SANDIA REPORT
SAND2024-11806
Printed September 2024

Sandia National Laboratories

Implementing One Sided Partitioned Communication in Open MPI

Matthew G. F. Dosanjh

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

This report introduces partitioned communication, a new MPI 4.0 interface that enables early bird communication by overlapping communication and computation. By partitioning messages into smaller sub-messages, MPI can start partial data transfers early. Performance studies show that the RMA implementation outperforms the Persistent implementation, despite some constraints. This report details a new opt-in RMA implementation, offering a high-performance option for partitioned communication that imposes some additional limitations.

ACKNOWLEDGEMENTS

CONTENTS

LIST OF FIGURES

This page left blank

ACRONYMS AND TERMS

Acronym/Term	Definition
MPI	Message Passing Interface
RMA	Remote Memory Access
osc	One sided communication

1. INTRODUCTION

Partitioned communication is a novel interface introduced in MPI 4.0, designed to facilitate early bird communication. This new interface allows for the overlap of communication and computation by enabling MPI to initiate partial data transfers early. It achieves this by partitioning a message into smaller sub-messages, providing an interface for users to specify which sub-messages are ready to send, and an interface to determine if a sub-message has arrived at the target.

Several implementations of partitioned communication have been developed, with Sandia playing both direct and indirect roles in their creation. The initial research implementation, known as Finepoints, was developed over Remote Memory Access (RMA) as an external library [GDL+19]. Following this, MPIPCL was designed in collaboration with Sandia by researchers at the University of Tennessee Chattanooga and Auburn University. MPIPCL was built as an external library on top of the MPI Persistent point-to-point interface. Finally, the Open MPI implementation consists of the 'part' component, which includes the user interface and selection of the underlying implementation, and the 'persist' module, which implements partitioned communication in a manner similar to MPIPCL [WPSS+21].

In a performance study comparing the RMA and Persistent implementations, it was found that the RMA implementation exhibited higher performance for send-side partitioning [DWS+21]. The initial implementation was based on the Persistent interface due to functional reasons; the RMA has restrictions on the initialization phase that require the initialization functions to be blocking. Consequently, the first implementation focused on providing specification compliance. This report describes the development of an opt-in RMA implementation, allowing users to choose a high-performance implementation that imposes additional limitations.

2. DESIGN AND IMPLEMENTATION

This section details the design and implementation of the 'direct' module for the part component.

2.1. Design Considerations

There are several design considerations that distinguish our implementation from the original external library described in the Finepoints paper.

Firstly, we utilize a single communicator and two separate windows: a data window and a flag window. The communicator is used to select the nodes for the sender and receiver, as windows can only be instantiated across entire communicators. In the new communicator, we assign the rank of 0 to the sender and 1 to the receiver. The data window operates as expected: the sender opens a window associated with a buffer on the receiver and initiates a put operation for each partition that is ready. Once all partitions are marked as ready, a second window comes into play. This differs from the original implementation in that a flag is needed on both sides. On the receiver side, this flag indicates completion, while on the sender side, it serves as a 'ready-to-receive' signal. The sender sets the flag on the receiver after the last put operation is called. Conversely, the receiver sets the flag on the sender when MPI Start is called, allowing the receiver to control when the data buffer can be written to. This is required for specification compliance, although the Finepoints implementation predates this specification.

Secondly, the progress function is crucial for supporting delayed data transfer as required by the specification. The MPI_Pready function simply sets a ready flag, which is then checked by the progress function to handle the MPI put call. Additionally, the structure of completion in Open MPI is funda- mentally different from the original Finepoints implementation; in this design, completion is managed within the progress function.

2.2. Limitations

As mentioned above there are a couple limitations to this approach. First this implementation requires that the MPI Precv init and MPI Psend init calls block for completion. This is necessitated by underlying calls MPI Comm create group and MPI Win create, both of which are blocking collectives on both sender and receiver.

A minor limitation is that the current flag system will only signal MPI Parrived after the full message is received. This is still compliant with the specification, but only allows for benefit from send side partitioning. However, this decision was made based on the poor performance of an RMA implementation with receive side partitioning.

3. PERFORMANCE

To evaluate the performance of this we used a modified version of the benchmark from the Finepoints paper. This benchmark highlights perceived bandwidth, by delaying a thread to simulate noise, and measuring the time from when that thread completes to the time the full message arrives. This was run 5 times, for 100 iterations on the Manzano cluster using the osc RDMA module each run was done with 32 partitions.

Figures 1,2 highlight the transition point where RMA starts to outperform Persistent. When using the persistent implementation, there's extra overhead associated with each partition, due to matching and completion, while the RMA implementation aggregates the completion overhead and avoids overheads like matching. The cost is that the single This transition point indicates that the communication and overhead of persistent point-to-point calls can no longer be hidden by the noise. For these experiments this happens at 8MiB messages.

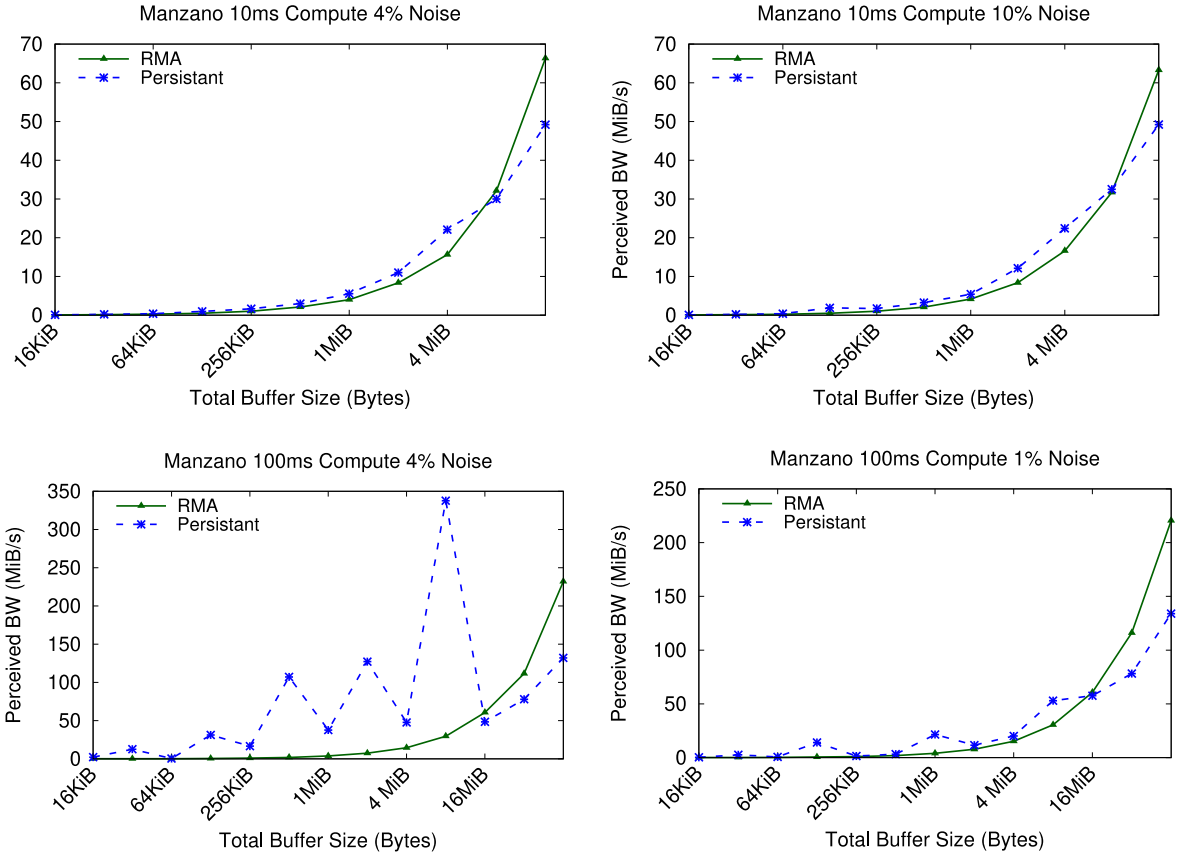

Figure 1. Performance evaluation.

4. CONCLUSIONS AND FUTURE WORK

This new module provides performance in line with the original research that inspired partitioned communication. While there remain some limitations to this implementation, this provides users a performance implementation alongside a spec-compliant implementation.

This work opens two significant avenues for future research and development. First, to fully leverage the benefits of MPI RMA, we need to develop a non-blocking function that simplifies its usage. This function should allow for the simultaneous creation of multiple windows, create pair-wise windows to reduce complexity, and use tag matching to avoid conflicts. Such a function would enable the implementation of spec-compliant partitioned communication over RMA, making it more accessible and practical for a broader range of applications.

Second, previous studies have shown that both RMA and Persistent implementations have their strengths, depending on the level of receive-side partitioning. While providing the ability to choose between these implementations is a step forward for the performance of partitioned communication in Open MPI, automatic selection could further enhance usability and efficiency. Future work should focus on creating a combined module that automatically selects the most appropriate implementation based on the number of receive-side partitions, optimizing performance without requiring manual intervention.

REFERENCES

[1] Matthew GF Dosanjh, Andrew Worley, Derek Schafer, Prema Soundararajan, Sheikh Ghafoor, Anthony Skjellum, Purushotham V Bangalore, and Ryan E Grant. Implemen- tation and evaluation of mpi 4.0 partitioned communication libraries. Parallel Computing, 108:102827, 2021.

[2] Ryan E Grant, Matthew GF Dosanjh, Michael J Levenhagen, Ron Brightwell, and An- thony Skjellum. Finepoints: Partitioned multithreaded mpi communication. In High Performance Computing: 34th International Conference, ISC High Performance 2019, Frankfurt/Main, Germany, June 16–20, 2019, Proceedings 34, pages 330–350. Springer, 2019.

[3] Andrew Worley, Prema Prema Soundararajan, Derek Schafer, Purushotham Bangalore, Ryan Grant, Matthew Dosanjh, Anthony Skjellum, and Sheikh Ghafoor. Design of a portable implementation of partitioned point-to-point communication primitives. In 50th International Conference on Parallel Processing Workshop, pages 1–11, 2021.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Technical Library	1911	sanddocs@sandia.gov

This page left blank

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.