Communication Abstraction Design

Tony Skjellum
University of Tennessee at Chattanooga

PSAAP III Review
September 29, 2022

Outline---UPDATE

- Some hurdles to higher performance
- Starting to solve CPU-GPU Issues
- Abstraction Strategy
- Targets for next APIs/abstractions
- MPI APIs/Abstractions Underway
- Partitioned Collectives & Partitioned Neighbor Collectives
- Next-generation partitioned communication
- Center C++ Efforts
- Asynchronous and triggered operations and schedules
- Discovery of optimizations
- Conclusions

[Some] Hurdles to higher performance

- GPU-CPU-NIC interactions
  - Packing/unpacking data (where and when and pipelined?)
  - Initiating transfers, triggering
- Overlapping communication and overheads
- Many apps spending significant time in MPI nowadays
- Mismatch of current APIs to application transfer patterns
- Scheduling of transfers to use resources efficiently
  - Progress
  - Predictability
  - Reproducibility
- Abstraction barriers from the C Interface
- Adoption/standardization/promulgation

Starting to Solve CPU-GPU issues

- Partitioned point-to-point communication, stage 1
  - Addresses overlap of communication and computing
  - Advanced by our center team w/ Sandia collaborations
  - Provides interface at the MPI+X (other than GPU)
  - Partitioning gives pipelining without derived datatypes
  - “Straightforward” to transform certain codes to use
- Started 2015, standardized in part in MPI-4 (June 2021)
- Demonstrates we can get new ideas “quickly” into MPI with application and architecture relevance
- It’s a start, its not completely solved!

Abstraction Strategy

- Analyze needs (done this)
- Determine what abstractions will help (design them)
- Prototype in MPI Advance (first working versions, gain usership, tweak performance, get feedback)
- Adoption into apps via MPI Advance
- Upstreaming into ExaMPI (deeper integration into implementation, more performance, understand tradeoffs)
- Push code and experience to MPICH and OpenMPI, etc

Targets for next APIs/abstractions

- Direct application use
  - Improve inline use of MPI that varies from app to app
  - Improve hiding of the specifics of how GPU-CPU communicate and gather/scatter data with greater performance portability
- Support enhanced performance of productivity libraries
  - Example: Make Kokkos/Raja better for inter-node comms
  - Apps that leverage these gain performance portability

MPI APIs/abstraction underway

- Forum working groups considering
  - Tweaks to partitioned point-to-point for accelerators (MPI-4.1)
    - Buffer readiness
    - Receive flexibility of partitions
  - Plan for partitioned collective operations (MPI-5)
  - Triggered/asynchronous operations at user-level (MPI-5)
- Center participants involved in relevant Forum WGs
- MPI Advance will be tracking proposed functionality

Partitioned Collectives

- Definitions of partitioned collectives underway
- Early prototypes of several partitioned collectives (MPIPCL -> MPI Advance)
- Application relevance clear in several cases
  - COMB, HiGrad (neighbor)
  - Multi-D FFT (partitioned Alltoallv)
  - Sparse iterative solvers (neighbor)
  - Large allreduces (e.g., for stochastic gradient descent)
- Matches to application patterns, exploits persistence, …

Partitioned Neighbor Collectives

- Clear specification and prototypes for partitioned neighbor collectives (e.g., MPI_Pneighbor_alltoall*) in MPI Advance
- Demo of performance gains from using partitioning in collective scenarios in exemplar apps
- Studying and simplify using graphs/topologies with these collectives
- Extending collectives to recognize graphs are bidirectional for some apps (cf, Dr. Bienz' work – Locality-Aware Persistent Collectives) – Gerald Collom internship @ LLNL
- Integrate into ExaMPI

C++ APIs/reimagining

- C++ APIs
  - Goal: Native as part of MPI-5
  - Ad hoc as demonstrated value in this R&D (ExaMPI, MPI Advance)
- Natural to C++17/20/23…, not clone of C interface
- Match application programming language
- Underway, we have efforts to use C++ directly, abandoning C interface for C++ apps and productivity libraries
- Support greater performance, not just more productivity
- Enable more flexibility to an MPI to offload and orchestrate (including with other libraries)

Center efforts in C++

- Proposals coming from our center with others for MPI-5 standardization (but still early)
- Upstream normative headers for C++ API for existing implementations (MPI Advance)
- Kokkos-enhanced ExaMPI – Kokkos used by app and by MPI for greater performance, all C++
- Potential for the graceful replacement of classical MPI derived datatypes (meta programming, gather/scatter without datatypes)

Next-generation partitioned communication

- Use memory management concepts from C++
- Exploit native memory spaces from Kokkos/Raja type system (within a fast MPI implementation)
- Generalize partitions in new ways
  - Unequal in size
  - Even more unordered
  - Memory managed by MPI
  - Friendly to adaptive routing

Next-generation partitioned communication

- Apply to point-to-point and forthcoming partitioned neighbor collective APIs
- Enables graceful replacement of classical MPI derived datatypes (meta programming, gather/scatter without datatypes)
- These are all doable for us in FY23-24 work, but standardizing will take community support

Maintaining Neighbor Information

- It is expensive to build a single neighbor collective operation based on a communicator
- When small or frequent changes occur in an applications' topology, costs prohibitive
- New APIs are needed to allow modification of graphs in neighbor collective operations without these high overheads
- Relevant in Adaptive Mesh, AMG within nonlinear solvers

Asynchronous and triggered operations and schedules

- We have prototyped schedule-based MPI (intra implementation and user-level)
- Amazon, Argonne, and others are proposing asynchronous and stream-based operations for triggering and offload
- Programming models like CUDA now incorporate schedules
- Can be useful together with partitioned communication
- Can be useful for fusing collective operations
- We have planned this for year 3-4, starting with ExaMPI

Exploratory Work

- Runtime system to support queries between Kokkos/Raja and MPI – two way communication about using smart buffer and pack/unpack choices
  - How large should partitions be?
  - What kind of buffers for allocation of buffers?

Summary

- Key achievements over – Year 2
  - MPI Advance Release (MPIPCL, Locality-Aware Neighbor Persistent Collectives, MPL-lite)
  - Early specification and prototyping of partitioned collectives (MPIPCL next release)
  - Conceptualizing generalized partitioned communication
  - Contributions to MPI-4.1 APIs for partition communication
- Key plans for communication abstraction design – Year 3
  - User-defined schedules
  - Partitioned collectives
  - Exploring efficient APIs for neighbor partitioned collectives (based on Locality-Aware Neighbor Collectives)
  - C++ API studies

Conclusions

- On-going Need for Communication Abstraction Design
- We want to make MPI's APIs and behaviors match what applications need and are optimizable for
  - C++ applications and libraries
  - CPU-GPU[-SmartNIC] architectures
- We have a strategy going from concept through deployment and eventual standardization
- We're engaged in standardizing the key improvements for CPU-GPU architectures, but much more to do
- Year-3 emphasizes improved APIs moved into apps and supporting libraries