LiMIC: Support for High-Performance MPI Intra-Node Communication on Linux Cluster

H. –W. Jin, S. Sur, L. Chai, and D. K. Panda

Network-Based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University
{jinhy, surs, chail, panda}@cse.ohio-state.edu
Contents

• Introduction
• Existing Intra-Node Communication Mechanisms
• Our Approach: LiMIC
• Design and Implementation Issues
• Performance Evaluation
• Conclusions and Future Work
Introduction

• Cluster of Workstations
  – Symmetric multi-processor (SMP) nodes
    • commonly 2 to 8 processors
    • up to 256 processors (e.g., NASA Columbia)
  – High-speed interconnects
    • InfiniBand, Myrinet, Quadrics, etc.

• Message Passing Interface (MPI)
  – *De facto* standard for writing parallel applications
Intra-Node Communication

MPI Application

Node 0

P0
P1
P2
P3

Intra-Node Communication

Node n

P0
P1
P2
P3

Inter-Node Communication

Switch
Contents

• Introduction
• Existing Intra-Node Communication Mechanisms
• Our Approach: LiMIC
• Design and Implementation Issues
• Performance Evaluation
• Conclusions and Future Work
Switch-Level Loopback
NIC-Level Loopback

Node 0

P0  P1  P2  P3

Buffer of P0
Buffer of P3

Memory

NIC

Switch

Node n

P0  P1  P2  P3

Memory

NIC
User-Space Shared Memory

Node 0

P0 P1 P2 P3

Memory

Buffer of P0

Shared Memory

Buffer of P3

NIC

Node n

P0 P1 P2 P3

Memory

NIC

Switch

Two Memory Copy Operations
Contents

• Introduction
• Existing Intra-Node Communication Mechanisms
• Our Approach: LiMIC
• Design and Implementation Issues
• Performance Evaluation
• Conclusions and Future Work
Kernel-Based Memory Mapping

Node 0

- P0
- P1
- P2
- P3

Buffer of P0

Buffer of P3

Memory

NIC

... Switch

Node n

- P0
- P1
- P2
- P3

Memory

NIC
Our Approach: LiMIC

(Linux Kernel Module for MPI Intra-Node Communication)

Earlier Design Approach

- Applications
- MPI Library
- User-Level Protocol
- Specific Network
- Kernel Support

LiMIC Design Approach

- Applications
- MPI Library
- User-Level Protocol
- Specific Network
- LiMIC
- Any Network

No portability to other user-level protocol
No optimization-space for the MPI library developer
No other current generation MPI implementations provide such a kernel support !!!

Portability and Optimizations
Contents

• Introduction
• Existing Intra-Node Communication Mechanisms
• Our Approach: LiMIC
• Design and Implementation Issues
• Performance Evaluation
• Conclusions and Future Work
Design and Implementation Issues

• Kernel Module Based Design
• Portable and MPI Friendly Interface
• Memory Mapping Mechanism
• Copy Mechanism
• MPI Message Matching
Kernel Module Based Design

- **Linux Run-Time Loadable Kernel Module**
  - No kernel modification
  - Portable across major versions

- **LiMIC Kernel Module Can Be Either:**
  - An independent module with device driver
  - Or extensions of device driver
Portable and MPI Friendly Interface

• Independent Interface on:
  – Communication library
  – MPI implementation

• Interfaces
  – LiMIC_Isend (dest, tag, context_id, buf, len, req)
  – LiMIC_Irecv (src, tag, context_id, buf, len, req)
  – LiMIC_Wait (src/dest, req)
  – Etc.

• Interfaces trap into the kernel internally by calling ioctl() system call
Memory Mapping Mechanism

Process 0
- MPI_Isend
- LiMIC_Isend
- ioctl

Send Buffer
- src, dest
- tag, context_id
- page descriptors
- etc.

LiMIC Request Queue

Receive Buffer
- MPI_Irecv
- LiMIC_Irecv
- ioctl

Data Copy

Process 1
- MPI_Irecv
- LiMIC_Irecv
- ioctl

LiMIC Kernel Module
- src, dest
- tag, context_id
- page descriptors
- etc.
Copy Mechanism

• Design Alternatives:
  – Copy on function calls of receiver
    • receiver dependent progress
  – Copy on wait function call
    • memory pin-down on both sender and receiver
  – Copy on send and receive calls
    • better progress
    • less resource usage
    • not prone to skew between processes
MPI Message Matching

- **MPI_Init**
  - Exchange node information for every processes

- **Message Matching Based on Source**
  - Source in the same node
  - Source in a different node
  - Source in the same node and MPI_ANY_TAG
  - MPI_ANY_SOURCE and MPI_ANY_TAG
    - receive request is posted in the MPI queue
    - progress engine calls LiMIC_Iprobe
    - If message matches with request, calls LiMIC_Irecv
MVAPICH: MPI over InfiniBand

- http://nowlab.cis.ohio-state.edu/projects/mpi-iba/
- Powering Supercomputers in the TOP 500 List
  - 7th, 1100-node dual Apple Xserve 2.3 GHz cluster at Virginia Tech
  - 98th, 288-node dual Opteron 2.2 GHz cluster at United Institute of Informatics Problems (Belarus)
- Being Used by More than 220 Organizations World-Wide
- Latest Release
  - MVAPICH (MPI-1): 0.9.5
  - MVAPICH2 (MPI-2): 0.6.0
Contents

• Introduction
• Existing Intra-Node Communication Mechanisms
• Our Approach: LiMIC
• Design and Implementation Issues
• Performance Evaluation
• Conclusions and Future Work
Performance Evaluation

• Experimental System
  – Cluster A
    • 8 nodes with dual Intel Xeon 3.0GHz processor and 512KB L2 cache
  – Cluster B
    • 8 nodes with dual Intel Xeon 2.4GHz processor and 512KB L2 cache

• Experimental Results
  – Microbenchmarks: Latency and Bandwidth
  – Cost Breakdown
  – HPCC Effective Bandwidth
  – NAS Integer Sort
For Small Messages, User Space Shared Memory is Better than Others

For Medium and Large Messages, LiMIC is Better than Others
  - For 128KB, the latency of LiMIC is 71% less than that of User Space Shared Memory
Bandwidth

- For Small Messages, User Space Shared Memory is Better than Others
- For Medium and Large Messages, LiMIC is Better than Others
  - For 128KB, the bandwidth of LiMIC is 313% higher than that of User Space Shared Memory
  - The peak bandwidth is 3929 MillionBytes/Sec
Threshold

- **MVAPICH-0.9.4**
  - User-space shared memory
    - Message Size <= 256KB
  - NIC-level loopback
    - Message Size > 256KB

- **MVAPICH-LiMIC**
  - User-space shared memory
    - Message Size <= 4KB
  - LiMIC
    - Message Size > 4KB

LiMIC Cost Breakdown

- For small messages, the kernel trap overhead is dominant
- For large messages, the copy overhead is dominant

- For small messages, the kernel trap overhead is dominant
- For large messages, the page descriptor mapping overhead is dominant
## HPCC Effective Bandwidth

(MB/s)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Config.</th>
<th>MVAPICH-0.9.4</th>
<th>MVAPICH-LiMIC</th>
<th>Improv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2x1</td>
<td>152</td>
<td>244</td>
<td>61%</td>
</tr>
<tr>
<td></td>
<td>2x2</td>
<td>317</td>
<td>378</td>
<td>19%</td>
</tr>
<tr>
<td></td>
<td>2x4</td>
<td>619</td>
<td>694</td>
<td>12%</td>
</tr>
<tr>
<td></td>
<td>2x8</td>
<td>1222</td>
<td>1373</td>
<td>12%</td>
</tr>
<tr>
<td>B</td>
<td>2x1</td>
<td>139</td>
<td>183</td>
<td>31%</td>
</tr>
<tr>
<td></td>
<td>2x2</td>
<td>282</td>
<td>308</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>2x4</td>
<td>545</td>
<td>572</td>
<td>5%</td>
</tr>
<tr>
<td></td>
<td>2x8</td>
<td>1052</td>
<td>1108</td>
<td>5%</td>
</tr>
<tr>
<td>A&amp;B</td>
<td>2x16</td>
<td>2114</td>
<td>2223</td>
<td>5%</td>
</tr>
</tbody>
</table>

- The improvement percentage remains constant as the number of processes is increased
NAS Integer Sort

- MVAPICH-LiMIC achieves 10%, 8%, and 5% improvement of execution time running Classes A, B, and C on 2x8, respectively
Contents

• Introduction
• Existing Intra-Node Communication Mechanisms
• Our Approach: LiMIC
• Design and Implementation Issues
• Performance Evaluation
• Conclusions and Future Work
Conclusions

• LiMIC
  – High performance MPI intra-node communication
  – MPI friendly interface
  – Independent on proprietary communication libraries

• Performance Results
  – Improvement of latency and bandwidth up to 71% and 313%
  – Improvement of effective bandwidth up to 12% on an 8-node cluster
  – 10% Improvement of NAS IS Class A on an 8-node cluster
Future Work

• Zero-Copy Intra-Node Communication
  – Copy-on-write

• Blocking Support
  – Message driven process scheduling

• Linux Kernel Version 2.6
  – NUMA architecture
Acknowledgements

Our research is supported by the following organizations:

• Current Funding support by

  - Office of Science
  - NSF
  - Intel
  - Mellanox Technologies
  - Sun

• Current Equipment donations by

  - Mellanox Technologies
  - Intel
  - AMD
  - Apple
Thank You

{ jinhy, surs, chail, panda}@cse.ohio-state.edu

Network-Based Computing Laboratory
http://nowlab.cis.ohio-state.edu/