MPI, CUDA, and Kokkos Profiling and Analysis

Rei Haskins

Patrick Bridges, Kurt Ferreira, Scott Levy

Department of Computer Science UNM

Motivation

- Determine whether changes made to communication patterns and how data is packed onto GPUs improves performance.
- Learn more about data movement and communication within HPC.

Analysis of GPU Applications

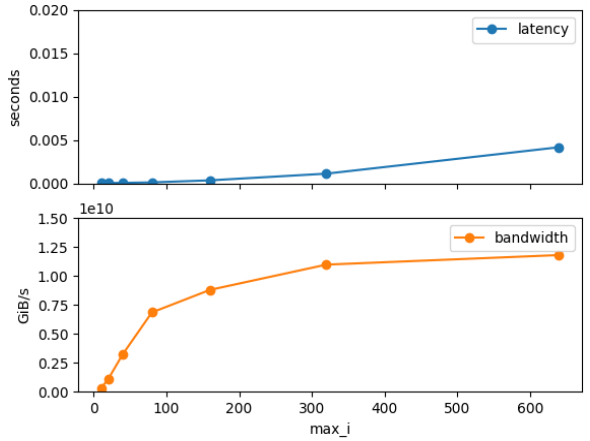
- Goal: Understand where time goes on the GPU during MPI communication.
- Created an MPI library to profile GPU activities during communication.
- Made a ping pong test that uses real application datastructures and MPI types.
- Plan to run on a variety of systems and MPI versions.

MPI/Cuda/Kokkos Ping Pong Test

- Extracted array datastructures from FIESTA application.
- Test supports three communication methods:
  - Direct: Use of MPI datatypes, sends and receives directly between GPUs without copying data first.
  - CUDA Aware: Packs data into a flat Kokkos buffer. Next sends and receives are used between GPUs.
  - Copy: Performs a deep copy to move data to the host before sending and receiving between GPU nodes.

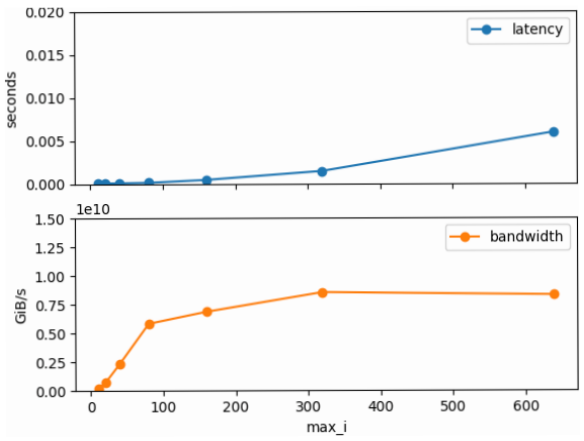
An Example Using The Ping Pong Test

Direct method, using C order layout.



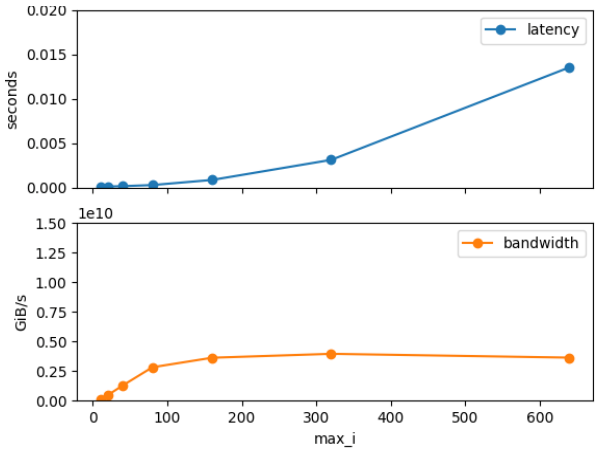

```
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   50.02%  921.67ms     47013  19.604us  1.6640us  25.983us  [CUDA
memcpy DtoH]
                   48.25%  888.95ms     47011  18.909us  1.4080us  25.216us  [CUDA memcpy
HtoD]
                    1.73%  31.819ms         5  6.3639ms  147.80us  31.228ms  void
Kokkos::Impl::cuda_parallel_launch_local_memory<Kokkos::Impl::ParallelFor<Kokkos::Impl:
:ViewValueFunctor<Kokkos::Cuda, double, bool=1>, Kokkos::RangePolicy<>,
Kokkos::Cuda>>j(Kokkos::Cuda)
```

CUDA aware method, using C order layout.




```
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   33.69k%  1.79666s      1000  1.7967ms  1.7772ms  1.8310ms
_ZN6Kokkos4Impl33cuda_parallel_launch_local_memoryINS0_11ParallelForIZ10cuda_awa
reiiNS_4ViewIPPPPdJNS_11LayoutRightEEEE11inputConfigiiS9_S9_S9_S9_P15ompi_dat
atype_tSC_SC_SC_EUliiiiE_NS_13MDRangePolicyIJNS_4RankILj4ELNS_7IterateE0ELSG
_0EEEEEENS_4CudaEEEEEvT_
                   33.45%  1.78390s      1000  1.7839ms  1.7704ms  1.8263ms
_ZN6Kokkos4Impl33cuda_parallel_launch_local_memoryINS0_11ParallelForIZ10cuda_awa
reiiNS_4ViewIPPPPdJNS_11LayoutRightEEEE11inputConfigiiS9_S9_S9_S9_P15ompi_dat
atype_tSC_SC_SC_EUliiiiE0_NS_13MDRangePolicyIJNS_4RankILj4ELNS_7IterateE0ELS
G_0EEEEEENS_4CudaEEEEEvT_
                   16.45%  877.22ms     47013  18.659us  1.6960us  22.944us  [CUDA memcpy
DtoH]
                   15.81%  843.23ms     47011  17.936us  1.4080us  20.000us  [CUDA memcpy
HtoD]
```

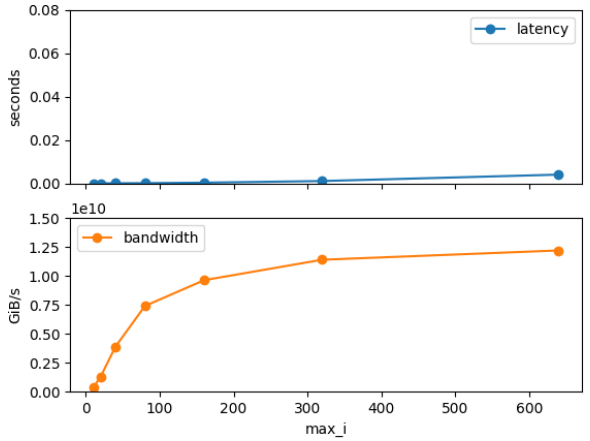
Copy method, using C order layout




```
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   33.69%  1.79666s      1000  1.7967ms  1.7772ms  1.8310ms
_ZN6Kokkos4Impl33cuda_parallel_launch_local_memoryINS0_11ParallelForIZ10cuda_awa
reiiNS_4ViewIPPPPdJNS_11LayoutRightEEEE11inputConfigiiS9_S9_S9_S9_P15ompi_dat
atype_tSC_SC_SC_EUliiiiE_NS_13MDRangePolicyIJNS_4RankILj4ELNS_7IterateE0ELSG
_0EEEEEENS_4CudaEEEEEvT_
                   33.45%  1.78390s      1000  1.7839ms  1.7704ms  1.8263ms
_ZN6Kokkos4Impl33cuda_parallel_launch_local_memoryINS0_11ParallelForIZ10cuda_awa
reiiNS_4ViewIPPPPdJNS_11LayoutRightEEEE11inputConfigiiS9_S9_S9_S9_P15ompi_dat
atype_tSC_SC_SC_EUliiiiE0_NS_13MDRangePolicyIJNS_4RankILj4ELNS_7IterateE0ELS
G_0EEEEEENS_4CudaEEEEEvT_
                   16.45%  877.22ms     47013  18.659us  1.6960us  22.944us  [CUDA memcpy
DtoH]
                   15.81%  843.23ms     47011  17.936us  1.4080us  20.000us  [CUDA memcpy
HtoD]
```

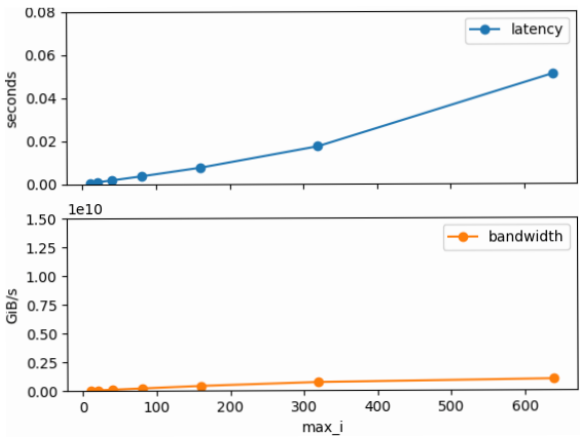
Changing Data Access Direction

Direct method in x direction, up to 640



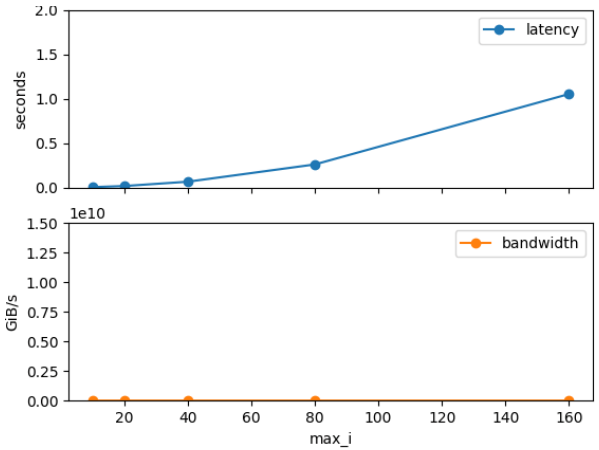
Type	Time(%)	Time	Calls	Avg	Min	Max	Name
GPU activities:	36.36%	24.672us	13	1.8970us	1.8240us	2.1760us	[CUDA memcpy DtoH]
	24.10%	16.352us	11	1.4860us	1.3760us	1.7920us	[CUDA memcpy HtoD]

Direct method in y direction, up to 640



Type	Time(%)	Time	Calls	Avg	Min	Max	Name
GPU activities:	51.01%	76.793ms	40011	1.9190us	1.3760us	2.5280us	[CUDA memcpy HtoD]
	48.97%	73.722ms	40013	1.8420us	1.7920us	3.6800us	[CUDA memcpy DtoH]

Direct method in z direction, up to 160



Type	Time(%)	Time	Calls	Avg	Min	Max	Name
GPU activities:	56.61%	2.94753s	1600013	1.8420us	1.7910us	3.0400us	[CUDA memcpy DtoH]
	43.39%	2.25893s	1600011	1.4110us	1.3120us	2.4960us	[CUDA memcpy HtoD]

Next Steps

- Modify the ping pong test to use other dimensions other than the x dimension and to use Fortran order arrays.
- Look at using other versions of MPI to see how GPU usage varies.
- Run on different systems or accelerators.
- Longer term:
  - - Non-NVIDIA accelerators.
  - - Integration with other performance monitoring systems.

Acknowledgements

- This work was [partially] supported by the U.S. Department of Energy's National Nuclear Security Administration (NNSA) under the Predictive Science Academic Alliance Program (PSAAP-III), Award #DE-NA0003966
- Advisor: Patrick Bridges
- Mentors: Kurt Ferreira and Scott Levy

Questions?