhead by using a hash table rather than a binary tree for the buffer reuse statistics.

![Search Time Comparison Graph](attachment:search_time_graph.png)
Protocol Adaptation

- Registering the (Eager-size) user buffers is useful for applications with high buffer reuse. **What if an application has low buffer reuse?**
  - While we won’t incur the cost of buffer registration, the overhead of manipulating buffer usage statistics may affect the performance, especially for small messages.

- We adaptively stop adding more buffers to the buffer usage table when the following two conditions are satisfied:
  - When the overall number of (registered) buffers in the table that are marked as frequently-used is less than 20% (a typical value) of all buffers in the table
  - When the hash table has started to grow linked list (due to linear search/insertion costs)
Presentation Outline

- Introduction
- Motivation
- Frequent-buffer Eager Communication
  - Proposed Eager Communication Mechanism
  - Detection of Frequently-used Buffers
  - Protocol Adaptation
- Micro-benchmark Results
- Application Results
- Conclusions and Future Research
Ping-Pong Latency

- Latency results are averaged across 1000 ping-pong operations. Two separate send and receive buffers are used in each process.

Close to theoretical maximum improvement
Spectrum Buffer-reuse Latency

- To simulate an application with different buffer reuse patterns: a 1000-unit buffer set, \( \text{buf}_i, 1 < i < 1000 \), in which \( \text{buf}_i \) is used \( i \) times. All buffers are of the same length.

![Graph showing new method improvement in spectrum buffer usage micro-benchmark.](image)

Showing the worst-case scenario.

MVAPICH2 registers buffers in one-page chunks, 4KB in our system.
Collective Communication Results

- Collectives implemented on top of point-to-point messages will benefit implicitly from our proposed technique.
  - Broadcast/scatter use a binomial tree algorithm.

![Collective Performance Improvement Graph]

- (4 x 1) Higher performance than point-to-point
- (4 x 4) Lower performance due to shared-memory intra-node communication
- Sending a combined message of more than 9KB size
Presentation Outline

- Introduction
- Motivation
- Frequent-Buffer Eager Communication
  - Proposed Eager Communication Mechanism
  - Detection of Frequently-used Buffers
  - Protocol Adaptation
- Micro-benchmark Results
- Application Results
- Conclusions and Future Research
Impact on MPI Applications

- These MPI applications have been chosen because of their different buffer reuse characteristics.

<table>
<thead>
<tr>
<th>MPI Application</th>
<th>Most Frequently-used Buffer Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPB CG (Class C)</td>
<td>0%</td>
</tr>
<tr>
<td>NPB LU (Class C)</td>
<td>32.29%</td>
</tr>
<tr>
<td>ASC AMG2006</td>
<td>9%</td>
</tr>
<tr>
<td>SPEC MPI2007</td>
<td>22.25%</td>
</tr>
<tr>
<td>104.MILC</td>
<td></td>
</tr>
</tbody>
</table>

LU and MILC: modest improvement: high buffer reuse and relatively larger Eager messages
CG: high buffer reuse, but small messages (8 bytes; our technique is essentially disabled)
AMG2006: buffer reuse statistics are very close to the speculative threshold point where we start to register buffers, suffering from the overhead of registering not sufficiently reused buffers. However, adaptation is activated to reduce the overhead on this application.
Conclusions

- We proposed a novel method to improve the MPI Eager protocol over RDMA-enabled networks, by avoiding the send-side buffer copy for frequently-used buffers and instead transfer data directly from the application buffer.

- Up to 14% and 19% improvements for pt-2-pt and collectives were observed, respectively.

- Applications with high buffer reuse statistics benefit from the proposed technique.
  - For the applications, process synchronization and process skew time, communication/computation overlap and communication/computation ratio may be the deciding factors in achieving the total improvement.
Future Research

- We would like to run our applications on a larger testbed.
  - Larger node count may prove to help having a larger improvement

- We would like to extend our work to:
  - Collective communications
  - user-level send/recev-based communication model
  - SDP
  - File systems
Acknowledgment

- This work was supported in part by:
  - Natural Sciences and Engineering Research Council of Canada (NSERC)
  - Canada Foundation for Innovation (CFI)
  - Ontario Innovation Trust (OIT)
  - Queen’s University
  - Mellanox Technologies
Questions?