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

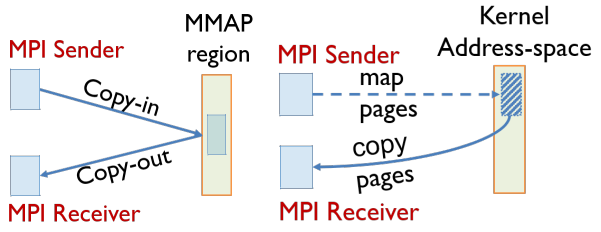
Jahanzeb Maqbool Hashmi and Dhabaleswar K. Panda (Advisor)
Department of Computer Science and Engineering,
The Ohio State University
hashmi.29@osu.edu, panda@cse.ohio-state.edu

ABSTRACT

The emergence of modern multi-/many-cores has put more emphasis on optimizing intra-node communication. Existing designs in MPI libraries that work on the concept of distributed address spaces incur the overhead of intermediate memory copies to stage the data between processes, leading to severe performance degradation especially on emerging many-core architectures like Intel Skylake and IBM OpenPOWER. This work proposes a high-performance "shared address space"-based MPI point-to-point and collective communication designs using XPMEM. We first characterize the bottlenecks associated with XPMEM based communication and propose new designs for efficient MPI large message communication. Then we propose novel collective designs that are contention-free and offer true zero-copy reduction operations. The proposed designs are evaluated on different multi-/many-core architectures using various micro-benchmarks and application kernels such as MiniAMR and AlexNet DNN training on CNTK. The proposed designs have shown significant performance improvement over state-of-the-art available in MPI libraries.

1 INTRODUCTION

The existing designs for intra-node communication in a process based MPI library mainly employ two approaches for inter-process communication: 1) POSIX shared memory-based double-copy, and 2) kernel-mapped single-copy. While the former can be good for small messages, it can prove detrimental to large message performance. The second approach works well for large messages because of single-copy design as the sender process maps the source buffer in kernel address space while the receiver process copies the data to the destination buffer. Figure 1 demonstrates the basic working of both the approaches. XPMEM [5] is another approach that allows a process to map a segment of another process' address space allowing it to directly access remote data.



(a) POSIX SHMEM based double-copy (b) Kernel-assisted single-copy

Figure 1: Existing MPI Intra-node Communication Designs

Researchers have proposed MPI point-to-point and collective communication operations using the aforementioned approaches. However, these designs incur several performance bottlenecks. For instance, XPMEM based MPI point-to-point communication requires attach/detach of remote buffer for each message communicated between a pair of processes. Figure 2 details the overheads associated with different XPMEM API functions. It can be seen that for medium messages up to 64% of the overall communication time is spent in registration overheads.

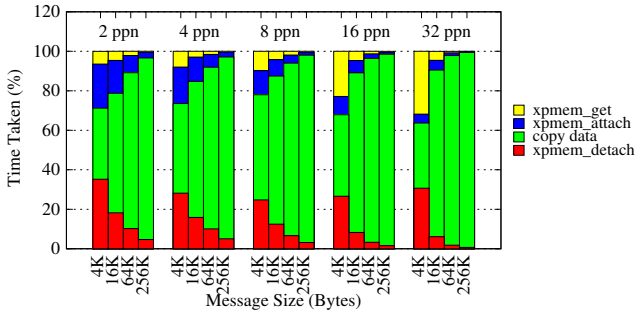


Figure 2: XPMEM API functions and their relative costs with different message sizes and PPN on Broadwell. For small messages, XPMEM overheads dominate the total time.

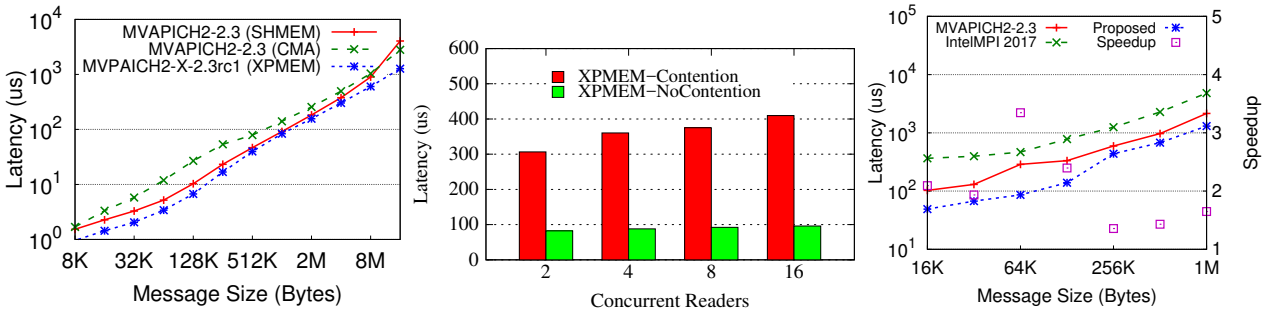
The collective operations in MPI that are implemented using basic point-to-point operations such as Send/Recv, suffer from unnecessary overheads such as tag-matching, rendezvous hand-shake, etc. On the other hand, POSIX SHMEM based collective implementations suffer from cache-thrashing and double-copy overheads. There has been some research to implement direct zero-copy collectives using kernel-assisted memory mapping by using LiMiC [2], and KNEM [4]. However, these approaches suffer from various performance limitations. For instance, 1) earlier research [1] has demonstrated that rooted collectives exhibiting one-to-all or all-to-one communication patterns, such implementation cause contention at the root process due to process-level lock contention in the kernel, and 2) for reduction collectives such as MPI_Reduce and MPI_Allreduce, these approaches do not eliminate the copy overhead as the data from remote processes must be brought into the local process's memory (usually a temporary buffer) before the reduction operation can be performed.

To remedy all these challenges, we proposed efficient shared address-space based designs that mitigate these overheads and achieve the best performance for MPI point-to-point and collective communication.

2 PROPOSED DESIGNS

We describe three designs to support efficient shared address-space based communication in MPI. The proposed designs are implemented in MVAPICH2X-2.3rc1 and are publicly available to download. The proposed designs are evaluated on Intel Broadwell, Intel KNL, and IBM OpenPOWER systems.

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



(a) Two process latency on Broadwell (b) Contention-free One-to-All Benchmark (c) Proposed Allreduce on 32 Broadwell Nodes (896 Processes)

Figure 3: Microbenchmark results of proposed point-to-point and reduction collectives designs on different architectures

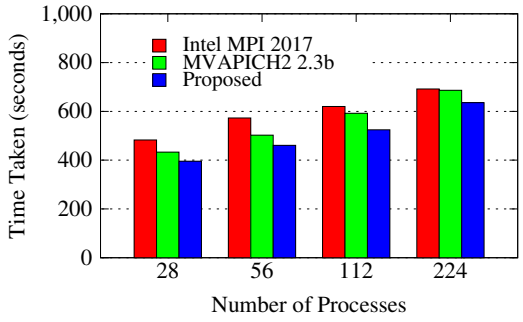


Figure 4: Training Time of AlexNet neural network using ImageNet dataset (processes per node = 28, iterations = 50)

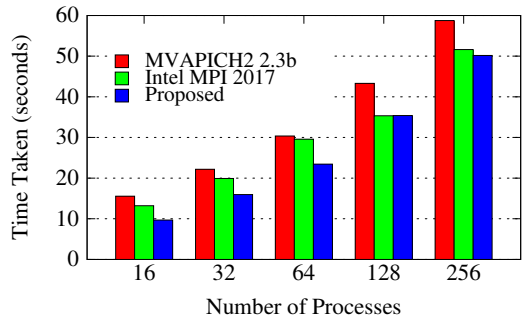


Figure 5: Performance of MiniAMR with the proposed designs on a Broadwell cluster (16 processes per node)

(1) **MPI Point-to-point Operations:** First, we proposed efficient XPMEM based design for MPI rendezvous communication to alleviate the bottlenecks of naive XPMEM based communication.

(2) **Contention-free MPI Collectives:** Second, to mitigate the contention overheads posed by rooted MPI collectives, we proposed XPMEM based direct collectives augmented by "registration-cache" based scheme [3] to avoid unnecessary overheads.

(3) **Zero-copy MPI Reductions:** Lastly, we designed truly zero-copy MPI_Allreduce and MPI_Reduce operations to take advantage of the vast parallelism of multi-/many-cores via multi-leader based schemes and avoid unnecessary staging of intermediate application buffers by using XPMEM based communications.

3 PERFORMANCE EVALUATION

We evaluate point-to-point and collectives performance using OSU Micro-benchmarks. We used MiniAMR and CNTK AlexNet training example to demonstrate the benefits of proposed designs at application level.

(1) **MPI Point-to-point Operations:** Figure 3(a) depicts the result of proposed design (1) where it outperforms the CMA based communication by up to 33% for large messages.

(2) **Contention-free MPI Collectives:** Figure 3(b) shows the result of proposed design (2) where our contention-free design, augmented by registration cache schemes shows up to 4.2X benefits over naive design.

(3) **Zero-copy MPI Reductions:** Figure 3(c) shows the result of proposed design (3). As we see that our zero-copy Allreduce achieves up to 3.5X improvement on 896 processes. Figure 4 and Figure 5 shows the application level evaluation of proposed Allreduce on Deep Learning and AMR workloads. As we can see that the proposed design shows up to 20% and 23% improvement for DL and AMR kernels respectively.

4 FUTURE WORK

We plan to design and enhance MPI datatypes and MPI RMA for shared address-space based paradigm.

REFERENCES

[1] S. Chakraborty, H. Subramoni, and D. Panda. 2017. Contention Aware Kernel-Assisted MPI Collectives for Multi/Many-core Systems . In *2017 IEEE International Conference on Cluster Computing*.

[2] Dhanraj, Vijay. 2012. *Enhancement of LiMIC-Based Collectives for Multi-core Clusters.* Master's thesis. The Ohio State University.

[3] Jahanzeb Hashmi, Sourav Chakraborty, Mohammadreza Bayatpour, Hari Subramoni, and Panda Dhabaleswar K. 2018. Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores . In *32nd IEEE International Parallel and Distributed Processing Symposium (IPDPS '18)* .

[4] Ma, Teng and Bosilca, George and Bouteiller, Aurelien and Goglin, Brice and Squyres, Jeffrey M and Dongarra, Jack J. 2011. Kernel-assisted Collective Intra-node MPI Communication among Multi-core and Many-core CPUs. In *Parallel Processing (ICPP), 2011 International Conference on.* IEEE, 532–541.

[5] Woodacre, M and Robb, D and Roe, D and Feind, K. 2003. The SGI Altix 3000 Global Shared-Memory Architecture–White paper, Silicon Graphics. (2003).