Design and Implementation of Open MPI over Quadrics/Elan4

W. Yu, T.S. Woodall*, R.L. Graham* and D.K. Panda

Dept of Computer Sci. and Engg.
The Ohio State University
{yuw,panda}@cse.ohio-state.edu

Los Alamos National Laboratory*
Computer and Computation Science.
{twoodall,rlgraham}@lanl.gov
Presentation Outline

- Motivation
- Communication Requirements and Objectives
- Design Challenges and Implementation
- Performance Evaluation
- Conclusions
Cluster Computing

- Parallel computing architecture
  - Evolving into tens of thousands of processors
  - More high performance interconnects
- MPI and MPI-2
  - The *de facto* industry standard
  - MPI-2 extends MPI with dynamic process management, IO, one-side communication, more collectives, language bindings, etc
Open MPI

- A new implementation of MPI-2
  - Component-based dynamic architecture
  - Dynamic, fault tolerant process management
  - Concurrent communication over multiple networks
  - Dual-mode communication progress
Presentation Outline

- Motivation
- Communication Requirements and Objectives
- Design Challenges and Implementation
- Performance Evaluation
- Conclusions
Open MPI Communication

- First implemented over TCP/IP
  - Able to aggregate messages over multiple NICs
  - Delivers comparable performance
- Communication stacks on top of two layers:
  - Point-to-point message management layer (PML)
    - Message fragmentation and assembly
    - Ordered *reliable* delivery
    - Scheduling and striping
  - Point-to-point message transport layer (PTL)
    - Network specific, managing network status and communication
    - Presents communication support to PML
Communication Architecture

- collective
- Point-to-point
  - PML
    - Base
    - PTL-TCP
    - PTL-Elan4
    - Ethernet
    - Quadrics
Flow of Open MPI Communication
PML Requirements to PTL Communication Support

• Fault-tolerance
  - Dynamic joining and disjoining of PTLs
  - Communication state monitoring and synchronization

• Concurrent communication
  - PML provides abstraction to handle semantics differences between networks

• Communication progress
  - Non-blocking polling-mode and thread-based asynchronous mode
Overview of Quadrics/Elan4

• **Quadrics Network: QsNet**
  - Tport (MPI oriented) and SHMEM libraries
  - Static communication model between processes
  - Hardware-based collectives
    • broadcast, barrier

• **Communication mechanisms**
  - Queue-based model
    • for messages up to 2KB
  - Remote DMA
    • Arbitrary size messages. RDMA write/read
  - Event mechanism
    • Completion notification
Objectives

• Support MPI-2 dynamic processes over Quadrics
• Incorporate Quadrics RDMA capabilities
• Support dual-mode communication progress
Presentation Outline

- Motivation
- Communication Requirements and Objectives
- Design Challenges and Implementation
- Performance Evaluation
- Conclusions
Design Challenges

• Dynamic MPI-2 process model
  - Communication Initialization and finalization

• Integrating RDMA Capabilities
  - Memory semantics compatibility
  - Protocol mapping

• Communication Progress
  - How to support asynchronous progress?
Dynamic MPI-2 Process Pool

- Communication Initialization and finalization
  - Break the coupling of MPI Rank and VPID
  - Remove the reliance on Global virtual memory
  - Allocate a capability with more contexts
  - Support dynamic and synchronized joining and disjoining of processes
Integrating RDMA Capabilities

- **Memory Descriptor**
  - Right now, an expansion with Elan4_Addr

- **Communication and Completion notification**
  - Using RDMA write/read
  - FIN with RDMA write
  - FIN_ACK with RDMA read

- **Optimization**
  - Chains the control message with RDMA
  - Provides fast, automatic transmission of control messages
RDMA Write

PML  PTL  PTL  PML

schedule  Data/rendezvous  match  complete

update  Ack  matched  update

schedule  RDMA Write  FIN  update

update  complete  update
RDMA Read

PML -> schedule -> Data/rendezvous -> match -> matched

PTL -> update

PTL -> FIN_ACK

PTL -> update

PTL -> complete

PTL -> complete

PML
• Non-blocking Polling Mode
  - PML iteratively checks all outstanding send and receive queues

• Thread-base asynchronous communication
  - Two thread based Communication Progress
    • One for the local completion of DMA descriptors
    • Another for the completion of incoming QDMA messages
  - One thread-based communication progress
    • QDMA messages + local DMA completion to a combined queue
Challenges in Asynchronous Progress with RDMA

• RDMA completion can only be detected with a separated event.
• The event mechanism
  - Supports the completion of N DMA operations with a count N
  - Cannot have one thread per RDMA descriptor
Chained Event

- Is it possible to use events with a count N for shared completion?
Possible Race Condition?

(c) A Count 1 Event Fired
(d) Racing condition
Chained Event + QDMA
Presentation Outline

- Motivation
- Communication Requirements and Objectives
- Design Challenges and Implementation
- Performance Evaluation
- Conclusions
Performance Evaluation

- **Experimental Testbed:**
  - A Quadrics cluster: QS-8A switch, Elan4 cards
  - Dual-SMP Intel Xeon 3.0GHz Processors
  - PCI-X 133MHz/64bit
  - 533MHz FSB
  - 1GB SDRAM memory

- **Experimental Results**
  - Performance with different numbers of completion queues
  - Communication cost in different layers
  - Threading cost
  - Overall performance
Basic Performance with RDMA Read and Write

- RDMA read performs better than RDMA write
- Rendezvous Message without inline data improves performance
- `memcpy()` is replacing the sophisticated datatype engine for
Performance with Chained DMA and Completion Queues

• Chain DMA provides little performance improvement
• ~1us penalty for shared completion queue
• No performance difference with one-Queue or two Queue
Measuring Communication Cost

- L1: PML cost
- L2: PTL latency
Communication Cost in Different Layers

- PML has about 0.5us overhead
- Compared to QDMA, PTL/Elan4 has virtually no overhead for 0-byte messages.
Thread-Based Progress

### Performance Analysis of Thread-based Progression (in us)

<table>
<thead>
<tr>
<th>Message Length</th>
<th>Basic</th>
<th>Interrupt</th>
<th>One-Thread</th>
<th>Two-Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDMA-Read (4B)</td>
<td>3.87</td>
<td>14.70</td>
<td>22.76</td>
<td>27.50</td>
</tr>
<tr>
<td>RDMA-Read (4KB)</td>
<td>15.25</td>
<td>27.16</td>
<td>32.80</td>
<td>47.72</td>
</tr>
</tbody>
</table>

- Open MPI w/ PTL/Elan4 thread-based progression has 18us overhead
- ~1us due to shared completion queue
- ~9us due to interrupts, ~8us due to threading
Overall Performance
- Latency

- Open MPI w/ PTL/Elan4 achieves similar latency for large messages, compared to MPICH-QsNet.

- For small messages, Open MPI w/ PTL/Elan4 has higher cost due to its host-based receive queue and tag matching.
Overall Performance

- Bandwidth

- Open MPI w/ PTL/Elan4 has slightly lower bandwidth compared to MPICH-QsNet for small and large messages.

- For medium messages, Open MPI w/ PTL/Elan4 has significant bandwidth because it does no pipelining.
Presentation Outline

- Motivation
- Communication Requirements and Objectives
- Design Challenges and Implementation
- Performance Evaluation
- Conclusions
Conclusions

• Designed and implemented Open MPI over Quadrics/Elan4
• Integrated Quadrics RDMA capabilities
• Provided dual-mode communication progress
• Support dynamic MPI-2 process model over Quadrics
Web Pointers

NBC-LAB

Homepage: http://nowlab.cis.ohio-state.edu