Designing Shared Address Space MPI libraries in the Many-core Era

The Ohio State University

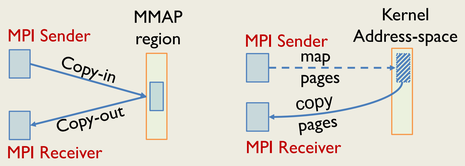
MVAPICH

Overview

Background

I. Intra-Node Communication in MPI

Two prevalent approaches to implement intra-node MPI communication.



POSIX Shared Memory	Kernel-Assisted (CMA)
No system call	System call overhead
Requires two copies	Requires one-copy
For small messages	For large messages

II. Process Shared Address-Spaces (XPMEM)

- Multiple processes share their address-spaces.
- The communication in realized via accessing the data from remote process' memory.
- XPMEM: A kernel module with user-level API for sharing address spaces among processes.

Motivation

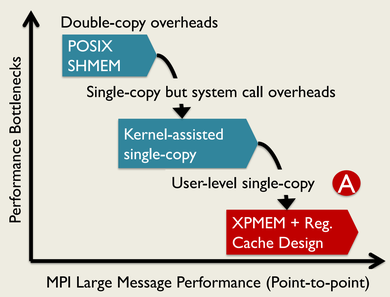
- XPMEM remote registration is costly
- Kernel-assisted zero-copy communication cause contention with increasing concurrency.
- Lack of true zero-copy reductions in MPI

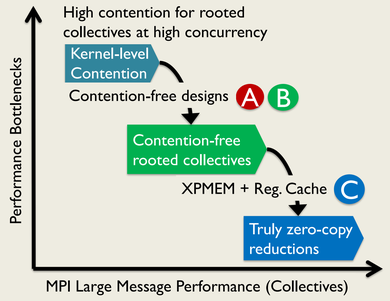
Contributions

- Efficient shared address-space based MPI point-to-point communication (A)
- Contention-free MPI collectives (B)
- Truly zero-copy MPI reduction collectives (C)

Summary

Identify fundamental overheads and propose new shared address-space based designs to improve the performance of MPI primitives on modern multi-/many-core systems



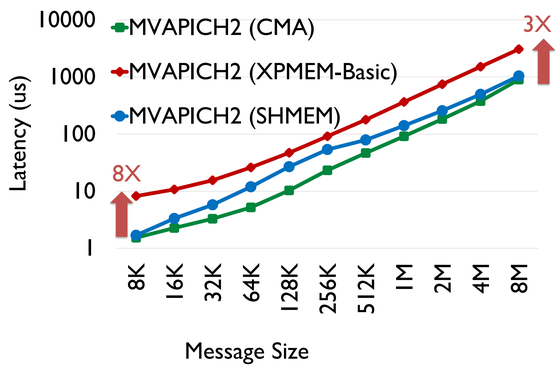


Research Challenges

A Point-to-point

Remote Memory Attach Overhead

- Remote memory attach overheads in XPMEM
- Each transfer requires registration of remote segment

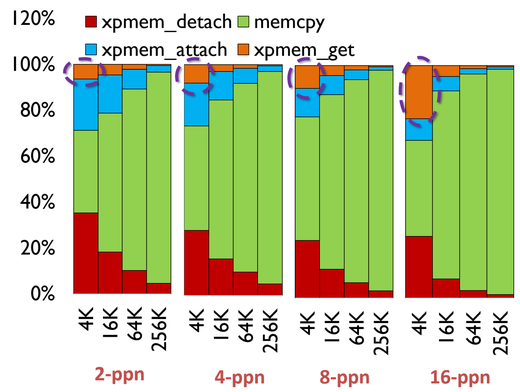


Two-process, osu_latency shows up to 3X degradation in latency.

B Contention-free Collectives

Contention Overheads in Kernel-assisted Collective Communication

- Process-level lock when accessing page-table → Contention
- Increased concurrency → more contention → degradation

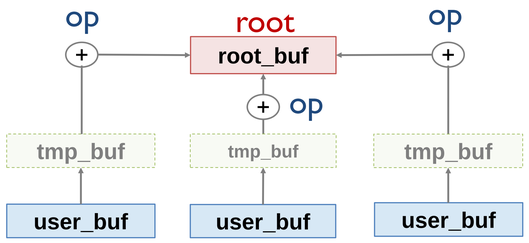


One-to-all collective communication benchmark on a dual-socket Broadwell system

C Zero-copy Reductions

Existing Limitation of Zero-Copy Reductions in MPI

- Reductions involve data-movement + computation
- Past research has shown kernel-assisted non-reduction zero-copy MPI collective communication using CMA, LiMIC, and KNEM.
- Existing approaches fail to deliver true zero-copy reductions.



root_data[i] = root_data[i] + user_buffer[i]

User_buf values need to be copied to temporary variables before they are added to root's vector → No Zero Copy

Proposed Designs

Registration Cache for XPMEM

- AVL tree based cache with O(logn) insertion/lookup cost.
- Registration cache based MPI rendezvous communication.

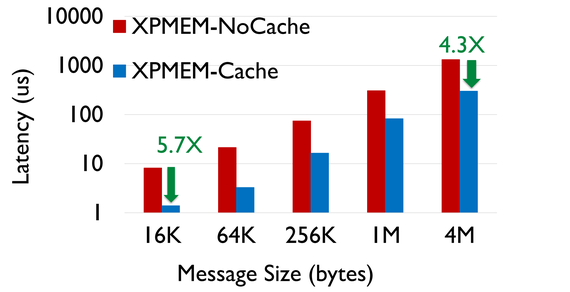
MPI Sender

```
/* create local buffer info pkt */
mv2_xpm_get_local_info(&info, sbuf);

/* share "local info" with remote rank
using any inter-process communication
mechanism */
```

MPI Receiver

```
/* retrieve peer's info object */
addr.apid = xpmem_get();
addr.offset = segment_base_addr;
dreg.lptr = xpmem_attach(addr, nbytes);
/* create rcache entry and insert */
mv2_xpm_avl_insert(peer_rank, dreg);
```



Up to 5.6X improvement over naïve design for osu_latency

Achieving Contention-free Collective Communication

- New user-level, shared address-space based design to realize the group communication.
- Direct Load/Store access to remote memory in user-space
- One-to-all benchmark with multiple concurrent readers.

Up to 4.2X better performance over baseline design

Achieving True Zero-copy Allreduce

1. Concurrent Intra-Node Reduction by all the Processes on Data Partitions with Same Index

Step-1: Parallel Intra-node Partitioned Reduce

2. Concurrent Inter-Node Allreduce by all the Processes on Same Index of Data Partitions

Step-2: Parallel Intra-node Partitioned Allreduce

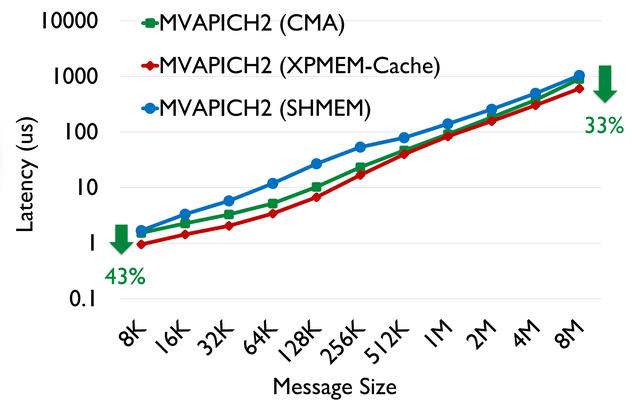
3. Copy Local Chunk (full result) to N-1 Processes (Bcast)

Step:3 Parallel Intra-node Partitioned Bcast

Performance Results

Impact of Enhanced MPI Point-to-Point Primitives on Xeon Broadwell

Two-process osu_latency benchmark



Up to 43% improvement over CMA based (single-copy) design.

Impact of Contention-free Collectives on Intel Xeon (Broadwell) and Xeon-Phi (KNL)

- Proposed contention-free design is compared against CMA
- It mitigates the contention overheads for rooted collectives e.g., one-to-all, all-to-one patterns

Performance improvement of up to 23X on Intel Broadwell, and 79X on Intel KNL.

Impact of True Zero-copy AllReduce

- Improved performance of miniAMR and AlexNet on multi-/many-core systems
- Up to 45% improvement over MVAPICH2
- Up to 23% improvement over Intel MPI

More Information

- Available in latest MVAPICH2X-2.3rc1 release
- http://mvapich.cse.ohio-state.edu/downloads/

This research is supported in part by National Science Foundation grants #CNS-1513120, #ACI-1450440, and #CCF-1565414.

References

1. *Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, D. K. Panda 32nd IEEE International Parallel & Distributed Processing Symposium (IPDPS '18), May 2018.*
2. Contention Aware Kernel-Assisted MPI Collectives for Multi/Many-core Systems S. Chakraborty, H. Subramoni, D. Panda 2017 IEEE International Conference on Cluster Computing, Sep 2017.

Jahanzeb Maqbool Hashmi, Dhabaleswar K. Panda (Advisor)
The Ohio State University