# Halo Exchange Performance on the Sierra Supercomputer


**Jason Burmark**

LLNL


## Physics Application

Communication performance has not improved at the same rate as computational performance on GPU based machines.

The production LLNL rad-hydro code Ares was ported to GPUs but has lower communication performance than expected.

Relevant characteristics:

- 2/3D semi-structured meshes
- parallelized by mesh splitting
- halo exchange communication

## Communication Benchmark

The Comb halo exchange communication performance benchmarking tool was written to find ways to improve communication performance.

- Explore memory spaces, execution methods, and communication methods
- Comb steps
  1. Placeholder computation
  2. Halo-exchange communication
     1. Irecv
     2. Pack and Isend
     3. Wait Recv and Unpack
     4. Wait Send
  3. Placeholder computation

## Communication Options

Look at communication performance on Sierra by combining various resources and techniques.

Memory spaces:

- Host based memory
  - host memory (malloc)
  - host pinned memory
- Device based memory
  - CUDA device memory
- CUDA managed memory

Execution methods :

- Serial
- OpenMP host threading
- CUDA kernels
- CUDA graphs
- Manual CUDA kernel fusion
- Automatic CUDA kernel fusion
- CUDA Aware MPI datatypes

Communication staging methods:

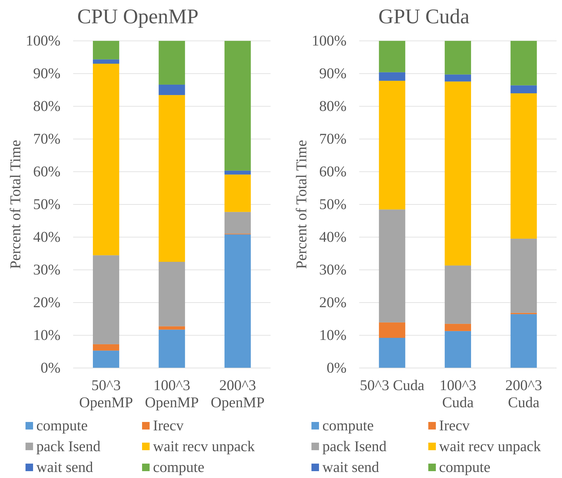
- pack and send one message at a time
- pack messages and send messages in groups
- Pack all and send all messages

Communication libraries and methods:

- MPI
- CUDA Aware MPI (GPU direct)
- Libgpump (GPU direct async)

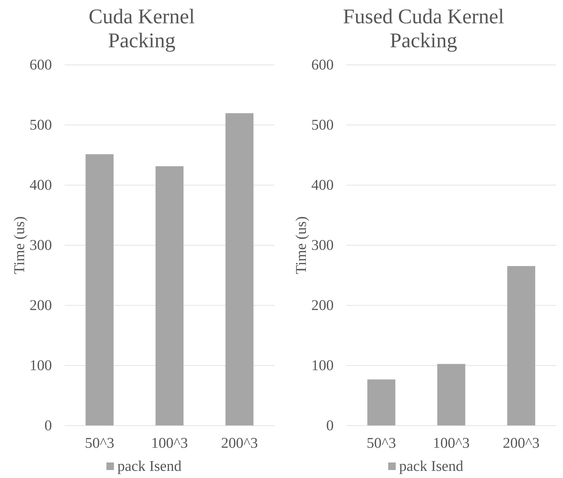
## Large Fraction of GPU Time in Communication

GPU tests spend a higher proportion of time in communication relative to the OpenMP CPU tests for large problem sizes even with the fastest known communication options due to higher on-node performance.



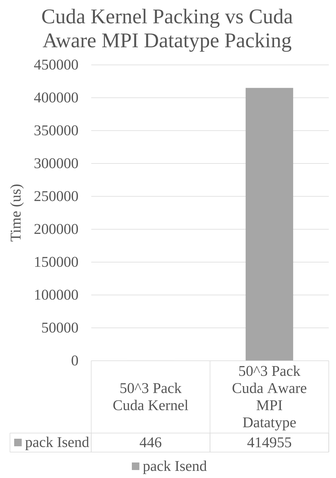

## Kernel Launch Overhead Dominates Packing

Kernel launch overhead dominates packing and unpacking performance. Manually fuse Cuda kernels to reduce the number of kernel launches and reduce the total overhead.




## MPI Datatype Packing is Slow

Spectrum MPI datatype packing and unpacking is not yet performant when used with device memory.




## Problem Details

Code:

- Comb 0.2.0
- https://github.com/LLNL/Comb

Problem:

- 3 variables
- 50^3, 100^3, 200^3 zones per process
- 1-zone halo
- 26 neighbors per process
- 16 nodes, 64 processes, 64 GPUs

Machine:

- rzansel.llnl.gov
- 2 x IBM Power9 + 4 x Nvidia V100

Compilers:

- xl-2019.12.23 (16.1.1.6)
- cuda-10.1.243

MPI:

- spectrum-mpi-2019.06.24

## Future Work

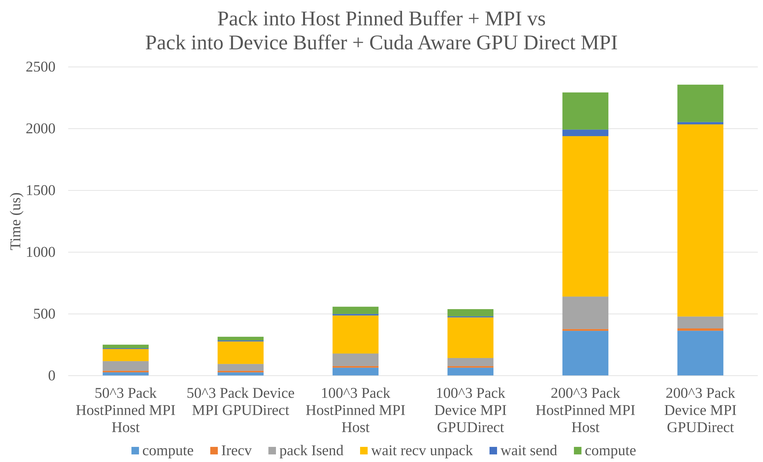
More fully examine:

- performance of MPI with one rank per core. CPU results here use OpenMP to utilize all CPU cores.
- parallelizing groups of messages using Cuda streams to parallelize packing and potentially increase packing and communication overlap.
- automatic Cuda kernel fusion to reduce amount of communication code rewrite while gaining the benefits of kernel fusion.
- stream triggered communication (GPU direct async) to eliminate device synchronization overhead, and potentially increase packing and communication overlap
- Cuda graphs as a means of combining automatic Cuda kernel fusion and stream triggered communication.

## Cuda Aware MPI with GPU Direct Performance is Mixed

Cuda Aware MPI with gpu direct sends data directly from device memory to the network, but the performance is mixed.

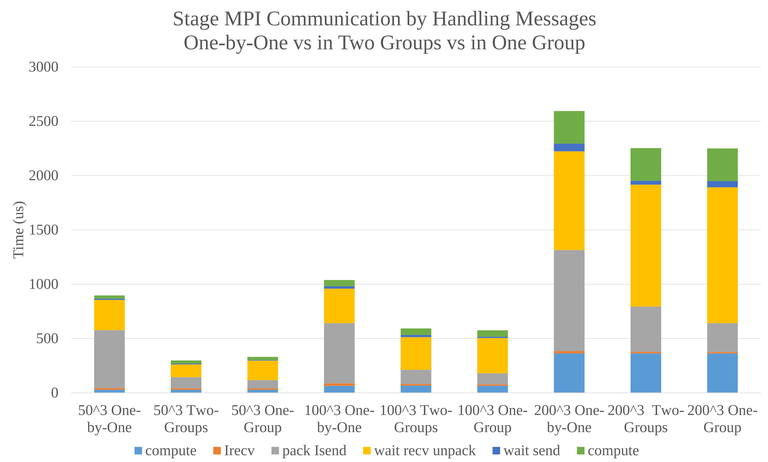
- Performance of packing to device memory is higher than to host pinned memory
- Device Bandwidth to network is lower (hardware issues)




## Stage Communication to Avoid Kernel Launch and Synchronization Overheads

Stage messages in groups to fuse packing kernels and device synchronization for multiple messages. Balance reduced overhead with reduced overlap between packing and communication.

- One-by-One: Pack message, send message (x26)
- Two-Groups: Pack some messages, send some messages (x2)
- One-Group: Pack messages, send messages (x1)




## Acknowledgements


LLNL

- Lee Ellison
- Arjun Gambhir
- Olga Pearce
- Brian Pudliner
- Brian Ryujin



LLNL-POST-804617

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52 07NA27344.