Sandia National Laboratories

Performance Insights into Device-initiated RMA using Kokkos Remote Spaces

Daniel Mishler[12], Jan Ciesko[1], Stephen L. Olivier[1], George Bosilca[2]

[1]Sandia National Laboratories, [2]University of Tennessee, Knoxville

Achieving scalable performance on supercomputers requires careful coordination of communication and computation. Often, MPI applications rely on buffering, packing, and sorting techniques to accommodate a two-sided API, minimize communication overhead, and achieve performance goals. As interconnects between accelerators become more performant and scalable, programming models such as SHMEM may have the opportunity to enable bandwidth maximization along with ease of programming. In this work, we take a closer look at device-initiated PGAS programming models using NVIDIA Corp's NVSHMEM communication library and our interface through the Kokkos Remote Spaces project. We show that benchmarks can benefit from this programming model in terms of performance and programmability. We anticipate similar results for mini-applications.

Kokkos Remote Spaces

- Kokkos: https://github.com/kokkos/kokkos/
  - Cross-platform framework which exposes parallelism.
  - Used in production applications at Sandia: reliable and well maintained.
- Kokkos Remote Spaces: https://github.com/kokkos/kokkos-remote-spaces
  - Also cross-platform framework on top of Kokkos: exposes PGAS & SHMEM.
  - Objective: provide a reliable and usable software layer for programmers.
- Potential application benefits
  - 3x less LOC for conjugate gradient compared to MPI two sided version.
  - Enables more flexible ad hoc data exchange for irregular applications (compared to send-lists in MPI).

MPI 2-sided (CUDA-unaware)

```
struct mpi_add {
  ViewType_t A, B;

  mpi_add(ViewType_t A_, ViewType_t B_) : A(A_), B(B_) {
    ;
  }
  KOKKOS_FUNCTION
  void operator()(int i) const { A(i) += B(i); }
}
void mpi_unaware_benchmark(int minterval) {
  Kokkos::deep_copy(A_h, A);
  RemoteSpace_t().fence();
  if (comm.me == 0) {
    MPI_Isend(A_h.data(), minterval, MPI_DOUBLE, 1, 1, comm.comm,
              &mpi_request_send);
    MPI_Wait(&mpi_request_send, MPI_STATUSES_IGNORE);
  }
  if (comm.me == 1) {
    MPI_Irecv(B_h.data(), minterval, MPI_DOUBLE, 0, 1, comm.comm,
              &mpi_request_recv);
    MPI_Wait(&mpi_request_recv, MPI_STATUSES_IGNORE);
  }
  Kokkos::deep_copy(B, B_h);
  RemoteSpace_t().fence();

  if (comm.me == 1) {
    Kokkos::parallel_for("mpi_stream", policy_t({0}, {minterval}),
                         mpi_add(A, B));
  }
  RemoteSpace_t().fence();
}
```

Kokkos Remote Spaces

```
struct remote_add {
  RemoteView_t A;
  int my_min_i, rstart;
  remote_add(ViewType_t A_, int my_min_i_, int rstart_)
      : A(A_), my_min_i(my_min_i_), rstart(rstart_) {
    ;
  }
  KOKKOS_FUNCTION
  void operator()(int i) const { A(my_min_i + i) += A(rstart + i); }
};
void remote_benchmark(int minterval) {
  if (comm.me == 0) {
    Kokkos::parallel_for("remote_stream", policy_t({0}, {minterval}),
                         remote_add(A, my_min_i, rstart));
  }
  RemoteSpace_t().fence();
}
```

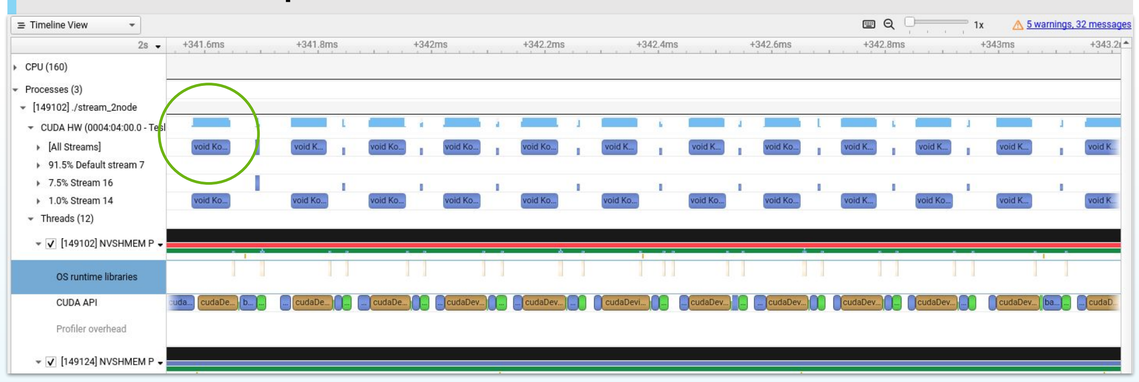
CUDA-unaware MPI host transfers (elided in CUDA-aware MPI version)

2 sided MPI version uses Isend/Irecv

Kokkos Remote Spaces exploits remote access through PGAS

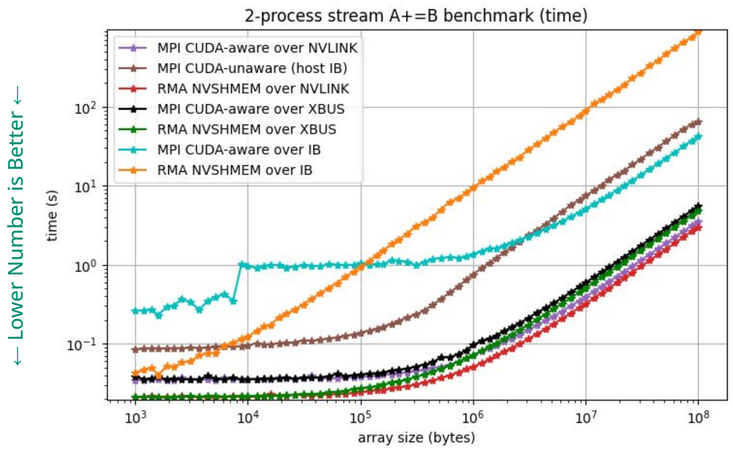
MPI 2-sided (CUDA-unaware)

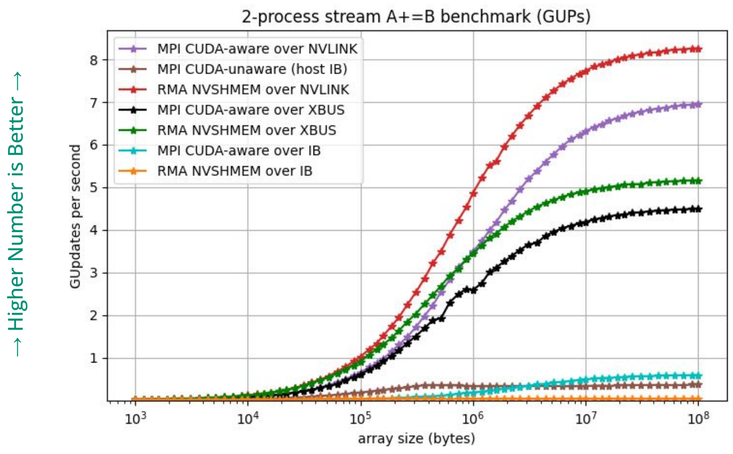
Kokkos Remote Spaces



Above: traces visualized with NVIDIA Nsight that show time taken for host-device transfers that can be skipped with device-device communication exposed through Kokkos Remote Spaces.

Results





Above are two representations of results from streaming A(i)+=B(i) element-wise on 2xV100 NVIDIA GPUs. The benchmark provides 2 arrays, A belonging to process 0, and B belonging to process 1.

MPI: *Uses 2 sided MPI to exchange data. B must be packed into a buffer and sent to process 0.*

RMA: *A and B both accessed through PGAS. Kokkos Remote Spaces exposes NVSHMEM to perform the operations.*

We performed the runs with CUDA-aware and CUDA-unaware MPI, the latter being a host transfer. "XBUS" represents non-NVLINK on-node GPU to GPU connections.

Poor performance is observed in Kokkos Remote Spaces over IB because at the time of this poster's publishing, data transfers only occur element-wise in Kokkos Remote Spaces.

Kokkos Remote Spaces Current Status

Feature	Current Capability
Supported Vendors	NVIDIA Corp, AMD
RMA Operations	Put, Get
Parallel Patterns	All routines Kokkos supports
Debugging support	All routines Kokkos supports

Future Challenges

- Add support for using the shmem calls for more than a single element (e.g. send 128 bytes in one call) to enable OSU latency benchmark.
- Include MPI one-sided codesize and performance in benchmarks.
- Investigate remaining performance anomalies.
- Evaluate on miniapplications (e.g. heat3d, conjugate gradient).

ECP EXASCALE COMPUTING PROJECT

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Sandia National Laboratories