Fast One-Sided Communication on Supercomputers and Application to Scientific Codes

Jeff R. Hammond\textsuperscript{1} (jhammond@alcf.anl.gov), Sreeram Potluri\textsuperscript{2}, Cynthia Gu\textsuperscript{1}, Alex Dickson\textsuperscript{1}, Jim Dinan\textsuperscript{1}, Ivo Kabadshow\textsuperscript{1}, Pavan Balajji\textsuperscript{1}, Vinod Tipparaju\textsuperscript{1}

\textsuperscript{1} Argonne National Laboratory \textsuperscript{2} Ohio State University \textsuperscript{3} Florida State University \textsuperscript{4} The University of Chicago \textsuperscript{5} Jülich Supercomputing Centre, \textsuperscript{6} Oak Ridge National Laboratory

Motivation

Hardware architectural features
- One-sided RDMA support is ubiquitous.
- Noncontiguous operations enabled in hardware.
- Processor and network both possess significant concurrency (10- to 100-fold).
- Torus topology with adaptive routing (to avoid both hotspots and faults).

Communication middleware
- Message-passing API MPI requires matching semantics.
- Portals 4 defaults to weak ordering.
- OpenG4HMEM is unordered (fence required).
- GA/ARMCI stipulates strict p2p ordering (location consistency) of blocking operations.

In order to provide the best performance for the most common applications, vendors do not provide location consistency in hardware. On Blue Gene/P Cray Gemini and Blue Gene/Q, enforcing both limits per-message bandwidth and extra messages to flush the network. Extensive analysis of three applications reveals that the one-sided applications built on top of Global Arrays (or ARMCI) do not require the level of memory consistency provided. By relaxing this consistency to the lowest level required for each application, we observe improved performance due to better alignment with hardware features. Further performance is gained from a stripped down implementation of one-sided. Rather than provide all possible features, the implementation only delivers those that the application requires. In many cases, these can be selected dynamically at runtime.

OSPR (One-Sided PRimitives)

OSPR (One-Sided PRimitives) is a new one-sided communication runtime system targeting implementation of the PGAS programming (e.g. Global Arrays, UPC and CAF). It supports the key primitives of remote put, get and accumulate for contiguous (both blocking and non-blocking), strided and general noncontiguous input buffers, remote atomics (e.g. fetch-and-add and lock/unlock), fence and request-based synchronization and utility functions for managing memory. Unlike MPI-2 RMA, local and remote completion are treated separately.

By design, this library is thread-safe and can utilize parallelism in the network controller. For example, the Blue Gene/Q PMI API exposes parallelism in the ML I/O context, which allow multiple threads to initiate communication independently without a global mutex. Modern interconnects frequently do not provide ordering by default, so we support three different ordering modes so that application programmers can enforce ordering only when necessary in order to maximum performance.

Application: ScaFaCoS

- The fast multiple method (FMM) is an exascale-oriented algorithm due to neighborhood communication, hierarchical structure and efficiency for very large problems.
- ScaFaCoS (developed at Jülich Supercomputing Centre) relies upon small to medium put operations and fast synchronization.
- The scaling of this implementation of FMM is limited if put latency is poor. The put latency of OSPR on Blue Gene/P is an order-of-magnitude less than MPI-2 or ARMACI, enabling scaling to 300K cores.

Application: NEUS

Non-equilibrium umbrella sampling (NEUS) replaces a sequential single dynamics trajectory with many trajectories which can be computed in parallel and is thus a massively-parallel algorithm for the time dimension of biomolecular simulation.

NEUS is implemented using a global state updated by atomic (complex) operations asynchronously. The first implementation of NEUS used Global Arrays (GA) Put/Get operations in conjunction with Lock/Unlock provide a means to implement the necessary tools in a straightforward way.

One-sided communication allows a distributed global state wherein atomic updates can occur simultaneously except when locked. A master-worker approach would force unnecessary serialization of updates.

Application: Cray Gemini

- Noncontiguous one-sided APIs are easier to use.
- There is a strong rationale for using noncontiguous operations in turn.
- The reason ARMCI is faster for large local put/get operations is that it offloads these to the DMA, but this consumes network resources that are better devoted to real (i.e. remote) communication.

NEUS can be implemented with Global Arrays, MPI-2 RMA, PGAS languages and any other situation that provides asynchronous communication. Therefore, we can use this code to evaluate the implementation quality and semantic efficiency of such codes.

The submitted poster has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). A Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. A U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.