Argonne NATIONAL LABORATORY

# MPI Forum: RMA WG

*William Gropp and Rajeev Thakur*

*RMA Working Group Chairs*

*Presenter: Pavan Balaji*

U.S. DEPARTMENT OF ENERGY

# Active Proposals

- MPI Generalized Atomics
- Threaded Communication for MPI RMA
- Neighborhood Communication in MPI RMA
- Nonblocking RMA synchronization
- RMA Notification
- Interoperability of MPI shared memory with C11, C++11 and other language semantics
- Proposals in collaboration with other WGs:
  - Fault tolerance in MPI RMA
  - MPI_IN_PLACE like semantics for collectives on shared-memory buffers

# MPI Generalized Atomics

- MPI-3 atomic operations are, in some cases, restrictive and are not precisely defined
- Two proposals:
  - Clarify what operations are atomic and what are not (minor change)
  - Allow for generality of atomic operations with room for performance optimization
- Generality: Ability for different atomic operations to be issued on the same target location
- Performance: Additional info hints to restrict what the user will use (e.g., only CAS, only FOP, only basic datatypes)

*https://github.com/mpi-forum/mpi-forum-historic/issues/416*

# Threaded Communication in MPI RMA

- MPI-3.1 RMA semantics are well defined for multithreaded communication, but can lead to livelocks
  - E.g., if one thread is continuously issuing RMA operations while another thread is waiting on a flush
- This proposal considers adding threaded RMA operations where each thread can independently issue and complete operations
  - Requires the MPI library to look up thread ID information
  - Might require new compilers for performance

# Neighborhood Communication in MPI RMA

- MPI-3 defined neighborhood collectives where a process only communicates with its neighbors
- Neighborhood RMA is a generalization of that concept to allow RMA to neighboring processes
  - Allows MPI implementations to optimize state that is internally managed
  - Primarily an optimization for memory usage (e.g., MPI does need to not store information about nonneighbor processes)
    - Can also improve performance in some rare cases

# Nonblocking RMA Synchronization

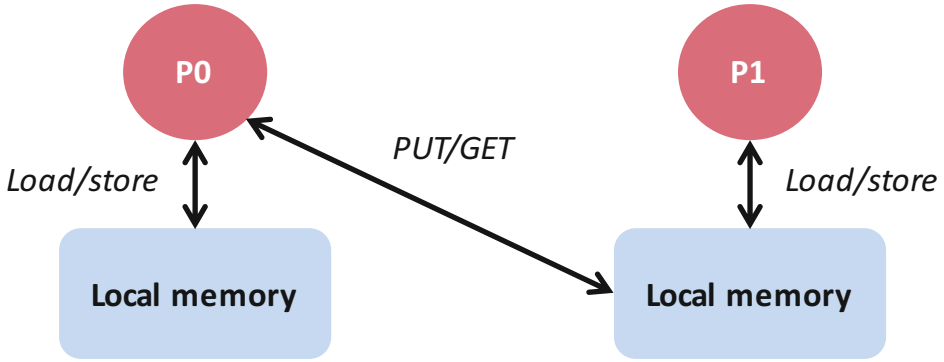
- RMA communication operations are nonblocking
- Some RMA synchronization operations are blocking
  - E.g., MPI_WIN_FENCE after issuing several PUT/GET operations
- Interferes with event-driven applications which want to process completion events as they occur
  - E.g., MPI_Waitany(…) followed by a handler to process whichever request completed
  - Can be done with threads where a thread blocks on call and then sends a "notification" message to unblock the MPI_Waitany
    - Cumbersome and requires a different thread for each simultaneously blocking operation
- Proposal: Nonblocking variants of synchronization operations

# RMA Notification

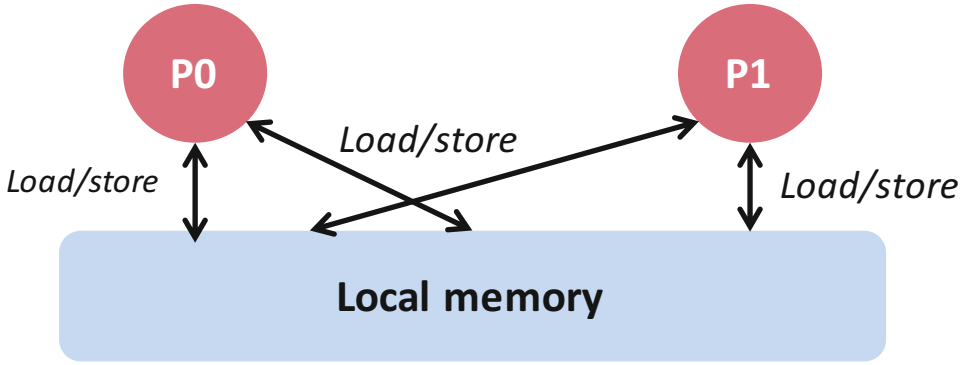
- In passive target mode, notifying the target that data has been transmitted is currently inefficient
- Two proposals for target notification:
  - Notification on PUT/GET
  - Notification on Flush
- Idea is to notify the target when the data has been deposited into the target public memory

*https://github.com/mpi-forum/mpi-issues/issues/59*

# MPI Shared Memory + Language Interoperability




***Traditional RMA windows***




***Shared memory windows***

- Shared memory windows allow application processes to directly perform load/store accesses on all of the window memory
  - E.g., x[100] = 10
- Subtle issues with respect to ordering of load/store operations are not clearly defined
  - Many debates on what the right way to use it is
- Working on defining shared memory access semantics
  - Interoperability semantics with languages (C11, C++11, new Fortran)
  - Usage of MPI_WIN_SYNC for older languages

https://github.com/mpi-forum/mpi-forum-historic/issues/481

# Join Us!

- Fortnightly meetings
- Every alternate Monday from 2-3pm US central time

- Contact one of the chairs if you want to contribute!