# Design and Implementation of Key Proposed MPI-3 One-Sided Communication Semantics on InfiniBand

Sreeram Potluri, Sayantan Sur, Devendar Bureddy, and Dhabaleswar K. Panda

Department of Computer Science and Engineering, The Ohio State University
{potluri, surs, bureddy, panda}@cse.ohio-state.edu

**Abstract.** Simultaneous use of powerful system components is important for applications to achieve maximum performance on modern clusters. MPI-2 had introduced one-sided communication model that enables for better communication and computation overlap. However, studies have shown limitations of this model both in the context of applications and higher-level libraries. As part of MPI-3 effort, the Remote Memory Access group has proposed several extensions to the existing one-sided communication interface to address these limitations. In this paper, we present design, implementation and evaluation of some of the key one-sided semantics proposed for MPI-3 over InfiniBand, using the MVAPICH2 library.

## 1 Overview

High-end computing systems have seen a tremendous growth in recent years, driven by advances in processor, network and accelerator technologies. As the capabilities of different components in a system increase, it is important for scientific applications to utilize all these components concurrently to achieve maximum performance. Programming models hold the key in enabling such usage. MPI had introduced non-blocking message passing and one-sided communication semantics that enable overlap between computation and communication. Earlier work [5] has shown how one-sided communication semantics can achieve superior overlap in applications than the message passing semantics. However, their adaptation has been limited because of the overheads imposed by synchronization operations in MPI-2 and a mismatch with real-world use cases for one-sided communication. Other one-sided models like Global Address-Space Languages, and Global Arrays have failed to utilize the portable nature on MPI because of these limitations. As part of the MPI-3 effort, the Remote Memory Access (RMA) group [2] has proposed several extensions to the existing model that promise to address many of these limitations [1].

Modern networks have played an indispensable role in scaling modern computing clusters. InfiniBand is a commodity interconnection network which has gained acceptance by the HEC community. It is the primary interconnect in around 40% of the Top500 supercomputing clusters in the world. The Remote Direct Memory Access (RDMA) operations offered by InfiniBand free the processor from managing data transfers. This allows communication libraries to achieve higher performance and better overlap.

The proposed MPI-3 one-sided interface promises to address the limitations of the MPI-2 one-sided interface. The newer additions include dynamic window creation, light weight synchronization (local and remote) and variety of other communication operations. However, in order for wide spread acceptance of this proposed interface, its performance advantages need to be clearly highlighted. We believe that this is a strong motivation for designing and implementing some of the key MPI-3 interfaces on a widely used commodity platform. In this work, we present an analysis of a key subset of the proposed MPI-3 extensions and through experimental evaluation we establish that they efficiently solve several issues faced by the MPI-2 standard. Our design of the proposed semantics is integrated in the MVAPICH2 library [3], to demonstrate a working prototype in an open-source production MPI library. To the best of our knowledge, this is the first design and implementation of the proposed MPI-3 one-sided interface. The semantics implemented in this work are highlighted in Figure 1.

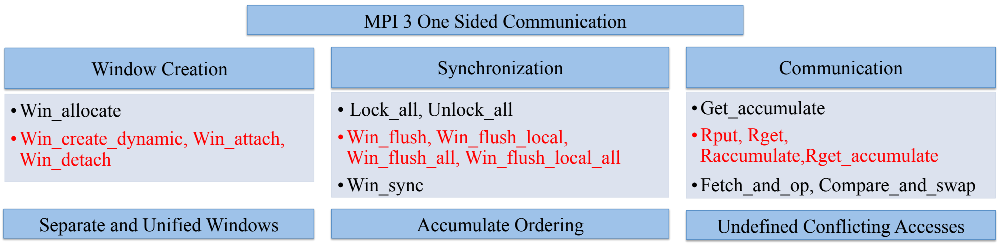

MPI 3 One Sided Communication
Window Creation
• Win_allocate
• Win_create_dynamic, Win_attach, Win_detach
Synchronization
• Lock_all, Unlock_all
• Win_flush, Win_flush_local, Win_flush_all, Win_flush_local_all
• Win_sync
Communication
• Get_accumulate
• Rput, Rget, Raccumulate,Rget_accumulate
• Fetch_and_op, Compare_and_swap
Separate and Unified Windows
Accumulate Ordering
Undefined Conflicting Accesses

**Fig. 1.** Proposed MPI-3 One-Sided Communication Standard Extensions

## 2 Design and Evaluation

In this section, we provide a brief overview of each of the semantics addressed in this work and their implementation highlights. A detailed description about the implementation and evaluation can be found in our technical report [6].

**Dynamic Windows:** A window defines the memory to be used for communication in the one-sided model. In MPI-2, the location and size of memory attached to a window is specified during window creation and cannot be changed at a later point of time. This is a misfit in the case of applications and programming models with dynamic memory requirements. MPI-3 allows "dynamic" windows where each process can asynchronously attach or detach memory from a window. Implementation of one-sided communication operations over RDMA requires exchange of buffer registration information. For MPI-2, this is usually done during the window creation phase. In the case of dynamic windows, such an exchange is required each time an access happens to a newly attached buffer. However, as multiple accesses happen to each buffer, this cost can be amortized efficiently. Through micro benchmark evaluation, we show that performance of dynamic windows is as good as that of static windows. The performance comparison of Put latency is shown in Figure 3(a). A complete set of results can be found in the technical report.

```
MPI_Win_lock
for i in 1, N
    MPI_Get (ith Block)
end for
MPI_Win_unlock

Compute (N Blocks)

MPI_Win_lock
for i in 1, N
    MPI_Put (ith Block)
end for
MPI_Win_unlock
```

(a) No Overlap

```
MPI_Win_lock
for i in 1, N
    MPI_Get (ith Block)
end for
MPI_Win_unlock

MPI_Win_lock
for i in 1, N
    Compute (ith Block)
    MPI_Put (ith Block)
end for
MPI_Win_unlock
```

(b) Overlap using Lock-Unlock

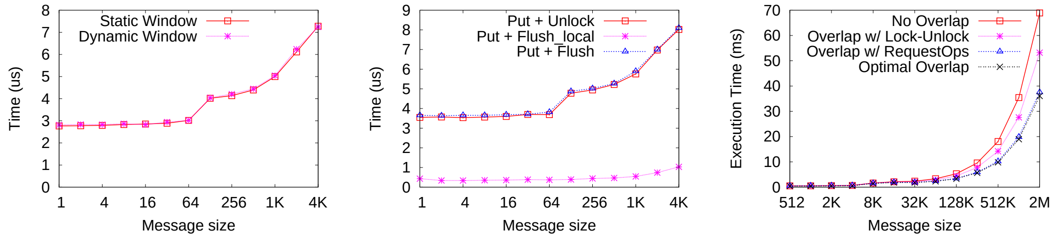
```
MPI_Win_lock
for i in 1, N
    MPI_Rget (ith Block)
end for

MPI_Wait_any (get requests)
while a get request j completes
    Compute (jth Block)
    MPI_Rput (jth Block)
    MPI_Wait_any (get requests)
end while
MPI_Wait_all (put requests)
MPI_Win_unlock
```

(c) Overlap using Request Operations

**Fig. 2.** Get-Compute-Put on N Blocks of Data

**Flush Operations:** All communication operations in MPI one-sided interface are non-blocking. In MPI-2, their completions (both local and remote) are bound to synchronization operations. This is heavy-weight. MPI-3 addresses this issue through flush operations, separating local completion from remote completion. Ensuring local and remote completions of different operations (writes, reads and atomics) in InfiniBand have different requirements and costs [4]. The flush semantics provide flexibility to match the completion of different one-sided communication operations to the completion requirements in InfiniBand and hence provide better efficiency. A comparison of Put completion times using Lock/Unlock and Flush semantics is shown in Figure 3(b).

**Request-based Operations:** Request-based operations provide an easy mechanism to wait for completion of *specific* operations. This allows for much finer grained overlap compared to flush or other synchronization calls, which wait for completion of all operations to a target or on a window. Figure 2 presents pseudo-code for three versions of a Get-Compute-Put benchmark which fetches N blocks of data from remote memory, computes on them and writes them back. Figure 2(a) shows a code without any overlap. The three phases: get, compute and put can be pipelined to overlap computation and communication. Figure 2(b) and (c) show overlapped versions using MPI-2 Semantics and using Request-based operations respectively. The performance results are shown in Figure 3(c). We see that request-based operations provide close to optimal overlap.



(a) Put Latency w/ Dynamic Windows (b) Put Latency w/ Flush Operations (c) Get-Compute-Put w/ Request Ops

**Fig. 3.** Performance using MPI-3 One-sided Semantics

## 3 Conclusion

In this paper, we presented design, implementation and evaluation of a key subset of newly proposed one-sided interface. Through micro-benchmark evaluation, we have shown that the newly proposed interfaces can provide improved performance over the MPI-2. In the near future, we would like to show these benefits using a real-world application, re-designing it to new the new functions and semantics.

## 4 Acknowledgments

This research is supported in part by U.S. Department of Energy grants #DE-FC02-06ER25749 and #DE-FC02-06ER25755; National Science Foundation grants #CCF-0833169, #CCF-0916302, #OCI-0926691 and #CCF-0937842.

## References

1. MPI-3 RMA. http://meetings.mpi-forum.org/secretary/2008/03/slides/mpi3-rma-summary-3-10.pdf
2. MPI-3 RMA Working Group. `http://meetings.mpi-forum.org/mpi3.0_rma.php`
3. MVAPICH2: MPI over InfiniBand, 10GigE/iWARP and RoCE. http://mvapich.cse.ohio-state.edu/

4. InfiniBand Trade Association: InfiniBand Architecture Specification, Release 1.2 (October 2004)
5. Potluri, S., Lai, P., Tomko, K., Sur, S., Cui, Y., Tatineni, M., Schulz, K., Barth, W., Majumdar, A., Panda, D.K.: Quantifying Performance Benefits of Overlap using MPI-2 in a Seismic Modeling Application. In: International Conference on Supercomputing (ICS'10) (2010)
6. Potluri, S., Sur, S., Bureddy, D., Panda, D.K.: Design and Implementation of Key Proposed MPI-3 One-Sided Communication Semantics on InfiniBand. Technical Report OSU-CISRC-7/11-TR19, The Ohio State University (2011)