The impact of process topology on RMA programming models: A study on NERSC Perlmutter

Nikodemos Koutsoheras[1], Sayan Ghosh[2], Joshua Suetterlein[2], Nathan Tallent[2], Abhinav Bhatele[1]

[1]University of Maryland, College Park, [2]Pacific Northwest National Laboratory

Introduction

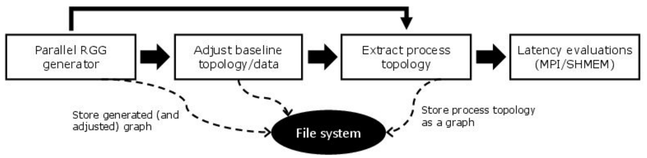
- Remote Memory Access (RMA) provides an alternate mechanism for data movement by separating communication with synchronization
- Current benchmarks conduct “best case” measurements assuming fixed topology, not considering varied process topologies induced by irregular applications like graph analytics

Process Topology Background

- A process graph topology is the logical arrangement of processes as per application data distribution in the form of a cartesian multidimensional grid or a graph (a process is represented as a “node”, with lines or edges connecting the nodes)
- Each node in a process grid has a fixed number of neighbors; each node in a process graph can have an arbitrary number of neighbors

Generating Process Topologies

- A Random Geometric Graph (RGG) is generated across processes in parallel, such that for the baseline topology every process shares data with its two adjacent processes, depicting a 1D stencil (grid) pattern
- Baseline topology is modified by adding customizable amount of "cross" edges to original RGG, increasing overall degree of the neighbors in process graph
- An extra 5% of edges could transform the process graph from sparse to dense
- Process graph topology can be saved as a binary file to avoid RGG creation overhead on simultaneous runs
- We measure communication latencies (less is better) between process pairs in a topology induced by a customizable RGG



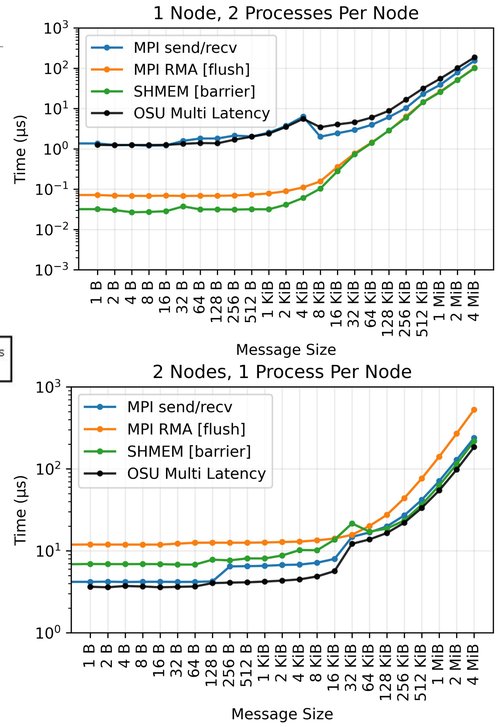
Algorithm 1 Test for data transfer latencies under process graph topology.
Input: $G = (V, E)$, (undirected) process topology graph, G.

1: **for** $s = 1$, s*=2, while $s <= 4MiB$ **do**
2: **for** $u \in$ adj(my_rank) **do** {my process neighbors}
3: async put s bytes to rank u
4: sync {complete outstanding communication}

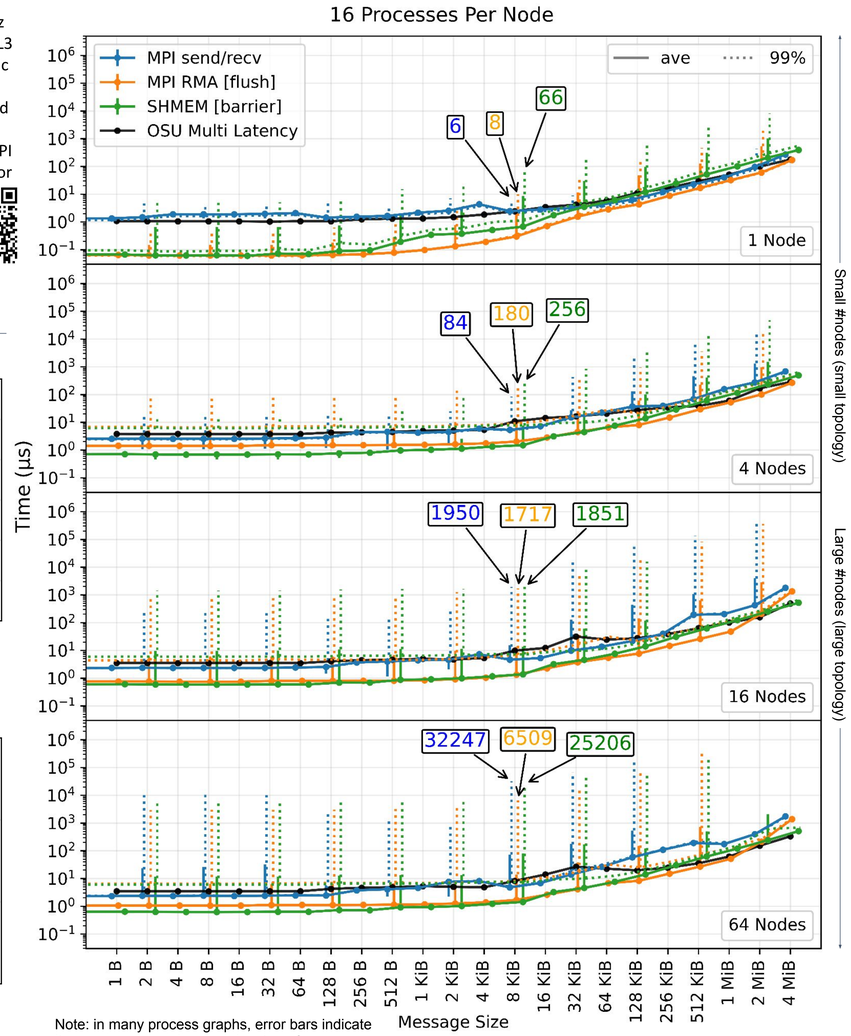
Experimental Setup

- NERSC Perlmutter supercomputer CPU partition: four-way 2.45GHz AMD Milan EPYC 7763 CPU nodes comprising of 64 cores, 256MB L3 cache, 512GB DDR4 memory with HPE Slingshot interconnect fabric (based on Dragonfly topology) connected to Cassini NIC
- GCC 11.2 compiler (PrgEnv-gnu/8.3.3), cray-mpich/8.2.25 (MPI) and cray-openshmemx/11.5.6 (SHMEM)
- SHMEM and MPI RMA use OFI libfabric as the low-level network API
- Latency data collected using NEVE, a proxy application generator for measuring network performance on distributed systems: https://github.com/sg0/neve/tree/rma

Latency Between Two Processes



Latency Between Many Processes



Note: in many process graphs, error bars indicate the maximum variation in network performance on an irregular topology produced by adding random edges proportional to 2% and 5% of overall edges in the original RGG.

Observations

- SHMEM and MPI RMA demonstrate the best sub-1us latency for small data sizes
- Two processes measurements not indicative of performance differences at scale (point-to-point performance exceeds RMA for a single process/node, but we see opposite trends with more processes/node)
- The impact of topology is seen from 8 nodes and beyond: gap between average and 99%-tile latencies surge with nodes
- OSU microbenchmark reports best case latencies with little variability, whereas gap between average and 99%-tile latencies surge to orders of magnitude with increasing the node counts (compare 99%-tile annotations between small and large nodes)
- For regular topologies on 1--64 nodes, both MPI RMA and SHMEM exceed the performance of point-to-point by up to 3.5x
- For irregular topologies, RMA exceeds the performance of point-to-point by up to about 2x for medium to large data
- On 8-64 nodes, SHMEM can exceed the performance of MPI RMA by up to about 2.5x for data sizes beyond 32KiB

functions	#instructions
shmem_char_put_nbi	192
MPI_Put	231
MPI_Isend	296
MPI_Irecv	204

- On a single node, SHMEM depicts 30% better performance up to 8KiB (39 less #instructions in SHMEM than MPI RMA)
- MPI RMA demonstrates better performance by up to 1.5x compared to SHMEM for larger data sizes
- MPI RMA should use XPMEM for single node large message transfers, but, our emulation (via Intel SDE) does not return any entry for XPMEM; SHMEM emulation reveals XPMEM usage

U.S. DEPARTMENT OF ENERGY | BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

For additional information, contact:
Sayan Ghosh | (509) 375-6323 | sayan.ghosh@pnnl.gov

8/18/20 | IR # PNNL-SA-190071

Pacific Northwest NATIONAL LABORATORY
www.pnnl.gov