Layout Aware Hardware Assisted Mechanisms for Non-Contiguous Data transfers

Talk at OSU Booth SC ’22

Kaushik Kandadi Suresh
The Ohio State University
kandadisuresh.1@osu.edu
Introduction to MPI and Derived Data Types (DDT)

- MPI provides the abstraction of rank with private address space
- MPI offers various communication primitives
- MPI provides datatypes for exchanging messages
- Intrinsic types
  - MPI_INT, MPI_DOUBLE, etc.
- Derived Datatypes (DDT)
  - MPI_Type_Contiguous, MPI_Type_Vector, MPI_Type_Indexed, etc.
- HPC applications exchange non-contiguous data
  - Eg: NAS LU, Minighost
- MPI DDT can be used to represent such data
- Transfers the onus of optimization to the implementation

Data Layout in NAS LU

Types of MPI DDT Schemes

(a) Hardware Assisted: Uses SGL/UMR based transfer

- MPI Type Index Layout
- IOV(0), IOV(1), IOV(2)
- Flatten
- Register IOVs
- `ibv_post_send` (with sgl/umr)

(b) Host based: Uses CPU to pack/unpack

- MPI Type Index Layout
- IOV(0), IOV(1), IOV(2)
- Flatten
- Pack
- Single Memory Buffer
- Register Buffer
- `ibv_post_send`
Challenges in DDT performance optimization

- Registration caches can have up to 20% overhead
  - How to enhance the existing registration cache?

- Applications tend to have varied memory layouts
  - How to choose a DDT scheme that performs best for all layouts?
• Observation: (Host vs Hardware Assisted)
  - Host based schemes are better for smaller block lengths
  - Hardware Assisted scheme are good for large block lengths
Experimental Setup

Cluster details

<table>
<thead>
<tr>
<th>Cluster Specs</th>
<th>Expanse</th>
<th>Frontera</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Processor</td>
<td>Dual-socket AMD EPYC 7742 2.25GHz, 64 Cores/socket</td>
<td>Dual-socket Intel Xeon Platinum 8280 2.7GHz, 28 Cores/socket</td>
</tr>
<tr>
<td>System Memory</td>
<td>256 GB</td>
<td>192 GB</td>
</tr>
<tr>
<td>Interconnects between nodes</td>
<td>Mellanox InfiniBand HDR-100 (one-way 12.5 GB/s)</td>
<td>Mellanox InfiniBand HDR-100 (one-way 12.5 GB/s)</td>
</tr>
</tbody>
</table>

• MPI libraries:
  – MVAPICH2, Intel MPI 2019, OpenMPI-4.1

• Benchmarks and Applications:
  – OMB with Vector DDT, DDT-Bench, MiniGhost-miniapplication
Performance of vector benchmark

- Vector of Block Length – 4KB
- Improvement up to 30% over OpenMPI, 2X over IntelMPI and 22% over MVAPICH2 (baseline)

<table>
<thead>
<tr>
<th>Number of Segments</th>
<th>MVAPICH2</th>
<th>Proposed</th>
<th>IntelMPI</th>
<th>OpenMPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Latency (us)

- MVAPICH2
- Proposed
- IntelMPI
- OpenMPI

Improvement:
- 30% over OpenMPI
- 2X over IntelMPI
- 22% over MVAPICH2 (baseline)
Performance of DDTbench

- NASMG: Block length is 8 bytes for X-direction and 256 bytes to 5KB in the Y-direction
- 28% improvement over MVAPICH2 and 2.5X over IntelMPI
- Inputs: \( A = (256,32,32) \), \( B = (512,66,66) \), \( C = (2048,66,120) \), \( D = (5120,92,120) \)

**NASMG**

<table>
<thead>
<tr>
<th></th>
<th>MVAPICH2</th>
<th>Proposed</th>
<th>IntelMPI</th>
<th>OpenMPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WRF**

- The datatypes used in WRF are struct of vectors for both X and Y direction
- We see improvements up to 1.75X compared to MVAPICH2 and up to 2.5X improvements over IntelMPI
- Inputs: \( A = (4,4018,8,4010) \), \( B = (4,2060,8,2056) \), \( C = (4,6012,8,6008) \)

*Latencies are normalized with respect to IntelMPI numbers*
Performance of MiniGhost miniapplication (Weak Scaling)

- Execution time of the proposed scheme is up to 35% better than Intel-MPI, 7.8% better than OpenMPI, and 9% better than MVAPICH2 at a scale of 128 nodes.
Conclusion and Future work

• Conclusion
  – DDT cost is impacted by transfer schemes, memory layouts, and DDT operation
  – Proposed dynamic scheme that considers:
    • Memory Layout
    • Frequency
    • DDT operation
  – Proposed design achieves up to 22% improvement in performance over state-of-the-art MPI libraries at the micro-benchmark level.
  – Demonstrated up to 9% improvement in MiniGhost performance at 128 nodes

• Future work
  – Comprehensive evaluation at large scales for more HPC applications
  – Scaling studies with larger number of processes per node
Thank You!

Kandadisuresh.1@osu.edu

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

Follow us on
https://twitter.com/mvapich

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/