A Memory Saving Communication Method Using Remote Atomic Operations

Takahiro Kawashima, Masaaki Fushimi, Takafumi Nose, Shinji Sumimoto, Naoyuki Shida
Fujitsu Limited
Motivation

- **Two point-to-point communication modes in K computer MPI:**
  - **Fast communication mode** for limited peers
    - For frequent peers
    - **Intermediate receive buffer:** Allocated 1Mi bytes for each peer lazily
  - **Memory-saving communication mode** for the rest of peers
    - For less frequent peers
    - **Intermediate receive buffer:** Allocated 2Ki bytes for each peer lazily

- **Issue of Post-K MPI:** Memory usage of MPI is still dominant
  - The intermediate receive buffer for **memory-saving** must be fundamentally reduced

![Diagram showing communication modes and intermediate buffer usage](image)
Related Work about Reducing Memory Usage

- **Hardware approach**
  - Shared Receive Queue of InfiniBand
  - Dynamic Connected Transport of InfiniBand [H Subramoni. et al. 2014]

- **Software approach**
  - Specific optimization in MPICH2 [D Goodell. et al. 2011]
  - Advanced Communication Primitives [S Sumimoto. et al. 2016]

- **Our approach**
  - Assumes RDMA-based interconnects with no specific memory-saving features
  - Targets MPI applications which communicate with a large number of peers
  - Aims to reduce intermediate receive buffer (needed by RDMA-based comm.)
  - Based on [*]
## Approach to Solve the Issue

### Approach

- Share an intermediate receive buffer using remote atomic operations
  - New memory-saving communication mode

### Two benefits

- Allocate **only one intermediate receive buffer** per process
- Share an intermediate receive buffer **without contention**

### Target application feature

- One process directly communicates with a large number of peers
  - **For frequent communication peers:** Use fast mode communication mode
  - **For less frequent communication peers:** Use new memory-saving communication mode
Example of Remote Atomic Operations

- Remote atomic operation flow
  1. An initiator specifies an operator and an operand
  2. A value on the remote node memory is loaded and the operation result is stored atomically without the remote node CPU processing
  3. The loaded value is returned to the initiator

- Special case
  - Atomic-or with an operand zero does not update the remote node memory and only returns the remote value
### Overview of Shared Receive Buffer Method

#### Control Variable | Description
--- | ---
**producer counter** | • Position in the shared receive buffer  
• Peers update it in order to know and reserve the writing region
**consumer counter** | • Position already read in the shared receive buffer  
• Peers refer it in order to know whether to be able to write data  
• Updated by its owner

---

![Diagram of Shared Receive Buffer Method](image)

- **Sender A**: Data to send
- **Sender B**: Data to send
- **Receiver**: Control variables
  - Producer counter
  - Consumer counter
  - Region to receive
  - Shared rbuf
- **(1) Atomic-add**
- **(2) Atomic-or**
- **(3) Put**
- **(4) Copy**
- **(5) These regions are zero-cleared after copying received data by receiver**

---

**Notes:***

- **(1) Atomic-add**
- **(2) Atomic-or w/ zero**
- **(3) Put (RDMA Write)**
- **(4) Copy by CPU**
**Result: Memory Usage of Memory-Saving Mode**

- Memory increase between MPI_Init and point-to-point communications (all-to-all)

**Memory Usage for PT2PT Communications**

- 8.3 GiB/node memory usage is expected for one million processes
  - Memory for connection information still increases in proportion to peers

**Environment**

<table>
<thead>
<tr>
<th>Environment</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine</td>
<td>PRIMEHPC FX100</td>
</tr>
<tr>
<td>CPU Frequency</td>
<td>1.975 GHz</td>
</tr>
<tr>
<td>Main Memory</td>
<td>32 GiB / node</td>
</tr>
<tr>
<td>Processes Per Node</td>
<td>32</td>
</tr>
<tr>
<td>Receive Buffer Size</td>
<td>2 KiB (allocated for all communication peers)</td>
</tr>
<tr>
<td>Shared Receive Buffer Size</td>
<td>64 KiB (allocated only one per process)</td>
</tr>
</tbody>
</table>
Result: Inter-Node Ping Pong Latency of Memory-Saving Mode

- Latency on eager protocol increased by 3.2 usec
  - Due to two atomic operations
- Latency on rendezvous protocol increased by 10.9 usec
  - Due to six atomic operations in order to send three control messages
Conclusion

Effect of shared receive buffer method
- Save memory usage for intermediate receive buffer dramatically
- Increase latency about 3 usec per message which is sent by eager protocol
  - Two memory-saving methods (new and existing) should be switched properly

Future work
- Reduce memory usage for connection information
  - Still increasing memory usage in proportion to the number of peers