High-Performance Adaptive MPI Derived Datatype Communication for Modern Multi-GPU Systems

Ching-Hsiang Chu, Jahanzeb Maqbool Hashmi, Kawthar Shafie Khorassani, Hari Subramoni, Dhabaleswar K. (DK) Panda

The Ohio State University

{chu.368, hashmi.29, shafiekhorassani.1}@osu.edu,
{subramon, panda}@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~panda
Outline

- Introduction
- Problem Statement
- Proposed Designs
- Performance Evaluation
- Concluding Remarks
Drivers of Modern HPC Cluster Architectures

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Multiple Accelerators (NVIDIA GPGPUs and Intel Xeon Phi) connected by PCIe/NVLink interconnects
- Available on HPC Clouds, e.g., Amazon EC2, NSF Chameleon, Microsoft Azure, etc.

Multi-core Processors

High Performance Interconnects - InfiniBand
<1usec latency, 200Gbps Bandwidth>

Accelerators / Coprocessors
high compute density, high performance/watt
>1 TFlop DP on a chip

SSD, NVMe-SSD, NVRAM
Non-contiguous Data Transfer for HPC Applications

- Wide usages of MPI derived datatype for Non-contiguous Data Transfer
  - Requires Low-latency and high overlap processing

---

**Quantum Chromodynamics**

[Diagram of Quantum Chromodynamics]

**Weather Simulation: COSMO model**

[Diagram of Weather Simulation]

---


State-of-the-art MPI Derived Datatype Processing

- GPU kernel-based packing/unpacking \([1-3]\)
  - High-throughput memory access
  - Leverage GPUDirect RDMA capability

Expensive Packing/Unpacking Operations in GPU-Aware MPI

- Significant overhead when moving non-contiguous GPU-resident data
  - Wasting cycles
  - Extra data copies
  - High Latency!!!

Overhead of MPI Datatype Processing

<table>
<thead>
<tr>
<th>Application</th>
<th>Kernels</th>
<th>Size</th>
<th>Latency (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAS</td>
<td>[32,16,16]</td>
<td>(3.28 KB)</td>
<td>3.28 us</td>
</tr>
<tr>
<td></td>
<td>[512,512,256]</td>
<td>(1012 KB)</td>
<td>512 us</td>
</tr>
<tr>
<td>specfem3D_cm</td>
<td>[1957x245]</td>
<td>(25.8 KB)</td>
<td>1957 us</td>
</tr>
<tr>
<td></td>
<td>[11797x3009]</td>
<td>(173.51 KB)</td>
<td>11797 us</td>
</tr>
</tbody>
</table>

Data transfer between two NVIDIA K80 GPUs with PCIe link.
Analysis of Packing/Unpacking Operations in GPU-Aware MPI

- **Primary overhead**
  - Packing/Unpacking
  - CPU-GPU synchronization
  - GPU driver overhead

- **Can we reduce or eliminate** the expensive packing/unpacking operations?

### Time Breakdown

<table>
<thead>
<tr>
<th>Data transfer</th>
<th>Pack/unpack kernels</th>
<th>CUDA Driver Overhead</th>
<th>Memory Allocation</th>
<th>CUDA Synchronization</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAS specfem3D_cm</td>
<td>3.28 KB</td>
<td>1024 KB</td>
<td>25.7 KB</td>
<td>173.51 KB</td>
<td>3.28 KB</td>
</tr>
<tr>
<td>MVAPICH-GDR 2.3.1</td>
<td>1024 KB</td>
<td>25.7 KB</td>
<td>173.51 KB</td>
<td>3.28 KB</td>
<td></td>
</tr>
<tr>
<td>OpenMPI 4.0.1 + UCX 1.5.1</td>
<td>NAS specfem3D_cm</td>
<td>1024 KB</td>
<td>25.7 KB</td>
<td>173.51 KB</td>
<td></td>
</tr>
</tbody>
</table>

*Data transfer between two NVIDIA K80 GPUs with PCIe link*
Outline

• Introduction
• **Problem Statement**
• Proposed Designs
• Performance Evaluation
• Concluding Remarks
Problem Statement

• How can we exploit load-store based remote memory access model over high-performance interconnects like PCIe and NVLink to achieve “packing-free” non-contiguous data transfers for GPU-resident data?

• Can we propose new designs that mitigate the overheads of existing approaches and offer optimal performance for GPU based derived datatype transfers when packing/unpacking approaches are inevitable?

• How to design an adaptive MPI communication runtime that can dynamically employ optimal DDT processing mechanisms for diverse application scenarios?
Outline

• Introduction
• Problem Statement

• Proposed Designs
  – Zero-copy non-contiguous data movement over NVLink/PCIe
  – One-shot packing/unpacking
  – Adaptive MPI derived datatype processing

• Performance Evaluation
• Concluding Remarks
Overview of Zero-copy Datatype Transfer

- Direct link such as PCIe/NVLink is available between two GPUs
- Efficient datatype layout exchange and cache
- Load-store data movement
Zero-copy Datatype Transfer: Enhanced Layout Cache

- Convert IOV list to displacement list
  - Improved reusability
  - One-time effort
- Cache datatype layout on the shared system memory
  - Accessible within the node without extra copies

```c
struct iovec {
    void *iov_base;
    size_t iov_len;
};

struct dt_layout_vec {
    int displacement;
    size_t iov_len;
};
```
Zero-copy Datatype Transfer: Copy vs. Load-Store

- Exploiting load-store capability of modern interconnects
  - Eliminate extra data copies and expensive packing/unpacking processing

**Existing Packing Schem**

**Proposed Packing-free Schem**

<table>
<thead>
<tr>
<th>Source GPU Memory</th>
<th>Destination GPU Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCIe/NVLink</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source GPU Memory</th>
<th>Destination GPU Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCIe/NVLink</td>
<td></td>
</tr>
</tbody>
</table>

**Zero-copy Datatype Transfer:**
- **Copy** vs. **Load-Store**

**Diagram Labels:**
- Load-Store
- Copy
One-shot Packing/Unpacking Mechanism

- Packing/unpacking is inevitable if there is no direct link
- Direct packing/unpacking between CPU and GPU memory to avoid extra copies

1. GDRCopy-based
   - CPU-driven low-latency copy-based scheme

2. Kernel-based
   - GPU-driven high-throughput load-store-based scheme
Adaptive Selection

- Availability of GPUDirect peer access and GPUDirect RDMA
- Latency- or throughput-oriented communication pattern
Outline

• Introduction
• Problem Statement
• Proposed Designs
• **Performance Evaluation**
• Concluding Remarks
# Experimental Environments

<table>
<thead>
<tr>
<th></th>
<th>Cray CS-Storm</th>
<th>NVIDIA DGX-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Model</td>
<td>Intel Haswell</td>
<td>Intel Skylake</td>
</tr>
<tr>
<td>System memory</td>
<td>256 GB</td>
<td>1.5 TB</td>
</tr>
<tr>
<td>GPUs</td>
<td>8 NVIDIA Tesla K80</td>
<td>16 NVIDIA Tesla V100</td>
</tr>
<tr>
<td>Interconnects</td>
<td>PCIe Gen3, Mellanox IB FDR</td>
<td>NVLink/NVSwitch, Mellanox IB EDR x 8 (Unused)</td>
</tr>
<tr>
<td>OS &amp; compiler version</td>
<td>RHEL 7.3 &amp; GCC 4.8.5</td>
<td>Ubuntu 18.04 &amp; GCC 7.3.0</td>
</tr>
<tr>
<td>NVIDIA driver &amp; CUDA versions</td>
<td>410.79 &amp; 9.2.148</td>
<td>410.48 &amp; 9.2.148</td>
</tr>
</tbody>
</table>

- Benchmarks: Modified DDTBench to use GPU-resident data
  - NAS_MG, MILC, Specfem3D_cm, and Specfem3D_oc
- Application kernels
  - COSMO model & Jacobi Method
- Baseline: MVAPICH2-GDR 2.3.1
Overview of the MVAPICH2 Project

- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.1), Started in 2001, First version available in 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  - Used by more than 3,050 organizations in 89 countries
  - More than 615,000 (> 0.6 million) downloads from the OSU site directly
  - Empowering many TOP500 clusters (Jun ‘19 ranking)
    - 3rd, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center
    - 5th, 448, 448 cores (Frontera) at TACC
    - 8th, 391,680 cores (ABCI) in Japan
    - 15th, 570,020 cores (Neurion) in South Korea and many others
  - Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, and OpenHPC)

- [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu) Partner in the TACC Frontera System

- Empowering Top500 systems for over a decade
Evaluation of Zero-copy Design: Dense Layout

- Zero-copy performs the best in almost all cases!

Please refer to the paper for more performance comparison!
Evaluation of Zero-copy Design: Sparse Layout

- Zero-copy performs the best in all cases by avoiding unnecessary data copies and CPU-GPU synchronization
Evaluation of One-shot Packing Design

**Dense Layout**

- GDRCopy-based scheme performs better for dense layout
- Kernel-based scheme performs better for sparse layout

*Platform: Cray CS-Storm; Two GPUs on different sockets without direct link*

---

**Distributed/Sparse Layout**

Please refer to the paper for more performance comparison!
Evaluation of Applications

- **COSMO Model**
  
  (https://github.com/cosunae/HaloExchangeBenchmarks)

- **Jacobi (2DStencil Computation)**

Platform: Cray CS-Storm, 8 NVIDIA K80 GPUs per node

Platform: NVIDIA DGX-2, 16 NVIDIA V100 GPUs per node

**Improved 3.4X**

**Improved 15X**

**Improved 13%**
Outline

• Introduction
• Problem Statement
• Proposed Designs
• Performance Evaluation
• Concluding Remarks
Concluding Remarks

• Non-contiguous data communication is common in many HPC applications
  – however, it is not optimized in current GPU-Aware MPI implementations

• Proposed designs significantly reduce the packing overhead
  – Zero-copy design eliminates expensive packing/unpacking operations
  – One-shot design avoids extra data copies
  – Adaptive scheme dynamically selects the optimal communication paths

• Publicly available since MVAPICH2-GDR 2.3.2 release
  – http://mvapich.cse.ohio-state.edu/
Thank You!

panda@cse.ohio-state.edu

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

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/